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

Zhefan Xu, Di Deng, 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 it 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 frontier-based planner and receding horizon next-best-view planner in terms of exploration time, path length, and computational time.**

***Index Terms*—Search and Rescue Robots, Motion and Path Planning; Mapping; Incremental Sampling; Unmanned Aerial Vehicle**

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.

While recent sampling-based approaches [1] [23] [18] [17] have already shown success in autonomous exploration, the randomness of sampling and computation still limits the performance. Besides, most of these approaches have not explicitly considered safety to avoid dynamic obstacles during the exploration. There exist different methods to solve the exploration problem. Generally, the problem can be viewed as how to determine a series of informative sensor positions [5]. In early work in frontier exploration, a robot successfully gains knowledge of the unknown region by visiting the border [24]. With its high computational efficiency and the various reliable information gain formulations, the sampling-based method is preferred for the UAV exploration. Typically, these methods [1] [23] [18] [17] 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) [12] 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) [11] can be more suitable for reducing computational cost and determining the informative viewpoints based on the accumulated knowledge. Even though some methods continuously add nodes to the tree branches [17] or reuses historical information [23] to overcome these problems, they still lack the flexibility to quickly generate the suboptimal alternative paths to avoid dynamics obstacles.

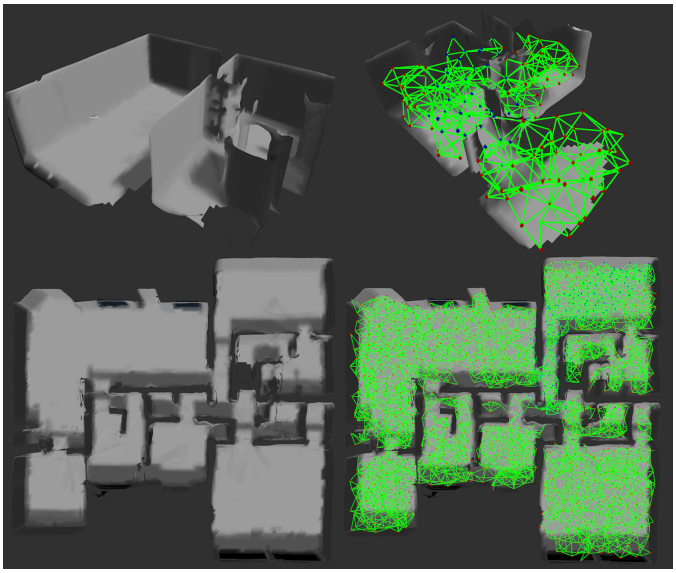


Fig. 1: Visualization of an explored map (left) and incrementally constructed roadmap (right) for a large office environment. Green lines and red dots represent edges and nodes, respectively. Note that blue dots are the nodes whose exploration utilities (gains) need to be updated.

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 in a short time 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

Xu, Deng, and Shimada are with the Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA., zhefanx@andrew.cmu.edu

trajectory execution time and maintain the tolerant distance to obstacles. We evaluate our algorithm in simulation experiments using the Robot Operating System [20]. 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 the roadmap visualization.

II. Related Work

There are three major categories of the autonomous exploration algorithms: the frontier-based method, the information-theoretic method, and the sampling-based method. The earliest frontier exploration algorithm makes the robot visit the closest frontiers to gain knowledge of an unknown region [24], proving the success in the ground robot. [4] extends this algorithm to support fast flying speed in UAV exploration, and [6] modifies this method to achieve 3D reconstruction by a low-cost 2D sensor. In contrast, the information-theoretic method [2] [3] focuses on reducing the current map entropy and use the formulated objective function to guide the exploration based on uncertainty. Their results indicate a significant reduction of exploration time compared to the frontier-based method. Unlike the former categories, the sampling-based method defines the information gain more flexibly, thus having the broad applicability in different tasks and environments. Besides, as the computation of the sampling-based methods does not increase significantly with the robot dimension, it is preferred in the UAV application. Other methods like the data-driven approach [13] [19] have also been experimented with but are hard to be generalized for different environments.

The fundamental idea of the sampling-based method is gathering information from sampled viewpoints. [5] first proposes a method to determine a series of best viewpoints for sensor placement to reconstruct the geometry of objects. Later, [7] and [21] brings the next-best-view (NBV) idea into the planning field. They introduce the sample evaluation by the cost-utility function to balance the traveling distance and the expected information gain. Recently, [1] incorporate the information gain into the RRT branch evaluation in their receding horizon next-best-view planner (RH-NBV). 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. Their results show the shorter exploration time compared to the frontier-based exploration.

RH-NBV planner brings a reliable solution to UAV exploration and has been improved by later researchers. To improve the sampling efficiency and avoid the planner from getting trapped in the local minima, [23] 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, [18] combines RH-NBV with the frontier-based algorithm to prevent early termination in local minima. The frontiers are cached for determining the exploration goals, and the cached nodes can help avoid recalculating the information gain when sampling in the nearby region. To avoid discarding the rest of nodes not in the best branch, [17] continuously grows and maintains the tree with rewiring for path refinement. 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, [16], [22], and [15] consider the SLAM uncertainty in their planning steps.

While most sampling-based algorithms with RRT backbone have proven successful in autonomous exploration, their ineffective reuse of information makes them not optimal. As the exploration is an iterative and multi-step process, the multi-query planner PRM has the better suitability. Also, the RRT-based planner has difficulty searching for the suboptimal alternative trajectories quickly. To solve these issues, we adopt the incremental PRM as our backbone to achieve both better region coverage with effective computations reuse and faster replanning speed for dynamic obstacle avoidance.

III. Problem Description

A bounded environment, $V_{\text{b}} \subset \mathbb{R}^3$, 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, $V^{\text{init}}_{\text{mapped}}$, 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, $V^{\text{init}}_{\text{mapped}}$, 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, $O^{\text{traj}}_{\text{dynamic}}$, 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 $O^{\text{traj}}_{\text{dynamic}}$ is detected, the robot must replan in a reasonable time to avoid expected collisions.

IV. Proposed Method

In each planning iteration, our method can be divided into four steps shown in Fig. 2 left: (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 last iteration while ensuring that the nodes are spread out evenly in space, and this process is visualized in Fig. 3. Then, in the second step, the information gain of newly added nodes and the old nodes is evaluated or updated. After that, 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 obstacles.

A. Roadmap Construction

The incremental PRM needs to add nodes when a new region is observed. To better estimate the exploration utility of positions

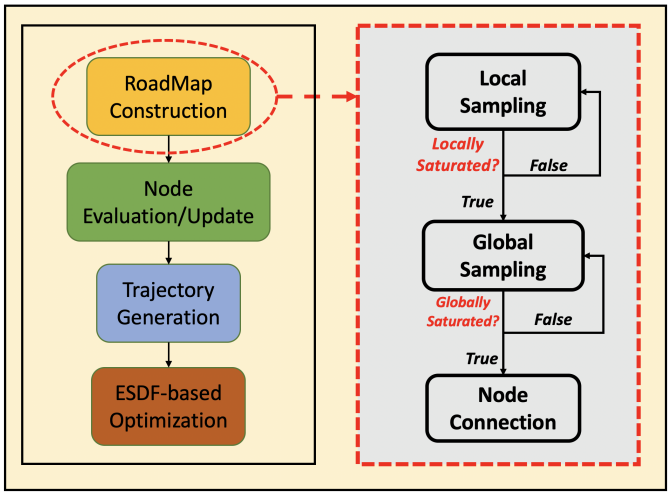


Fig. 2: Planner overview. Left: The planner follows an iterative process with four steps. Right: Roadmap construction includes a two-stage (local and global) sampling process and a node connection process after sampling.

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, as presented in Fig. 2 right. 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 12). 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 $\mathcal{N}_{\text{f}}$ (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 [1], 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

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 $\quad \mathcal{N}_{\text{fail}} \leftarrow 0$;
6 $\quad$ **while** *true* **do**
7 $\quad\quad$ **if** $\mathcal{N}_{fail} > \mathcal{N}_{max}$ **then**
8 $\quad\quad\quad \mathcal{S}_{\text{cond}} \leftarrow true$;
9 $\quad\quad\quad$ **break**;
10 $\quad\quad \mathcal{R}_{\text{s}} \leftarrow$ **getSampleRegion**(ξ_0);
11 $\quad\quad n_{\text{new}} \leftarrow$ **randomConfig**$(\mathcal{R}_{\text{s}})$;
12 $\quad\quad d_{\text{nn}} \leftarrow$ **disToNearestNeighbor**(n_{new});
13 $\quad\quad$ **if** $d_{nn} < d_{th,min}$ **then**
14 $\quad\quad\quad \mathcal{N}_{\text{fail}} \leftarrow \mathcal{N}_{\text{fail}} + 1$;
15 $\quad\quad$ **else**
16 $\quad\quad\quad$ Add n_{new} to $\mathcal{R}$;

Algorithm 2: Node Connection

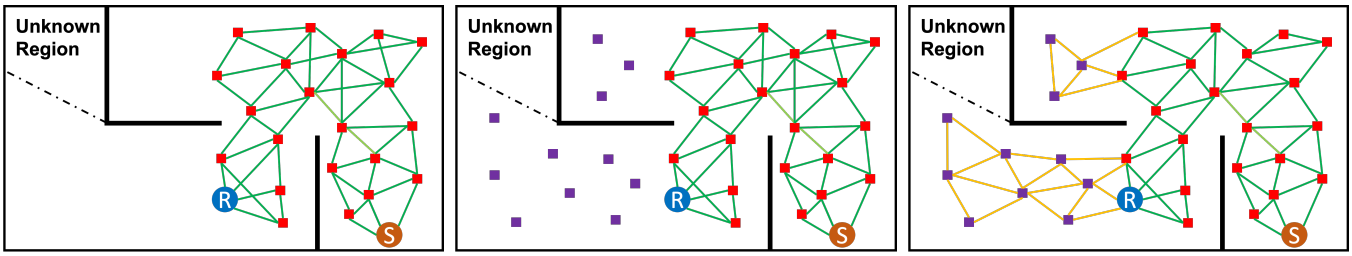
1 **for** *Node* n_i *in Roadmap* $\mathcal{R}$ **do**
2 $\quad Neighbors \leftarrow$ **neighborHood**(n_{i});
3 $\quad$ **for** n_N *in Neighbors* **do**
4 $\quad\quad \mathcal{C}_{\text{collision}} \leftarrow$ **checkCollision**$(n_{\text{i}}, n_{\text{N}})$;
5 $\quad\quad \mathcal{C}_{\text{dis}}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

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:

$$\textbf{Gain}(n) = w_{\text{n}} \cdot \mathcal{U}_{\text{n,tot}} + w_{\text{f}} \cdot \mathcal{U}_{\text{f,tot}} + w_{\text{s}} \cdot \mathcal{U}_{\text{s,tot}}, \qquad (1)$$

where $w_{\text{n}}, w_{\text{f}}$, and w_{s} denotes the weights for normal unknown, frontier unknown, and surface unknown voxels, respectively. $\mathcal{U}_{\text{n,tot}}, \mathcal{U}_{\text{s,tot}}$, and $\mathcal{U}_{\text{f,tot}}$ 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.

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



(a) Robot completes the previous trajectory (b) Incremental new node sampling (c) New node connection

Fig. 3: Illustration of the incremental roadmap construction. The robot's current position and start position are denoted by R and S. After completing the previous trajectory, new nodes (purple rectangles) are incrementally sampled, and new edges (yellow lines) are added to update the roadmap.

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, $0 < \lambda < 1$:

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

where $n_{\max}$ 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, $\lambda \cdot \textbf{Gain}(n_{\max})$. 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:

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

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:

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

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

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

where $\mathbf{n}$ represents the waypoints in the trajectory. Note that Eq. 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 Eq. 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 (Eq. 5c), where 1 indicates the condition being satisfied and 0 otherwise. Eq. 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 so that nodes do not overlap their corresponding gain voxels. The local

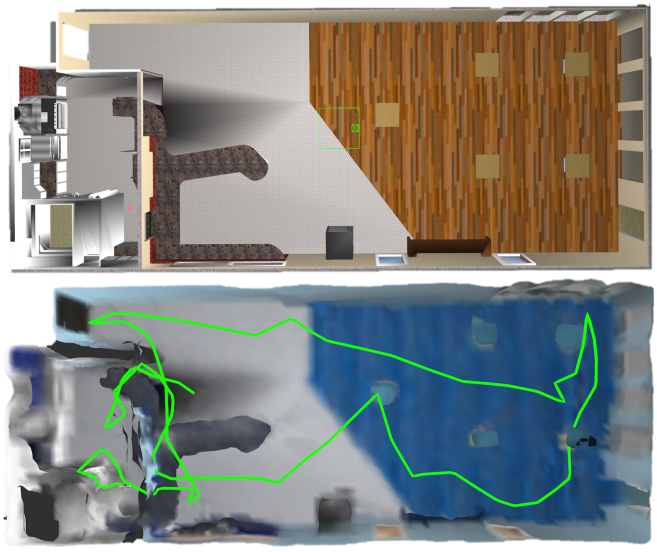


Fig. 4: Visualization of the Cafe (Small) environment with its fully explored map and exploration trajectory.

optimization is reflected in Eq. 5b, where each node has to be in the local region of the initial value.

V. Implementation Details

We use the Octomap [8] package for planner implementation. In the node connection step (Alg. 2), we define a neighborhood as the $\mathcal{K}$ nearest neighbors for the given node. As [1] suggests, the sensor range used for the planner to calculate the information gain is less than the actual maximum sensor range, $d_{\text{planner}} < d_{\max}$. The goal node's yaw angle optimization is similar to [23]. 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 [14] package to generate the TSDF/ESDF map. The optimization formulation is implemented as a nonlinear minimization problem in NLopt [9]. 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. Experiments 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 implemented in the Robot Operating System (ROS) [20] running on Intel i7-7700HQ at 2.4 GHz. The exploration performance is evaluated in the total exploration time, path length, computational time, and safety. We choose RH-NBV [1] and the frontier-based exploration algorithm mentioned in [10] as our benchmarks. We also evaluate the ESDF-based trajectory optimization in different scenarios. The video of all the experiments can be found on: https://youtu.be/4awOhb9TkCY

TABLE I. Parameters for our DEP planner.

Parameter	Value
Node Min. Distance, $d_{\text{th,min}}$	0.8 m
Node Max. Distance, $d_{\text{th,max}}$	1.5 m
Max. Sample Failure, $\mathcal{N}_{\max}$	50
Gain Parameters $[w_{\text{n}}, w_{\text{f}}, w_{\text{s}}]$	$[1, 2, 4]$
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	Auditorium	$20 \times 15 \times 4$
5	Tunnel	$20 \times 25 \times 9$
6	Dynamic Field	$24 \times 24 \times 3$

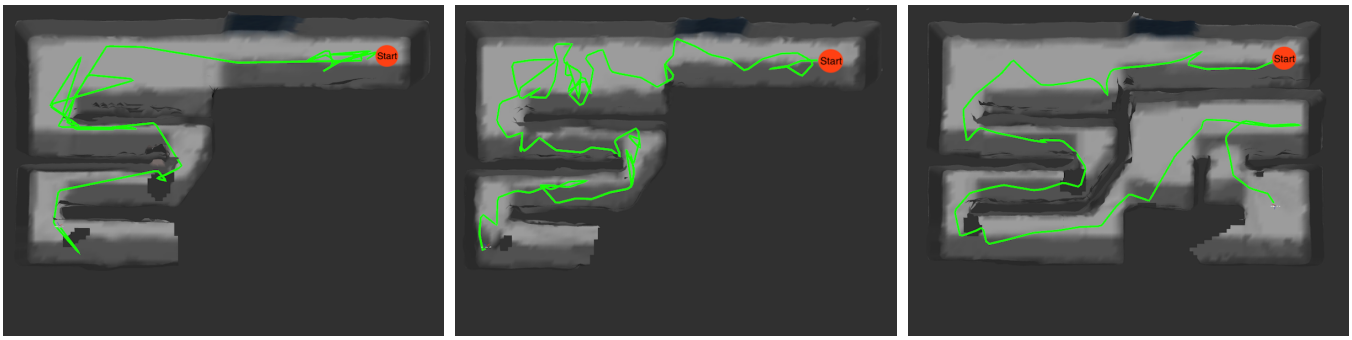
A. Simulation Environments

We select a total of six target environments in the 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 evaluate the ESDF-based optimization, we include the additional two environments: Auditorium and Tunnel. By testing with environments of different scales, we can comprehensively compare the exploration performance for different planners. To demonstrate the ability to generate safe exploration paths, we also include the Dynamic Field environment, which contains several walking people whose trajectories are unknown to the robot.

B. Exploration Performance

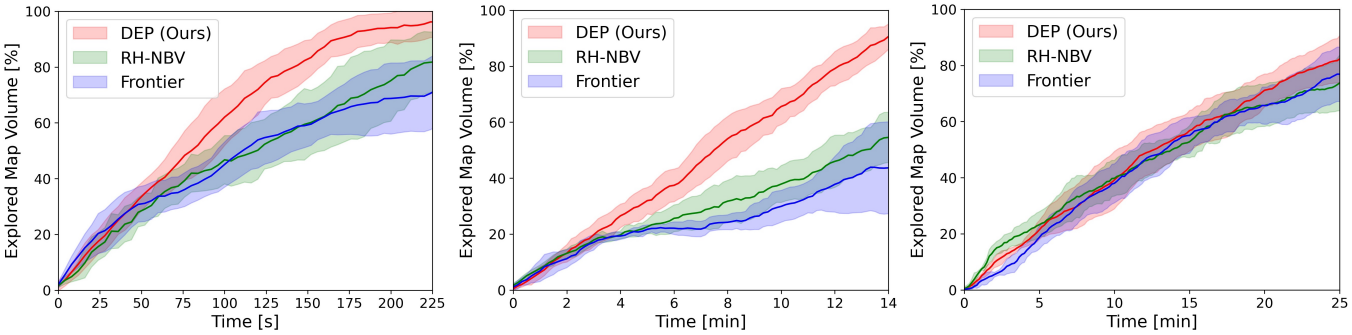
We evaluate the exploration performance for three environments of different scales: Cafe (Small), Maze (Medium), and Office (Large). We record the total exploration time, path length, and computational time for each environment and take ten experiments to obtain the means and standard deviations. The experiment parameters and our proposed planner parameters are shown in TABLE III and I, respectively.

In the Cafe (Small) environment, the exploration rate of RH-NBV and the frontier-based planner drops after 50 seconds while our planner still maintains a high level (Fig. 9a). This exploration rate drop can be explained by failing to generate the most informative path. RH-NBV is prone to generate "crooked" trajectories, which increase the motion time. Besides, the planner also fails to sample in some high-utility regions with a limited number of sampled nodes. The frontier-based planner does not determine the best viewpoint regarding the exploration utility, leading to myopic exploration for closest frontiers. Since our planner incrementally builds the roadmap based on the previously sampled nodes, each area's utility for exploration can be easily obtained to determine the optimal set of viewpoints. The trajectory optimization also improves the path quality and eliminates the nodes' sampling randomness. As expected, TABLE V shows that our



(a) Frontier-based Exploration (b) RH-NBV (c) 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 and less back-and-forth behavior.



(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) environments. Our planner has the highest exploration rate in the Cafe and Maze environments. Three planners have similar exploration rates in the Office environment.

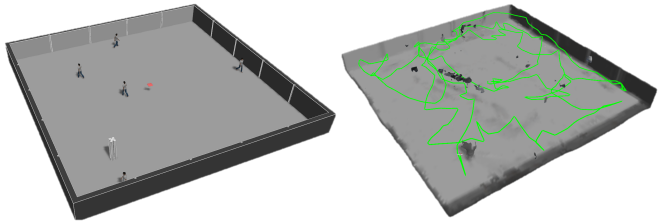


Fig. 7: Visualization of the Dynamic Field environment with its fully explored map and exploration trajectory.

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	4m
Map Resolution	0.2	Camera FOV	[103.2, 77.4]°

planner spends the least exploration time among the three. The environment and fully explored map are shown in Fig. 4.

This performance gap is even more evident for the Maze (Medium) environment, which has a more complex structure. Fig. 9b shows that the exploration rate for both RH-NBV and the frontier-based planner drops dramatically after 3 minutes, while our planner almost keeps the same exploration rate to the end. Due to the environmental complexity, the viewpoint and trajectory utility become more critical in guiding the exploration. The lack of utility evaluation in the frontier-based planner leads to the worst performance, and the planner tends to have back-and-forth behaviors in the explored region. Fig. 5 visualizes the explored maps of three planners in the Maze (Medium) environment after 10 minutes, and our planner has the largest explored area. The total exploration time of our planner from TABLE V is nearly the half of the other two planners'.

In the Office (Large) environment, Fig. 9c indicates that all the planners have a similar exploration rate. Unlike the previous two environments, the Office (Large) environment has larger open and free areas with multiple optimal viewpoints and trajectories. In this way, all the planners can obtain similar-quality trajectories and similar exploration rates. However, TABLE V shows that the total exploration time of our planner is less than the others because that RH-NBV and frontier-based exploration spend a longer time to discover some small rooms at the very end of the exploration process. Since the maximum sample number is limited in RH-NBV, when the target area is large, it will take many iterations to find small unexplored rooms.

Considering the path length, our planner has the least value

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 $\frac{T_{\text{opt}}}{T_{\text{non}}}$	Length Ratio $\frac{L_{\text{opt}}}{L_{\text{non}}}$	Distance Ratio $\frac{D_{\text{opt}}}{D_{\text{non}}}$
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%**

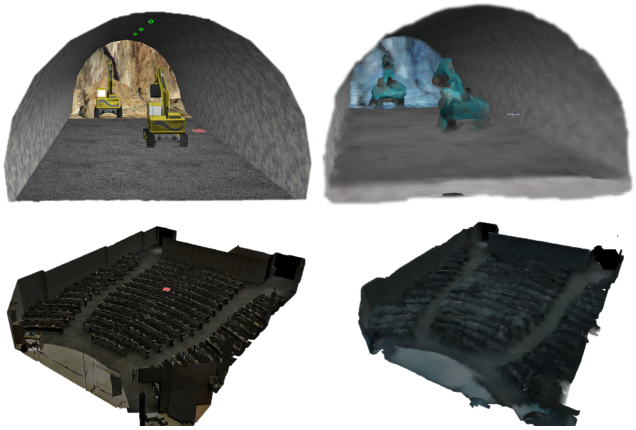


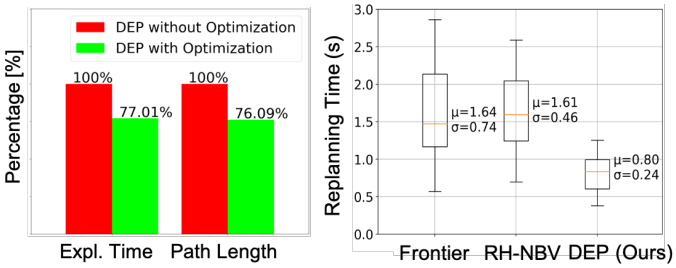
Fig. 8: Visualization of tunnel and auditorium environments in Gazebo (left) with their fully explored maps (right)

in both the Cafe (Small) and Maze (Medium) environments, indicating fewer back-and-forth trajectories. In contrast, the frontier-based exploration has the shortest path length in the Office (Large) environment. The trajectories generated by the frontier-based planner are more straightforward than ours and RH-NBV. Even though our optimization helps reduce the trajectory length in each planning iteration, the sampled nodes' randomness still can result in a slightly longer trajectory in some environments. 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 computations.

C. Evaluation of ESDF-based Optimization

To thoroughly analyze the ESDF-based trajectory optimization's importance, we add two more environments: auditorium and tunnel. The fully explored maps for the two environments are shown in Fig. 8. Our optimization aims to reduce the trajectory length while keeping the robot safe from obstacles and maintaining a high exploration gain.

Fig. 9a shows the overall performance comparison of our planner with and without trajectory optimization. We compare the exploration time and path length in percentage by the average of ten experiments for each environment. One can see that 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.



(a) (b)

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

D. Evaluation of Replanning Time

To verify the safety in exploring dynamic environments, we run experiments in the Dynamic Field environment, which contains five walking people. The trajectories of people are predefined but unknown to the robot. Through the experiments, our planner successfully explores the whole field space without any collision, and the replanning speed is only 35.4% of the regular planning iteration.

The comparison of the replanning time with the benchmarks is shown in Fig. 9b. The data is obtained by running each planner for the Dynamic Field five times when the robot encounters dynamic obstacles. The result shows that our planner has the least replanning time, and the other two planners take approximately twice the time as ours. The reduction of replanning time can be explained by the reuse of previously maintained roadmap.

VII. Conclusion and Future Work

In this paper, 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.

In some situations, however, due to the robot sensor's limitation, some dynamic obstacles cannot be detected. In this way, our assumption that the robot only needs to avoid its detected obstacle may not guarantee complete safety. As future work, a central detection system may be applied to track the dynamic obstacles in an environment.

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	7.12	1.40	76.80	14.91	0.50	0.10
	Frontier	6.44	0.52	56.11	5.10	0.37	0.15
Maze (Medium)	DEP (Ours)	**17.75**	0.71	**146.44**	21.19	**1.05**	0.11
	RH-NBV	31.44	3.80	271.60	31.93	8.10	1.01
	Frontier	34.74	3.35	330.10	31.13	2.01	1.15
Office (Large)	DEP (Ours)	**31.84**	2.63	318.58	78.94	**2.45**	0.31
	RH-NBV	48.21	9.89	421.80	80.92	12.13	2.83
	Frontier	38.14	13.80	**253.63**	43.26	12.29	9.49

REFERENCES

[1] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon "next-best-view" planner for 3d exploration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1462–1468, 2016.

[2] B. Charrow, S. Liu, V. Kumar, and N. Michael. Information-theoretic mapping using cauchy-schwarz quadratic mutual information. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4791–4798, 2015.

[3] Benjamin Charrow, Gregory Kahn, Sachin Patil, Sikang Liu, Kenneth Y. Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-theoretic planning with trajectory optimization for dense 3d mapping. In *Robotics: Science and Systems*, 2015.

[4] T. Cieslewski, E. Kaufmann, and D. 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, 2017.

[5] C. Connolly. The determination of next best views. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 432–435, 1985.

[6] M. Faria, A.S. Ferreira, H. Pérez-Leon, I. Maza, and A. Viguria. Autonomous 3d exploration of large structures using an uav equipped with a 2d lidar. *Sensors (Switzerland)*, 19(22), 2019.

[7] Héctor H. González-Baños and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.

[8] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at http://octomap.github.com.

[9] Steven G. Johnson. The nlopt nonlinear-optimization package. Software available at http://github.com/stevengj/nlopt.

[10] Miguel Juliá, Arturo Gil, and Oscar Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444, 2012.

[11] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[12] Steven M Lavalle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[13] L. Ly and Y. R. Tsai. Autonomous exploration, reconstruction, and surveillance of 3d environments aided by deep learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5467–5473, 2019.

[14] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373, 2017.

[15] N. Palomeras, N. Hurtós, E. Vidal, and M. Carreras. Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner. *IEEE Robotics and Automation Letters*, 4(2):1619–1625, 2019.

[16] C. Papachristos, S. Khattak, and K. Alexis. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575, 2017.

[17] 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 Robotics and Automation Letters*, 5(2):1500–1507, 2020.

[18] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt. Efficient autonomous exploration planning of large-scale 3-d environments. *IEEE Robotics and Automation Letters*, 4(2):1699–1706, 2019.

[19] R. Shrestha, F. Tian, W. Feng, P. Tan, and R. Vaughan. Learned map prediction for enhanced mobile robot exploration. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1197–1204, 2019.

[20] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[21] Hartmut Surmann, Andreas Nüchter, and Joachim Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45(3):181 – 198, 2003.

[22] D. G. Vutetakis and J. Xiao. An autonomous loop-closure approach for simultaneous exploration and coverage of unknown infrastructure using mavs. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2988–2994, 2019.

[23] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. 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, 2018.

[24] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation ICRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997.