TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments

Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang

***Abstract*— We present a method for autonomous exploration in complex three-dimensional (3D) environments. Our method demonstrates exploration faster than the current state-of-the-art using a hierarchical framework – one level maintains data densely and computes a detailed path within a local planning horizon, while another level maintains data sparsely and computes a coarse path at the global scale. Such a framework shares the insight that detailed processing is most effective close to the robot, and gains computational speed by trading-off details far away from the robot. The method optimizes an overall exploration path with respect to the length of the path and produces a kinodynamically feasible local path. In experiments, our systems autonomously explore indoor and outdoor environments at a high degree of complexity, with ground and aerial robots. The method produces 80% more exploration efficiency, defined as the average explored volume per second through a run, and consumes less than 50% of computation compared to the state-of-the-art.**

I. Introduction

We consider the problem of exploring three-dimensional (3D) spaces unknown *a priori* with an autonomous robot. Such a problem remains challenging as the problem has to deal with two tasks simultaneously – 1) online updating a representation of the environment to keep track of explored areas, and 2) searching the representation for a continuous traversable path to guide the exploration. In cases where the environment is large-scale, structurally complex, and 3D, the problem becomes computationally complex, and ensuring complete exploration of the environment can become a challenge.

The benefit of our method is that it can explore 3D spaces faster than the current state-of-the-art. The strength of the approach is based on a hierarchical framework to separate the processing at two levels. The first level maintains a high-resolution representation of the environment surrounding the robot, namely, the local planning horizon (green box in Fig. 1). Within that, a kinodynamically feasible path is generated for the robot to follow. The second level maintains a low-resolution representation and computes a path connecting distant areas, namely subspaces (solid green cubes in Fig. 1), in the global environment. The insight of such a framework is that detailed processing is most effective in the vicinity of the robot, while limited processing provides sufficient utility far away from the robot. The framework performs a bulk of the processing inside the local planning horizon and trades-off details for fast processing at the global scale.

The method first plans a path through the global representation. Such a path identifies the areas in the robot's free space that requires detailed exploration, hence the need for a detailed path to guide the robot to explore locally (green box in Fig. 1). The method uses the path through the global representation to move over large distances to areas throughout the free space (solid green cubes in Fig. 1). Combination of the paths at both levels allows the robot to explore areas in a sequence jointly determined by local and global information.

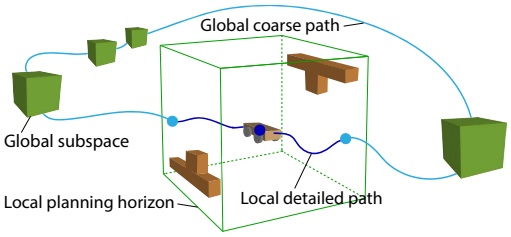


Fig. 1. Illustration of our exploration framework. Inside the local planning horizon (green box), data is densely maintained and a local detailed path is computed (dark-blue curve). At the global scale, data is sparsely maintained in distant subspaces (solid green cubes) and a global coarse path is computed (light-blue curve). The local path and global path are connected on the boundary of the local planning horizon to form the exploration path.

Our method is evaluated using both ground and aerial robots. We show results where the ground robot explores the interior of a complex multi-storage building in physical experiment and the aerial robot explores a large-scale outdoor environment in simulation. We compare the results to state-of-the-art methods and conclude that the proposed method produces 80% more exploration efficiently, defined as the average explored volume per second, and uses computation less than 50% of the state-of-the-art methods.

Further, we release a software environment for benchmarking exploration algorithms and facilitating the development of complete autonomous navigation systems. The environment contains representative simulation environment models, fundamental navigation modules, e.g. collision avoidance, terrain traversability analysis, way-point following, and visualization tools. Our algorithm code is made publicly available[1].

II. Related Work

The problem of autonomous exploration has been tackled from multiple angles. The approach described in this paper is based on key results in information theory, frontier-based exploration, topological exploration, and a few random sampling-based methods briefly discussed in this section.

Information Theory: A popular approach in solving the exploration problem is using information theory. The methods maximize the information gain over the next few actions [1]–[4]. The above-mentioned work are also extended to solving

TARE is named after an effort to develop Technologies for Autonomous Robot Exploration. All authors are with the Robotics Institute at Carnegie Mellon University, Pittsburgh PA. Emails: {ccao1, hongbiaz, choset, zhangji}@cmu.edu.

[1]Sim. environment and source-code: www.cmu-exploration.com

the multi-robot exploration problem [5]–[7]. In summary, majority of these existing methods rely on greedy strategies, where efficiency is limited due to the methods being myopic. In contrast, our method seeks the optimal exploration path as a whole other than maximizing the instant rewards.

Frontier-based Exploration: A common formulation of the problem uses frontiers, i.e. boundary between mapped and unmapped areas. When exploring, the vehicle keeps moving toward frontiers, which extends the boundary of the mapped areas until the entire environment is explored [8]–[14]. While most of these methods greedily select frontiers to explore, Faigl and Kulich's method determines a minimum set of viewpoints to cover the frontiers by solving a variant of the art gallery problem [15]. Similarly, our method also finds a minimum set of viewpoints but does so by recursively and randomly sampling the viewpoints.

Topological Exploration: Other approaches model the environment with topological representations [16]–[18]. Most of these approaches divide the environment into topologically distinct sections, where an exhaustive traversal through the sections entails complete exploration of the environment. While these methods focus on topological completeness, our method aims at producing a detailed map of the environment.

Random Sampling-based Methods: Recent work develops a few methods based on the Rapidly-exploring Random Tree (RRT) [19] or Rapidly-exploring Random Graph (RRG) [20]. The methods span RRTs or RRGs in the environment to find the traversable space, within which the most informative branch on the RRT or RRG is selected as the exploration path. Specifically, Bircher et al. propose the Next-Best-View Planner (NBVP) [21]. The method spans an RRT, the nodes of which are modeled as viewpoints. The next viewpoint on the branch that maximizes a reward function is chosen as the navigation goal. Witting et al. [22] extend NBVP by seeding the RRT with the vehicle's trajectory, which allows the vehicle to explore further in areas passed by previously. Dang et al. improve the scheme further by proposing the Graph-Based exploration Planner (GBP) [23]. The method constructs a global RRG along the vehicle trajectory. When the local area is explored, the method plans routes based on the global RRG to distant areas for further exploration. Note that in this method, the exploration mode and relocation mode are explicitly switched using heuristics. Recently, Dharmadhikari at al. propose the Motion primitives-Based exploration Planner (MBP) [24], a variant of GBP that constructs the RRT with motion primitives, which produces smoother local paths spanning in constrained directions. In essence, these methods adopt greedy strategies. Due to the randomness of RRT and RRG, these methods also tend to overlook areas that have not been completely explored.

The main contribution of our work is a hierarchical framework to enable highly efficient exploration. The framework does not involve heuristics, as GBP and MBP, for explicit mode switch. Experiment comparisons to NBVP [21], GBP [23] and MBP [24] show that our method explores much more completely and efficiently while consuming less computation.

III. Problem Definition

Define $\mathcal{Q} \subset \mathbb{R}^3$ as the work space to be explored. Let $\mathcal{Q}_{\text{trav}} \subset \mathcal{Q}$ be the traversable subspace. Define viewpoint $\mathbf{v} \in \mathbf{SE}(\mathbf{3})$ to describe the pose of the sensor onboard the robot, $\mathbf{v} = [\mathbf{p_v}, \mathbf{q_v}]$ where $\mathbf{p_v} \in \mathcal{Q}_{\text{trav}}$ and $\mathbf{q_v} \in \mathbf{SO}(\mathbf{3})$ respectively denote the position and orientation. Denote $\mathcal{L} \subset \mathbf{SE}(\mathbf{3})$ as the set of viewpoints along the vehicle past trajectory. We use the term "surface" to refer to the generalized boundary between free space and non-free space, the latter includes both occupied and unknown spaces. Let $\mathcal{S_v} \subset \mathcal{Q}$ be the surfaces perceived by the sensor at $\mathbf{v}$. Note that the same surface can be perceived from multiple viewpoints. The perceived surfaces so far are

$$\mathcal{S} = \bigcup_{\mathbf{v} \in \mathcal{L}} \mathcal{S_v}. \tag{1}$$

Here, $\mathcal{S}$ contains covered surfaces, denoted as $\mathcal{S}_{\text{cov}} \subset \mathcal{S}$, and yet uncovered surfaces, denoted as $\bar{\mathcal{S}} = \mathcal{S} \setminus \mathcal{S}_{\text{cov}}$. We would like to find the shortest path, which when followed by the vehicle covers $\bar{\mathcal{S}}$. The path must meet kinodynamic constraints. Let $\mathbf{v}_{\text{current}}$ be the viewpoint located at the vehicle's current sensor pose. Our problem can be defined as follows.

Problem 1: Given $\bar{\mathcal{S}}$ and $\mathbf{v}_{\text{current}}$, find the shortest path $\mathcal{T}^*$ formed by viewpoints $\mathbf{v}_1, \mathbf{v}_2, ...$, which when followed by the vehicle covers $\bar{\mathcal{S}}$, such that $\mathbf{v}_{\text{current}} \in \mathcal{T}^*$, and $\mathcal{T}^*$ is kinodynamically feasible.

Problem 1 is solved repetitively at each planning cycle. We use $\bar{\mathcal{S}}$ to compute the exploration path. When executing the path, we online update $\mathcal{S}$ with up-to-date sensor readings, processing both displaced and newly perceived surfaces. Then, we move surfaces become covered from $\bar{\mathcal{S}}$ to $\mathcal{S}_{\text{cov}}$, and use $\bar{\mathcal{S}}$ in the next planning cycle, hence the exploration continues.

IV. Methodology

A. Viewpoint Sampling

We define the criteria for a surface point to be covered by the sensor. Consider a surface patch centered at $\mathbf{p}_s \in \mathcal{Q}$ with normal $\mathbf{n}_s \in \mathbb{R}^3$ pointing toward the free-space side, the center point on the surface patch is covered by viewpoint $\mathbf{v}$, if

$$|\mathbf{p}_s - \mathbf{p_v}| \leq D, \tag{2}$$

$$\mathbf{n}_s \cdot (\mathbf{p_v} - \mathbf{p}_s)/|\mathbf{n}_s||\mathbf{p_v} - \mathbf{p}_s| \geq T, \tag{3}$$

where D and T are two constants constraining the relative distance and orientation of the surface patch w.r.t. the viewpoint. Such criteria encourage the surfaces to be perceived well. In practice, D is set to be shorter than the sensor range.

Define $\mathcal{H} \subset \mathcal{Q}$ as the local planning horizon as shown in Fig. 2. Let $\mathcal{H}_{\text{trav}} \subset \mathcal{H}$ be the traversable subspace identified by considering collision and connectivity, and let $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$ be the corresponding configuration space considering rotation and translation. Define $\bar{\mathcal{S}}_{\mathcal{H}} \subset \bar{\mathcal{S}}$ as the uncovered surfaces that can be perceived from viewpoints in $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$. The problem of viewpoint sampling is to select a minimum set of viewpoints in $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$ to cover $\bar{\mathcal{S}}_{\mathcal{H}}$. Let us use $\bar{\mathcal{S}}_{\mathbf{v}} \subset \bar{\mathcal{S}}_{\mathcal{H}}$ to denote the uncovered surfaces to be perceived from $\mathbf{v} \in \mathcal{C}^{\mathcal{H}}_{\text{trav}}$. The reward of $\mathbf{v}$ is defined as the area of $\bar{\mathcal{S}}_{\mathbf{v}}$, denoted as $A_{\mathbf{v}}$.

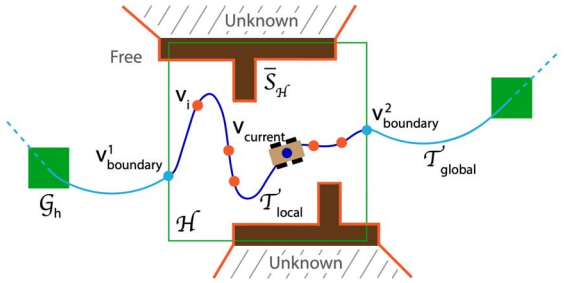


Fig. 2. Illustration of mathematical definitions. The green box represents the local planning horizon $\mathcal{H}$. The solid green squares represent the exploring subspaces $\mathcal{G}_h$, $h \in \mathbb{Z}^+$. The dark-blue curve is the local path $\mathcal{T}_{\text{local}}$. The light-blue curve is the global path $\mathcal{T}_{\text{global}}$. The dark-blue dot on $\mathcal{T}_{\text{local}}$ is the current viewpoint $\mathbf{v}_{\text{current}}$. The light-blue dots are viewpoints $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$ on the boundary of $\mathcal{H}$ where $\mathcal{T}_{\text{local}}$ and $\mathcal{T}_{\text{global}}$ are connected. The orange dotes are sampled viewpoints $\mathbf{v}_i$, $i \in \mathbb{Z}^+$. The orange lines are uncovered surfaces $\bar{\mathcal{S}}_{\mathcal{H}}$ to be perceived by the viewpoints. The method uses an iterative random sampling process in determining $\mathbf{v}_i$ to cover $\bar{\mathcal{S}}_{\mathcal{H}}$.

Note that the problem exhibits submodularity [25], i.e. with more viewpoints selected, the reward of selecting an additional viewpoint decreases. This is because nearby viewpoints have overlapping field-of-views, and the same surface can be perceived from multiple viewpoints. Consequently, the reward of a viewpoint is dependent on the viewpoints selected earlier. Let $\mathbf{v}_i$, $i \in \mathbb{Z}^+$, be the i-th viewpoint selected. The uncovered surfaces to be perceived from $\mathbf{v}_i$, $\bar{\mathcal{S}}_{\mathbf{v}_i}$, needs to be adjusted to $\bar{\mathcal{S}}_{\mathbf{v}_i} - \bigcup_{j=1}^{i-1}(\bar{\mathcal{S}}_{\mathbf{v}_i} \cap \bar{\mathcal{S}}_{\mathbf{v}_j})$. Then, $A_{\mathbf{v}_i}$ is adjusted accordingly.

Algorithm 1 presents the process of viewpoint sampling. The algorithm first generates a set of viewpoint candidates $\mathcal{V}$ uniformly from a lattice pattern in $\mathcal{H}_{\text{trav}}$ (line 1). Second, it computes the rewards $A_{\mathbf{v}}$ for all viewpoint candidates $\mathbf{v} \in \mathcal{V}$ by estimating their coverages $\bar{\mathcal{S}}_{\mathbf{v}}$ with the updated environment representation (line 3). Then, a process is iterated for K times to determine the viewpoints. At each iteration, the algorithm randomly samples a subset of viewpoints from $\mathcal{V}$ that covers $\bar{\mathcal{S}}_{\mathcal{H}}$ (line 6-11). Three viewpoints are pre-selected (line 6), one as the current viewpoint $\mathbf{v}_{\text{current}}$, and the other two as the viewpoints on the boundary of $\mathcal{H}$, $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$, connecting the local path and global path. The process of determining $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$ is given in Section IV-C. A priority queue $\mathcal{Q}'$ is used to manage the viewpoint candidates. The priority of a viewpoint $\mathbf{v}$ is set to its reward $A_{\mathbf{v}}$. Viewpoints are selected from the priority queue with probabilities proportional to their rewards (line 8). Due to the submodularity, the rewards of the remaining viewpoints in the priority queue are reduced accordingly after a viewpoint is selected, accounting for the overlapping field-of-views (line 10). The viewpoint sampling process finishes when the priority queue is empty or the marginal reward of adding a new viewpoint is negligible. The algorithm calls Algorithm 2 to generate a path through the sampled viewpoints (line 12), discussed in Section IV-B. Via iterations, paths with lower costs are found and the path with the lowest cost is returned as the local path, denoted as $\mathcal{T}_{\text{local}}$.

Algorithm 1: Compute Local Path

input : traversable C-space $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$, uncovered surfaces $\bar{\mathcal{S}}_{\mathcal{H}}$, current viewpoint $\mathbf{v}_{\text{current}}$, boundary viewpoints $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$
output: local path $\mathcal{T}_{\text{local}}$

1 Generate a set of viewpoint candidates $\mathcal{V}$ in $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$;
2 Initialize priority queue Q;
3 For every $\mathbf{v} \in \mathcal{V}$, estimate coverage $\bar{\mathcal{S}}_{\mathbf{v}}$, then push $\mathbf{v}$ into Q with the priority set to its reward $A_{\mathbf{v}}$;
4 $\mathcal{T}_{\text{local}} \leftarrow \emptyset$, $c_{best} \leftarrow +\infty$;
5 **for** $i := 1$ to K **do**
6 $\quad Q' \leftarrow Q$, $\mathcal{V}' \leftarrow \{\mathbf{v}_{\text{current}}, \mathbf{v}^1_{\text{boundary}}, \mathbf{v}^2_{\text{boundary}}\}$;
7 $\quad$ **while** $Q' \neq \emptyset$ and Q' contains at least one non-zero priority **do**
8 $\quad\quad$ Probabilistically pick viewpoint $\mathbf{v}'$ in Q', then remove $\mathbf{v}'$ from Q';
9 $\quad\quad$ $\mathcal{V}' \leftarrow \mathcal{V}' \cup \mathbf{v}'$;
10 $\quad\quad$ Update priorities for all viewpoints in Q';
11 $\quad$ **end**
12 $\quad$ Compute smooth path $\mathcal{T}'_{\text{smooth}}$ and cost c'_{smooth} using Algorithm 2;
13 $\quad$ **if** $c'_{\text{smooth}} < c_{\text{best}}$ **then**
14 $\quad\quad$ $\mathcal{T}_{\text{local}} \leftarrow \mathcal{T}'_{\text{smooth}}$, $c_{\text{best}} \leftarrow c'_{\text{smooth}}$;
15 $\quad$ **end**
16 **end**
17 **return** $\mathcal{T}_{\text{local}}$;

B. Path Generation and Smoothing

Given a set of sampled viewpoints $\mathcal{V}'$, we want to generate a path through each of the viewpoints. Ideally, we would like the path to be kindodynamically feasible, which can be followed by the vehicle at a high speed. Here, we focus on the curvature constraint such that the maximum curvature of the path is determined by the vehicle's minimum turning radius when moving at the desired speed. Due to the distribution of the viewpoints in $\mathcal{V}'$ and structures in the environment, a continuous path that meets the curvature constraint can be impossible. The method computes the path in smooth segments as shown in Fig. 3. The vehicle stops at the end of each segment before moving on to the next segment in a different direction. This requires that the vehicle can turn in one place. Considering the focus of this paper is not on path planning using sophisticated motion models, we adopt the generic differential motion model, and for aerial vehicles, with independent altitude control. Let us denote the path as $\mathcal{T}'_{\text{smooth}} = [\mathbf{v}^1_1, \mathbf{v}^1_2, ...][\mathbf{v}^2_1, \mathbf{v}^2_2, ...]...$, where $[\cdot]$ represents a segment and $\mathbf{v}^j_k$ is the k-th viewpoint on the j-th segment, $j, k \in \mathbb{Z}^+$. Note that the last viewpoint on a segment and the first viewpoint on the following segment share the same viewpoint. Let $\tilde{n} \in \mathbb{Z}^+$ be the number of segments, $\tilde{n} \geq j$. Define l_j as the length of the j-th segment. We discourage stopping too frequently by applying a penalty p at each stop. The cost of $\mathcal{T}'_{\text{smooth}}$ is defined as

$$c'_{\text{smooth}} = \sum_{j=1}^{\tilde{n}} l_j + p(\tilde{n} - 1). \tag{4}$$

The problem of computing the path can be stated as follows.

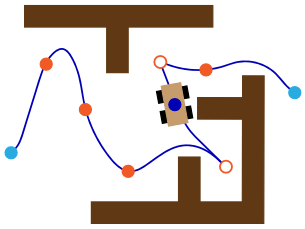


Fig. 3. Illustration of smoothed path $\mathcal{T}_{\text{smooth}}$. The dark-blue curves represent path segments which satisfy the curvature constraint. The dark-blue dot is $\mathbf{v}_{\text{current}}$. The light-blue dots are $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$ connecting to $\mathcal{T}_{\text{global}}$. The orange dots are sampled viewpoints and the hollow-centered orange dots are break-points between the path segments. The vehicle makes a stop at each break-point before moving on to the next path segment.

Problem 2: Given viewpoints $\mathcal{V}'$, find a path $\mathcal{T}^*_{\text{smooth}} = [\mathbf{v}^1_1, \mathbf{v}^1_2, ...][\mathbf{v}^2_1, \mathbf{v}^2_2, ...]...$ with the lowest c'_{smooth} such that $\mathcal{T}^*_{\text{smooth}}$ visits each of the viewpoints in $\mathcal{V}'$ once and each segment on $\mathcal{T}^*_{\text{smooth}}$ satisfies the curvature constraint.

Problem 2 is NP-hard given that it is an extended version of the Traveling Salesman Problem (TSP) [26]. Instead of attempting to find an exact solution, we solve the problem using approximation algorithms in two steps. First, we solve for an order of the viewpoints with a standard TSP. Second, we separate the viewpoints into segments following the order. This is to determine if a viewpoint, except the first and the last ones, is an inner-point within a segment or a break-point between two segments. Let $n' \in \mathbb{Z}^+$ be the number of viewpoints in $\mathcal{V}'$. Define $\mathbf{x} = [x_1, x_2, ..., x_{n'-2}]$ as a sequence of boolean variables describing the status of the $n'-2$ viewpoints (excluding the first and last) and a function $f(\mathbf{x}) = c'_{\text{smooth}}$. The problem of determining the viewpoint status can be formulated as,

Problem 3: Given a sequence of bolean variables $\mathbf{x}$ and a function $f(\mathbf{x}) = c'_{\text{smooth}}$, find $\mathbf{x}^*$ such that

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, c'_{\text{smooth}}. \tag{5}$$

Problem 3 is a nonlinear integer programming problem and known to be NP-hard [27]. Especially, finding smooth paths involves time-consuming trajectory optimization. Instead of using an existing heuristic method such as [28] with a higher computational complexity, our method applies a greedy strategy to check each viewpoint only once, maximally reducing the runtime. Algorithm 2 gives the procedure of computing $\mathcal{T}'_{\text{smooth}}$ from a set of viewpoints $\mathcal{V}'$. The algorithm finds the shortest collision-free path between every viewpoint pair in $\mathcal{V}'$ using the A* algorithm [29] and construct a distance matrix $\mathbf{D}'$ containing the length of the paths (line 1). Next, the algorithm solves a TSP for a traversal order of the viewpoints (line 2). Upon determining the segments on $\mathcal{T}'_{\text{smooth}}$, the algorithm initializes all $n'-2$ viewpoints as break-points (line 3) and smooths the segments between consecutive viewpoints (line 4). Later, the algorithm attempts to reduce the cost c'_{smooth} by sequentially setting each viewpoint $\mathbf{v}_i$ as an inner-point and re-smoothing the segment through $\mathbf{v}_i$ (lines 5-11). Each path segment is smoothed with a trajectory optimization method similar to [30], where the segment is modeled as a set of cubic splines connected at the viewpoints. Boundary conditions are applied at the viewpoints. Control points for the splines are placed on the initial paths given by the A* search and adjusted by a nonlinear optimization solver [31] to account for collision clearance and path smoothness. To accelerate processing, the nonlinear optimization is marginalized where only the splines between the two adjacent viewpoints of $\mathbf{v}_i$ are optimized. The algorithm returns $\mathcal{T}'_{\text{smooth}}$ with c'_{smooth}.

C. Global Planning

We divide the space outside $\mathcal{H}$ into even cuboid subspaces. Each subspace stores the covered and uncovered surfaces developed during the exploration. Note that the data is kept in the subspaces only for storage, while the data in $\mathcal{H}$ is actively updated as the exploration proceeds. Each subspace holds a status from "unexplored", "exploring", and "explored". If a subspace does not contain any covered or uncovered surfaces, the status is "unexplored". If a subspace contains only covered surfaces, the status is "explored". If a subspace contains any uncovered surfaces, the status is "exploring". We only consider the exploring subspaces in global planning. Denote $\mathcal{G}_h \subset \mathcal{Q}$, $h \in \mathbb{Z}^+$, as an exploring subspace and $\hat{\mathcal{G}}$ as the set of exploring subspaces. The global planning problem is to find a global path $\mathcal{T}_{\text{global}}$ that goes through the current viewpoint $\mathbf{v}_{\text{current}}$ and the centroid of each subpace in $\hat{\mathcal{G}}$. During the course of the exploration, we construct a sparse random roadmap in the traversable space expanded from the past trajectory. Similar to local planning, we use A* search on the roadmap for shortest paths between the subspaces followed by solving a TSP.

Algorithm 3 gives the overall procedure for computing the exploration path. The algorithm constructs a distance matrix containing the length of the paths found on the roadmap (line 1) and solves a TSP (line 2). The two points on $\mathcal{T}_{\text{global}}$ intersecting with the boundary of $\mathcal{H}$ are extracted as $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$ (line 3). Then, Algorithm 1 is used to compute $\mathcal{T}_{\text{local}}$ (line 4). Finally, $\mathcal{T}_{\text{local}}$ and $\mathcal{T}_{\text{global}}$ are concatenated by substituting the part of $\mathcal{T}_{\text{global}}$ inside $\mathcal{H}$ with $\mathcal{T}_{\text{local}}$ (line 5). Fig. 4 shows an example of the exploration where the dark-blue and light-blue paths represent $\mathcal{T}_{\text{local}}$ and $\mathcal{T}_{\text{global}}$.

When the exploration completes in $\mathcal{H}$ ($\bar{\mathcal{S}}_{\mathcal{H}} = \emptyset$), $\mathcal{T}_{\text{local}}$ reduces to the shortest paths connect from $\mathbf{v}_{\text{current}}$ to $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$, which are further connected to the adjacent exploring subspaces on $\mathcal{T}_{\text{global}}$. The vehicle follows the path to transit to an exploring subspace to resume exploration. In other words, the algorithm implicitly transitions between exploration and relocation to another area to explore further. If $\bar{\mathcal{S}}_{\mathcal{H}} = \emptyset$ and $\hat{\mathcal{G}} = \emptyset$, the exploration terminates.

D. Theoretical Analysis

1) Computational Complexity: Let $n \in \mathbb{Z}^+$ be the number of viewpoint candidates generated in Algorithm 1. For a fixed number of iterations, the sampled viewpoints is no more than n. After selecting each viewpoint, adjusting rewards of the remaining viewpoints in the priority queue takes $O(n)$ time. In Algorithm 2, we model the time of finding the shortest

Algorithm 2: Compute Local Path from Viewpoints

input : traversable C-space $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$, viewpoints $\mathcal{V}'$
output: smooth path $\mathcal{T}'_{\text{smooth}}$, cost c'_{smooth}

Compute shortest paths between viewpoint pairs in $\mathcal{V}'$, then create distance matrix $\mathbf{D}'$;
Compute path $\mathcal{T}'$ by solving TSP using $\mathbf{D}'$;
Initialize $\mathcal{T}'_{\text{smooth}}$ using $\mathcal{T}'$, then set viewpoints $\mathbf{v}_2$, $\mathbf{v}_3$, ..., $\mathbf{v}_{n-1}$ on $\mathcal{T}_{\text{smooth}}$ as break-points;
Smooth path segments between consecutive viewpoints on $\mathcal{T}'_{\text{smooth}}$ and compute cost c'_{smooth};
for $i := 2$ to $n' - 1$ **do**
    Temporarily set viewpoint $\mathbf{v}_i$ as an inner-point by connecting the two segments on both sides of $\mathbf{v}_i$;
    Smooth the path segment through $\mathbf{v}_i$ and compute cost c'_{temp};
    if $c'_{\text{temp}} < c'_{\text{smooth}}$ and $\mathcal{T}'_{\text{smooth}}$ meets curvature constraint **then**
        Finalize $\mathbf{v}_i$ on $\mathcal{T}'_{\text{smooth}}$ as an inner-point;
        $c'_{\text{smooth}} \leftarrow c'_{\text{temp}}$;
    end
end
return $\mathcal{T}'_{\text{smooth}}$, c'_{smooth};

path between two viewpoints and the time of smoothing a path segment as bounded by two constants (the trajectory optimization is marginalized where only a bounded number of control points are optimized). The time complexity of constructing the distance matrix is $O(n^2)$. The TSP is solved using the Lin–Kernighan heuristic which consumes $O(n^{2.2})$ time [26]. The path smoothing processes each viewpoint once and takes $O(n)$ time. In Algorithm 3, computing the distance matrix takes $O(m^2)$ time and solving the TSP runs in $O(m^{2.2})$ time, where $m \in \mathbb{Z}^+$ is the number of exploring subspaces. Concatenating $\mathcal{T}_{\text{local}}$ and $\mathcal{T}_{\text{global}}$ takes constant time. The time complexity of our algorithm is stated in Theorem 1.

Theorem 1: Algorithm 3 runs in $O(n^{2.2} + m^{2.2})$ time.

2) Probabilistic Completeness: Given a set of viewpoints $\mathcal{V}$, let $A_{\mathcal{V}}$ be the corresponding area of covered surfaces. It is our observation that $A_{\mathcal{V}}$ monotonically increases as more viewpoints are added to $\mathcal{V}$. Define a set function $a(\mathcal{V}) = A_{\mathcal{V}}$. For any two sets of viewpoints $\mathcal{V}$ and $\mathcal{V}'$, we have

$$\mathcal{V}' \subseteq \mathcal{V} \Rightarrow a(\mathcal{V}') \leq a(\mathcal{V}). \tag{6}$$

The above relationship helps us draw Lemma 1 as follows.

Lemma 1: The set function $a(\mathcal{V}) = A_{\mathcal{V}}$ is monotone.

Further, we observe that $A_{\mathcal{V}}$ exhibits submodularity, i.e. as more viewpoints are selected, the marginal reward of adding a new viewpoint monotonically decreases. As discussed, this is due to that the same surface is perceived by multiple viewpoints with overlapping field-of-views. For any two sets of viewpoints $\mathcal{V}$ and $\mathcal{V}'$, and a single viewpoint $\mathbf{v}$,

$$\mathcal{V}' \subseteq \mathcal{V} \Rightarrow a(\mathcal{V}' \cup \mathbf{v}) - a(\mathcal{V}') \geq a(\mathcal{V} \cup \mathbf{v}) - a(\mathcal{V}). \tag{7}$$

Algorithm 3: Compute Exploration Path

input : local planning horizon $\mathcal{H}$, traversable C-space $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$, uncovered surfaces $\bar{\mathcal{S}}_{\mathcal{H}}$, exploring subspaces $\hat{\mathcal{G}}$, current viewpoint $\mathbf{v}_{\text{current}}$
output: exploration path $\mathcal{T}$

Compute shortest paths between centroids in $\hat{\mathcal{G}}$ and $\mathbf{v}_{\text{current}}$, then create distance matrix $\mathbf{D}$;
Compute global path $\mathcal{T}_{\text{global}}$ by solving TSP using $\mathbf{D}$;
Extract $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$ as the intersections between $\mathcal{T}_{\text{global}}$ and the boundary of $\mathcal{H}$;
Compute local path $\mathcal{T}_{\text{local}}$ using Algorithm 1;
Concatenate $\mathcal{T}_{\text{local}}$ and $\mathcal{T}_{\text{global}}$ to generate $\mathcal{T}$;
return $\mathcal{T}$;

The above relationship helps us draw Lemma 2 as follows.

Lemma 2: The set function $a(\mathcal{V}) = A_{\mathcal{V}}$ is submodular.

Let $\mathcal{S}_{\mathcal{H}} \subset \mathcal{Q}$ be the surfaces perceivable from viewpoints in $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$. As our method uses random sampling in selecting the viewpoints, Theorem 2 states the probabilistic completeness.

Theorem 2: Algorithm 3 is probabilistically complete in computing path $\mathcal{T}$ to cover $\mathcal{S}_{\mathcal{H}}$.

Sketch Proof: Let $\mathcal{S}_{\mathcal{H},\text{cov}} \subset \mathcal{S}_{\mathcal{H}}$ be the covered surfaces and recall $\bar{\mathcal{S}}_{\mathcal{H}} \subset \mathcal{S}_{\mathcal{H}}$ as the uncovered surfaces. Since function $a(\mathcal{V}) = A_{\mathcal{V}}$ is monotone, as more viewpoints are selected, $\mathcal{S}_{\mathcal{H},\text{cov}} \to \mathcal{S}_{\mathcal{H}}$ and correspondingly $\bar{\mathcal{S}}_{\mathcal{H}} \to \emptyset$. The probability that $\bar{\mathcal{S}}_{\mathcal{H}}$ remains nonempty $p(\bar{\mathcal{S}}_{\mathcal{H}} \neq \emptyset) \to 0$. ■

The exploration process covers all surfaces in $\mathcal{S}$ at the end since the termination criterion is $\bar{\mathcal{S}}_{\mathcal{H}} = \emptyset$ and $\hat{\mathcal{G}} = \emptyset$.

3) Approximation Ratio: Let us analyze the approximation ratio introduced by the hierarchy. To simplify the problem, we evaluate a path with its length and ignore its kinodynamic feasibility. Define $\mathcal{T}^*$ as the shortest possible path to complete the coverage. $\mathcal{T}^*$ is computed without using the hierarchy and considered the theoretically optimal solution. Denote l^* as the length of $\mathcal{T}^*$. Let $D_{\mathcal{H}}$ be the longest distance needed to travel between any two viewpoints in $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$, i.e. for all the shortest paths between any two viewpoints in $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$, $D_{\mathcal{H}}$ is the length of the longest path. As defined in (2), D is the distance limit for covering surfaces. Denote D_{width}, D_{length}, and D_{height} as the dimensions of a subspace in $\hat{\mathcal{G}}$. We consider a group

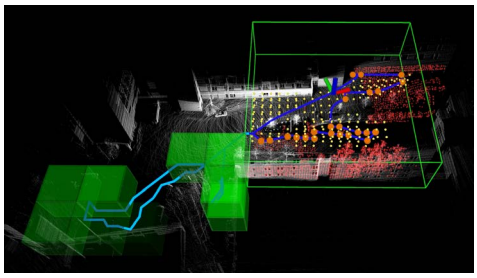


Fig. 4. An example exploration process with real data. The figure uses the same color code as Fig. 2. The white points show lidar scan data, with which the method extracts uncovered surfaces (red points). The yellow dots are the viewpoint candidates, from which viewpoints are sampled (orange dots).

of $([D/D_{\text{width}}]+2)\times([D/D_{\text{length}}]+2)\times([D/D_{\text{height}}]+2)$ connected subspaces where the sensor can stay in one subspace and "see through" to cover surfaces in another subspace. Let $D_{\mathcal{G}}$ be the longest distance needed to travel between any two viewpoints in such a group of subspaces. The approximation ratio introduced by the hierarchy is stated in Theorem 3.

Theorem 3: The approximation ratio of the path length by the hierarchy has $\sigma_{\text{hier}} \leq (l^* + 4D_{\mathcal{H}} + 2mD_{\mathcal{G}})/l^*$.

Proof: As shown in Fig. 5, let $\mathcal{T}^*_{\text{local}}$ be the part of $\mathcal{T}^*$ inside $\mathcal{H}$ and let $\mathcal{T}^*_{\text{global}}$ be the part outside $\mathcal{H}$. The length of $\mathcal{T}^*_{\text{local}}$ and $\mathcal{T}^*_{\text{global}}$ are denoted as l^*_{local} and l^*_{global}, respectively. Define $\mathcal{T}^\dagger_{\text{global}}$ as the shortest global path connecting to $\mathcal{T}^*_{\text{local}}$ at B, C and going through the centriod of each subspace in $\hat{\mathcal{G}}$. The length of $\mathcal{T}^\dagger_{\text{global}}$ is denoted as $l^\dagger_{\text{global}}$. We have

$$l^\dagger_{\text{global}} \leq l^*_{\text{global}} + 2mD_{\mathcal{G}}. \tag{8}$$

Eq. (8) can be proved by contradiction. On $\mathcal{T}^*_{\text{global}}$, we add two pieces of short paths to connect it to the centroid of each subspace in $\hat{\mathcal{G}}$. One short path is for traveling to the centroid and the other for coming back to $\mathcal{T}^*_{\text{global}}$. Since $\mathcal{T}^*$ completes the coverage, $\mathcal{T}^*_{\text{global}}$ has to approach each subspace in $\hat{\mathcal{G}}$ for the underlying surfaces to be covered. These short paths can be no longer than $D_{\mathcal{G}}$. This way, we form a new path with $\mathcal{T}^*_{\text{global}}$ and a number of $2m$ short paths. The new path connects to B, C and visits the centroid of each subspace in $\hat{\mathcal{G}}$. If (8) does not hold, the new path becomes shorter than $\mathcal{T}^\dagger_{\text{global}}$ and a contradiction occurs. Let l_{global} be the length of the global path $\mathcal{T}_{\text{global}}$ found by our method. Note that Theorem 3 states the approximation ratio introduced by the hierarchy only. Here, we do not consider the effect of the approximation algorithm used to solve the TSP. In other words, we assume an exact solution is obtained from solving the TSP and $\mathcal{T}_{\text{global}}$ is the shortest possible path connecting to E, F and going through the centroid of each subspace in $\hat{\mathcal{G}}$. We can show that

$$l_{\text{global}} \leq l^\dagger_{\text{global}} + 2D_{\mathcal{H}}. \tag{9}$$

Again, we use proof by contradiction. We find two short paths connecting between B, E and C, F, respectively. Each short path can be no longer than $D_{\mathcal{H}}$. This way, we form a new path with $\mathcal{T}^\dagger_{\text{global}}$ and the two short paths. The new path connects to E, F and visits the centroid of each subspace in $\hat{\mathcal{G}}$. If (9) does not hold, the new path becomes shorter than $\mathcal{T}_{\text{global}}$. Therefore, a contradiction occurs. Further, let l_{local} be the length of the local path $\mathcal{T}_{\text{local}}$ found by our method. We do not consider the effect of the approximation algorithm solving the TSP or require the path being kinodynamically feasible. $\mathcal{T}_{\text{local}}$ is assumed to be the shortest possible path connecting to E, F and fulfilling a coverage within $\mathcal{H}$. we have

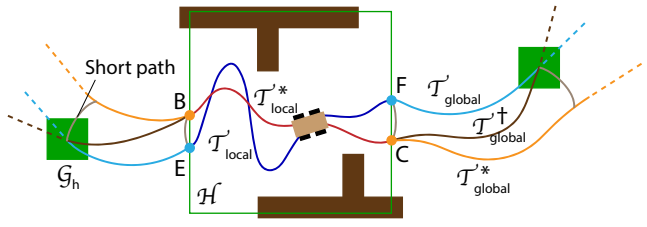


Fig. 5. Illustration of mathematical definitions in the proof of Theorem 3.

$$l_{\text{local}} \leq l^*_{\text{local}} + 2D_{\mathcal{H}}. \tag{10}$$

Eq. (10) can be proved by forming a new path with $\mathcal{T}^*_{\text{local}}$ and the two short paths connecting between B, E and C, F. If (10) does not hold, the new path becomes shorter than $\mathcal{T}_{\text{local}}$ and a contradiction occurs. Combining (8)-(10) gives

$$l_{\text{local}} + l_{\text{global}} \leq l^* + 4D_{\mathcal{H}} + 2mD_{\mathcal{G}}. \tag{11}$$

Therefore, the approximation ratio is bounded by

$$\sigma_{\text{hier}} \leq (l^* + 4D_{\mathcal{H}} + 2mD_{\mathcal{G}})/l^*. \tag{12}$$

■

In practice, the subspaces in $\hat{\mathcal{G}}$ are often located at intersections where the vehicle takes one branch and leaves exploring subspaces at the entrances of the other branches. These exploring subspaces are converted to "explored" when the vehicle revisits the intersections and completes the coverage. When the environment is large-scale, the exploring subspaces are usually sparsely distributed through the environment. With a large l^*, m/l^* is relatively small and σ_{hier} is close to one. Finally, considering that Theorem 3 does not take into account the segmented path smoothing, we analyze it below. Recall that p is the penalty for stopping the vehicle at a break-point and n' is the number of viewpoints on $\mathcal{T}'_{\text{smooth}}$. We reuse l_{local} and refer to the length of the local path where all $n'-2$ viewpoints are set as break-points. This path has the shortest length since no boundary conditions are applied at the viewpoints as compared to that any viewpoints are set as inner-points. We have

Theorem 4: The approximation ratio of cost c'_{smooth} by the path smoothing has $\sigma_{\text{smooth}} \leq (l_{\text{local}} + p(n'-2))/l_{\text{local}}$.

Proof: Since l_{local} is the length of the shortest local path, the cost of the theoretically optimal local path has

$$c^*_{\text{smooth}} \geq l_{\text{local}}. \tag{13}$$

In Algorithm 2, the algorithm starts with initializing all $n'-2$ viewpoints as break-points and reduces the cost by setting each viewpoint as an inner-point and re-smoothing the segment through the viewpoint. The resulting path has a cost

$$c'_{\text{smooth}} \leq l_{\text{local}} + p(n'-2). \tag{14}$$

Therefore, the approximation ratio is bounded by

$$\sigma_{\text{smooth}} \leq (l_{\text{local}} + p(\tilde{n}-2))/l_{\text{local}}. \tag{15}$$

■

Theorem 4 implies that our path smoothing is most suitable when p is relatively small as compared to l_{local}.

V. Experiments

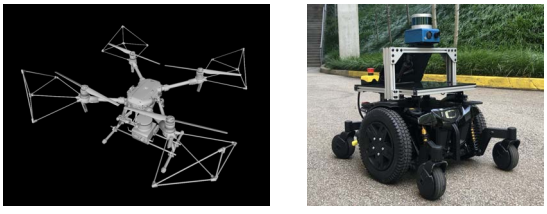
We conduct experiments in both simulation and real-world experiments. Fig. 6 shows our simulated aerial vehicle model and ground vehicle platform. The maximum speed is set to 5m/s for the aerial vehicle and 2m/s for the ground vehicle.

Both vehicles are equipped with a Velodyne Puck lidar, used for exploration and mapping. In addition, the ground vehicle has a camera at 640×360 resolution and a MEMS-based IMU, coupled with the lidar for state estimation [32]. The onboard autonomy system incorporates navigation modules from our benchmark environment, e.g. collision avoidance, terrain traversability analysis, way-point following, as the mid-layer, while the exploration algorithm is at the top layer.

The exploration algorithm runs on a 4.1GHz i7 computer using a single CPU thread. Our method re-plans at 1Hz. The configuration space in our method is evenly divided into blocks, where each block represents a subspace. For aerial vehicle experiments, the block size is set to 16m×16m×10m, and for ground vehicle experiments, the block size is set to 8m×8m×5m. The local planning horizon consists of 5×5×3 blocks with the vehicle in the center block. Point clouds are used to model surfaces, which is kept at 1.2m and 0.2m resolutions for the aerial and ground vehicle experiments. We compare our method against three state-of-the-art methods, all using open-source code adapted to the specific evaluation environments.

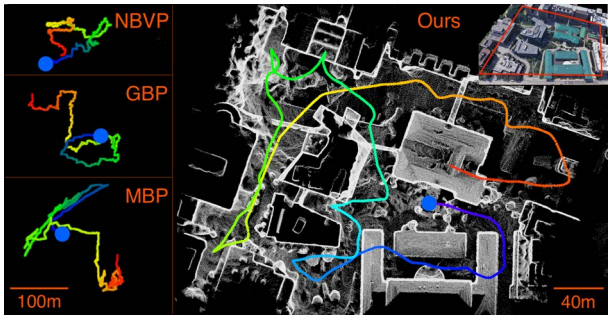
- *NBVP* [21]: A method spans an RRT in the free space and finds the most informative branch in the RRT as the path to the next viewpoint.
- *GBP* [23]: An extension of NBVP, where the method constructs an RRG in the global space and searches the RRG for routes to relocate the vehicle. The method explicitly switches between exploration mode and relocation mode.
- *MBP* [24]: A variant of GBP, which constructs the local RRT using motion primitives. The resulting paths are smoother but only span in constrained directions.

Test 1 uses the aerial vehicle to explore a university campus in simulation. Fig. 7(a) shows the resulting point cloud map and trajectory of our method in a representative run, with the trajectories of the other methods shown on the left. Each method is run 10 times starting at the same position indicated by blue dots. Our method is able to cover the entire environment using 1318m of travel on average and 366s in the longest run. The time limit for the other methods is set to four times of our longest run. Within the time limit, none of the three methods is able to completely explore the environment. Fig. 7(b) gives results on the explored volume over time. The dotted lines represent the upper-bound and lower-bound, and the solid lines represent the mean over the 10 runs. Table I compares the exploration efficiency for all methods. Here, the efficiency ϵ is defined as the average explored volume per

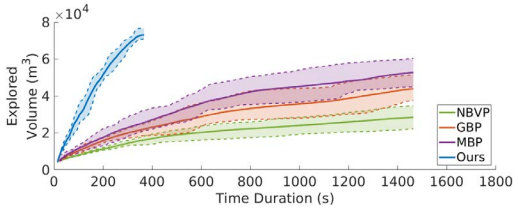


(a) (b)

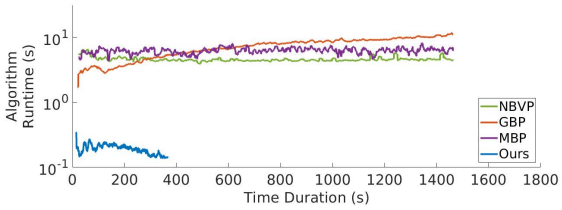
Fig. 6. (a) Simulated aerial and (b) Ground experiment platforms.



(a)



(b)

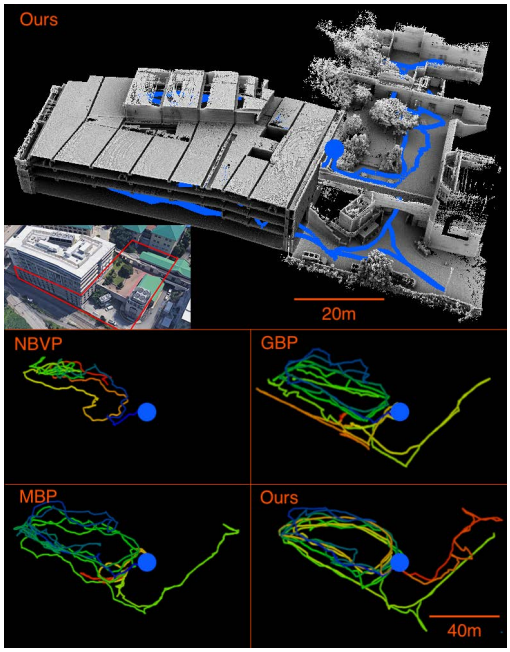


(c)

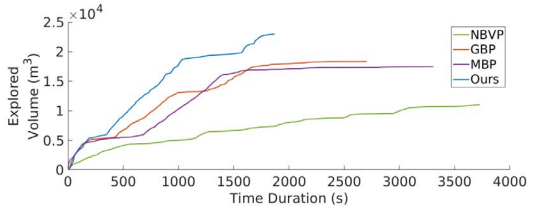
Fig. 7. Results of Test 1 with the aerial vehicle in simulation. (a) shows the resulting map and trajectory of our method from a representative run. Trajectories from NBVP, GBP, and MBP are shown on the left. The blue dots indicate the start point. (b-c) are exploration metrics for all methods.

second over the entire runs, and the relative efficiency r_ϵ is defined as the ratio compared to our method. Our method produces an efficiency more than 7 times of the other methods. Fig. 7(c) shows the algorithm runtime. The average runtime for NBVP is 4.6s, for GBP is 7.4s, for MBP is 6.3s, and for ours is 0.21s, where ours is a magnitude lower.

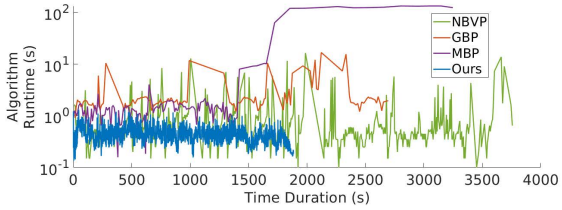
Test 2 uses the ground vehicle to explore a four-storage garage and a connected patio in real-world experiments. The start point is set at the entrance of the garage. During experiments, a run is terminated if the exploration algorithm reports completion, the vehicle almost stops moving (less than 10m of movement within 300s), or the time limit is met (set as twice of our method). Only our method is able to explore the whole environment and report completion after 1839m of travel in 1907s. All other methods terminate inside the garage and miss the patio connected to the garage from the top floor. In particular, NBVP reaches the time limit, and GBP and MBP are terminated due to they almost stop moving. Fig. 8(a) shows the resulting point cloud map and trajectory from our method, with trajectories from the other methods on the bottom. Fig. 8(b) compares the explored volume. Note that compared to Test 1, the topology in Test 2 is simpler, with fewer intersections and branches. Especially, all methods exhibit similar performance during the first half of the run, where the vehicle follows the main spiral driveway in the



(a)



(b)



(c)

Fig. 8. Results of Test 2 with the ground vehicle in physical experiments. The figure shares the same layout as Fig. 7.

garage from the top floor to the bottom floor. As indicated in Table I, our method is still 80% more efficient than the other methods for the overall run. Fig. 8(c) presents the algorithm runtime. The average runtime for NBVP is 0.88s, for GBP is 2.64s, for MBP is 10.13s, and for our method is 0.42s. Our runtime is 50% less than the other methods.

Table II presents the runtime breakdown for our method. One can see that majority of the processing is spent on local planning. Despite different scales and complexity levels of the environments in the two tests, the overall runtime of our method has the same scale. Further, we conduct tests with different $\mathcal{H}$ to inspect the corresponding algorithm runtime. Table III shows the result. As expected, the computation in local planning increases dramatically as $\mathcal{H}$ increases from the default value on the left. The largest $\mathcal{H}$ on the right covers the entire environment, i.e. the runtime for local planning is

TABLE I

COMPARISON OF EXPLORATION EFFICIENCY

Test	NBVP		GBP		MBP		Ours	
	ϵ	r_ϵ	ϵ	r_ϵ	ϵ	r_ϵ	ϵ	r_ϵ
Test 1	15.7	0.079	23.9	0.12	26.9	0.13	**199.7**	**1.0**
Test 2	3.0	0.24	6.8	0.55	5.3	0.43	**12.3**	**1.0**

TABLE II

RUNTIME BREAKDOWN

Test	Local Planning			Global Planning	Overall
	Update Representation	Sample Viewpoints	Find/Opt. Path		
Test 1	129.7ms	2.1m	45.8ms	28.6ms	206.2ms
Test 2	382.9ms	7.3ms	5.4ms	24.8ms	420.4ms

TABLE III

RUNTIME WITH DIFFERENT $\mathcal{H}$ FOR TEST 1

$\mathcal{H}$ (m)	80×80×30		160×160×60		320×320×120	
Average Runtime (ms)	Local	Global	Local	Global	Local	Global
	177.6	**28.6**	2197.6	27.9	5340.7	23.25

equivalent to our method reconfigured to not use the hierarchy. The result consolidates the strength of our hierarchical framework in producing high-efficiency processing.

Finally, our method is used by the CMU-OSU team in attending the DARPA Subterranean Challenge. Fig. 9 shows a representative result from a competition that takes place in Satsop Nuclear Plant, WA. Our vehicle fully autonomously explores the entire floor traveling over 886m in 1458s. Due to space issue, we eliminate the details of this result.

VI. CONCLUSION

We propose a method for highly efficient exploration of large and complex environments. Our method uses a hierarchical framework to plan detailed paths within the local planning horizon and coarse paths at the global scale. The method optimizes the full exploration path rather than greedily maximizes the marginal rewards. We provide theoretical analysis on the approximation ratio introduced by the hierarchy. Our method is evaluated against state-of-the-art methods in simulation and physical environments. Experiment results show that our method is 80% more efficient in covering spaces and consumes less than 50% of computation compared to the state-of-the-art.

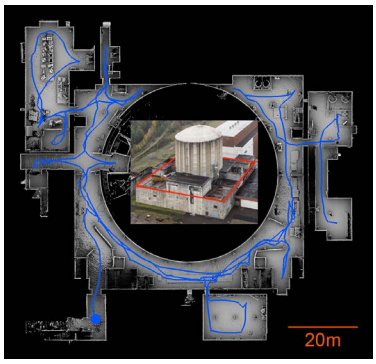


Fig. 9. Result from DARPA Subterranean Challenge in Satsop Nuclear Plant, WA. The photo shows the exterior of the building where the event takes place. Our vehicle travels over 886m in 1458s to explore the entire floor.

REFERENCES

[1] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, "Information based adaptive robotic exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, Oct. 2002.

[2] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters." in *Robotics: Science and Systems*, Cambridge, MA, June 2005.

[3] W. Tabib, M. Corah, N. Michael, and R. Whittaker, "Computationally efficient information-theoretic exploration of pits and caves," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, Oct. 2016.

[4] S. Bai, J. Wang, F. Chen, and B. Englot, "Information-theoretic exploration with bayesian optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, Oct. 2016.

[5] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[6] C. Nieto-Granda, J. G. Rogers III, and H. I. Christensen, "Coordination strategies for multi-robot exploration and mapping," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 519–533, 2014.

[7] M. Corah and N. Michael, "Efficient online multi-robot exploration via distributed sequential greedy assignment." in *Robotics: Science and Systems*, Cambridge, MA, July 2017.

[8] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997, pp. 146–151.

[9] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK)*, Munich, Germany, 2010.

[10] M. Kulich, J. Faigl, and L. Přeučil, "On distance utility in the exploration task," in *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: IEEE, May 2011.

[11] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," *Advanced Robotics*, vol. 27, no. 6, pp. 459–468, 2013.

[12] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, May 2015.

[13] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canda, Sept. 2017.

[14] M. Kulich, J. Kubalík, and L. Přeučil, "An integrated approach to goal selection in mobile robot exploration," *Sensors*, vol. 19, no. 6, p. 1400, 2019.

[15] J. Faigl and M. Kulich, "On determination of goal candidates in frontier-based multi-robot exploration," in *European Conference on Mobile Robots*, Barcelona, Spain, Sept. 2013.

[16] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, "Sensor-based exploration: Incremental construction of the hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 126–148, 2000.

[17] E. U. Acar and H. Choset, "Sensor-based coverage of unknown environments: Incremental construction of morse decompositions," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 345–366, 2002.

[18] S. Kim, S. Bhattacharya, R. Ghrist, and V. Kumar, "Topological exploration of unknown and partially known environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo, Japan: IEEE, November 2013, pp. 3851–3858.

[19] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[21] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3D exploration," in *IEEE international conference on robotics and automation (ICRA)*, Stockholm, Sweden, May 2016.

[22] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. Siegwart, "History-aware autonomous exploration in confined environments using mavs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018.

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

[24] M. Dharmadhikari, T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis, "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, May 2020.

[25] M. Roberts, D. Dey, A. Truong, S. Sinha, S. Shah, A. Kapoor, P. Hanrahan, and N. Joshi, "Submodular trajectory optimization for aerial 3D scanning," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, Oct. 2017.

[26] C. H. Papadimitriou, "The complexity of the lin–kernighan heuristic for the traveling salesman problem," *SIAM Journal on Computing*, vol. 21, no. 3, pp. 450–465, 1992.

[27] D. Li and X. Sun, *Nonlinear integer programming*. Springer Science & Business Media, 2006.

[28] D. Bergman, A. A. Cire, W.-J. van Hoeve, and T. Yunes, "BDD-based heuristics for binary optimization," *Journal of Heuristics*, vol. 20, no. 2, pp. 211–234, 2014.

[29] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[30] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[31] S. G. Johnson, "The nlopt nonlinear-optimization package," http://github.com/stevengj/nlopt, 2014.

[32] J. Zhang and S. Singh, "Laser-visual-inertial odometry and mapping with high robustness and low drift," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1242–1264, 2018.