Concave-Hull Induced Graph-Gain for Fast and Robust Robotic Exploration

Zezhou Sun, Banghe Wu, Chengzhong Xu, *Fellow, IEEE*, and Hui Kong, *Member, IEEE*

***Abstract*—Existing RRT-based exploration methods often suffer from interruptions in the exploration process due to the inability to detect all frontiers of the drivable area in the mapping map. These methods cannot detect all frontiers because the RRT expansion is disturbed by factors such as RRT preset parameters, sliding window constraints, complex external environment, etc, and thus cannot completely cover the drivable area within a limited time. We address this problem by redefining exploration frontiers, designing a novel exploration gain, and constructing minimum RRT search spaces. Our method is evaluated against the existing state-of-the-art RRT-based methods in simulated benchmarks and outdoor environments. The results show that our method is more robust to the above factors while reducing computational cost. Our method is made open source to benefit the community.**

***Index Terms*—Autonomous agents, motion and path planning, view planning for SLAM.**

I. Introduction

In THE last two years, the field of autonomous robotic exploration has shown great potential to replace human-controlled mapping. In general, the exploration methods aim to completely cover the mapped regions, find all the map frontiers, and plan a collision-free path toward them. Then, after prioritizing all frontiers, robots drive to the frontier with the highest priority. The Rapidly-exploring Random Tree (RRT) algorithm [1] shows the potential to solve this problem by its property of being heavily biased towards unknown regions of the map, which biases the tree to detect map frontier [2]. The Rapidly-exploring Random Graph (RRG) [3] formed by interconnecting the RRT nodes, which acts as a sparse sampling of the drivable region, allows the robot to quickly plan paths in it. Although it has been shown that RRT is of probabilistic completeness, we have to provide sufficient expansion time for RRT to cover the entire drivable space and find all the reachable map frontiers. However, this usually leads to frequent long pauses during the exploration process. Therefore, due to the real-time requirement or efficiency purposes, the common practice of existing methods [4], [5], [6], [7] is to limit the expansion time, the expansion range, or the number of nodes after expansion. But this causes the issue of missed frontiers, which results in incomplete exploration of an unknown environment.

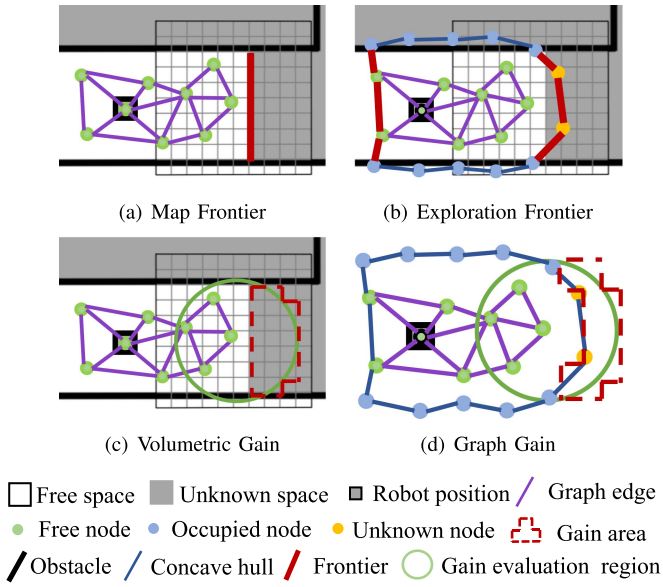


Fig. 1. (a) Map frontiers defined by existing RRT-based exploration methods. (b) Our redefined exploration frontiers. (c) Schematic of the gain area for the existing volumetric gain. (d) Schematic of the gain area for our graph gain. Our graph gain calculation is independent of the occupancy grid map.

To address this issue, we consider improvements in the following three aspects. **First**, existing RRT-based methods [4], [5], [6], [7] are devoted to finding map frontiers, which are the boundaries between free and unknown space in the occupancy grid map, as shown in Fig. 1(a). Differently, we redefine the *frontier*, called the exploration frontier, as special edges (Fig. 1(b)) on the concave hull construct by the RRT. Compared to the map frontier, exploration frontiers can be found at any stage of RRT expansion by constructing concave hulls without being affected by factors such as limited expansion time, occupancy grid map, etc. **Second**, in the existing RRT-based exploration methods [4],

Manuscript received 6 March 2023; accepted 5 July 2023. Date of publication 19 July 2023; date of current version 25 July 2023. This letter was recommended for publication by Associate Editor D. Schulz and Editor S. Behnke upon evaluation of the reviewers' comments. This work was supported in part by the Science and Technology Development Fund of Macau SAR under Grants AGJ-2021-0046, AFJ-2022-0123, AFJ-2023-0067, and SKL-IOTSC(UM)-2021-2023, and in part by the Startup Project of Macau University under Grant SRG2021-00022-IOTSC. *(Corresponding author: Hui Kong.)*

Zezhou Sun and Banghe Wu are with the Department of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: sun_zezhou@njust.edu.cn; banghe_wu@njust.edu.cn).

Chengzhong Xu is with the State Key Laboratory of Internet of Things for Smart City (SKL-IOTSC), Department of Computer Science, University of Macau, Macau 999078, China (e-mail: czxu@um.edu.mo).

Hui Kong is with the State Key Laboratory of Internet of Things for Smart City (SKL-IOTSC), Department of Electromechanical Engineering (EME), University of Macau, Macau 999078, China (e-mail: huikong@um.edu.mo).

Source-code: https://github.com/IMRL/Graph_Gain_Exploration.git

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3297062, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3297062

[5], [6], [7], the volumetric gain of each RRT node needs to be calculated to determine whether this node needs to be visited. The existing volumetric gains are designed for the map frontier in the occupancy grid map (Fig. 1(c)). In contrast, we design a novel gain for the redefined exploration frontiers, called graph gain, as shown in Fig. 1(d). We can calculate the graph gain for nodes at any stage of the RRT extension to determine the future visit order. **Third**, the update of the exploration frontiers depends on the extension of the RRT. We expect RRT to expand outward efficiently. However, unlike path planning problems given start and destination points, RRT does not know the number and location of frontiers in the exploration problem. Therefore, only inefficient conventional RRT methods [1] can be used in the exploration problem instead of improved methods that construct ellipses with the start and destination points as focal points, such as informed RRT* [8] or Batch Informed Trees (BIT*) [9]. In this regard, we propose a minimum search space construction method that takes the difference between the concave hull constructed by the RRT and the latest LiDAR frame coverage as the search space for future RRT expansion.

To sum up, this letter mainly solves the problem that existing RRT-based exploration methods cannot ensure finding all frontiers in a limited extension time, which leads to the abortion of exploration. The main contributions of this letter are as follows. (1) We redefine the frontiers to be searched by the RRT-based exploration method so that the robot can find all the frontiers to be explored at any stage of the RRT expansion. (2) A new type of exploration gain, graph gain, is designed to accurately compute the exploration gain of nodes on the exploration frontiers. (3) A minimum search space is constructed to motivate the RRT to expand and update the exploration frontiers more efficiently.

II. Related Works

During exploration, mobile robots can navigate autonomously and build a 3D model of an unknown environment without a priori. This involves an exploration module and a SLAM one. The exploration module can be divided into three stages: frontier detection, frontier prioritization, and execution. The robot detects a set of destination candidates close to the frontiers for future visits based on the current map. These destination candidates are ranked according to the predefined exploration gain. Then, the robot navigates to the highest-ranked destination. In the SLAM module, the robot can simultaneously create a map of the environment and localize itself in it. By interleaving the exploration and SLAM modules, the environment map can be dynamically updated and expanded with new observations. The focus of this work is on the exploration module and the SLAM module can use the classical SLAM method [10], [11].

Exploration modules expect to simultaneously detect destination candidates and plan collision-free paths to them, so path planners are often used to accomplish this task. Path planning algorithms can be generally classified into two categories: grid- and sampling-based planners [12]. Accordingly, existing exploration modules are also classified into grid-based and sampling-based depending on their planner. Both types of modules require searching for map frontiers in the map.

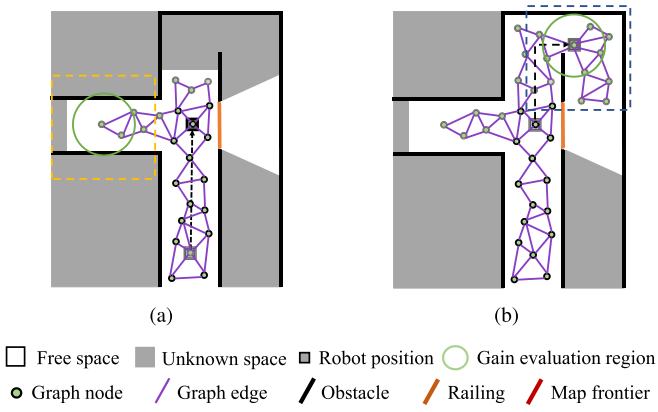


Fig. 2. Examples of existing RRT-based exploration methods [5], [6], [7] that miss map frontiers. (a) When the robot is at the intersection, the RRT does not cover all the free space on the left side (the white area in the yellow box). Since the RRT nodes in the yellow box are not close to the map frontier, there is no unknown space within the gain evaluation region of these nodes. As a result, the robot incorrectly believes that the left side of the intersection is fully explored and thus explores forward, resulting in the unknown area being ignored. (b) The RRT cannot be extended through obstacles, such as glass or railings, although the LiDAR can observe the space behind them. In this case, the limitation of the sliding window (blue box) prevents the RRT from extending to the entire free space, which leads to the missing of map frontiers and interruption of exploration.

The grid-based methods aim to detect dense map frontiers. As mentioned before, heuristic planners such as A* [13] or D* Lite [14] cannot be used since there is no explicitly pre-given goal before covering the free space. Yamauchi et al. [15] proposed the seminal work to find map frontiers by searching the entire map using Breadth-first search (BFS). The Wavefront Frontier Detection [16] reduces the search space from the entire explored space to the free space. The Expanding-Wavefront Frontier Detection [17] only incrementally searches in the latest explored areas. Hess et al. [18] perform frontier detection in the sub-maps, reducing the search space to the total frontier length of all previous sub-maps and excluding the effect of graph optimization on the map. Sun et al. [19] reduces the search space to the frontier length of the latest modified sub-maps.

With the need for outdoor large-scale environmental exploration, sampling-based exploration methods, especially RRT, have gained immense interest. RRT-based exploration methods have been widely used in exploration, such as the Receding Horizon Next-Best-View method (NBVP) [4] and its variants [20], [21]. The exploration gain of RRT nodes is designed as the unmapped volume explored on the path to the node, with a penalty for long-path nodes. The Multiple Rapidly-exploring Randomized Trees [2] enhance the efficiency of exploring local unknown regions by constructing the global and local RRTs. The search space of these methods is the free space that expands with the progress of robot exploration, which means that the RRT expansion efficiency continues to decrease. The decrease in RRT expansion efficiency implies an increase in the probability of missing frontiers within a limited expansion time, as shown in Fig. 2(a). Therefore, it can take an extremely long time to explore the complete environment using these methods.

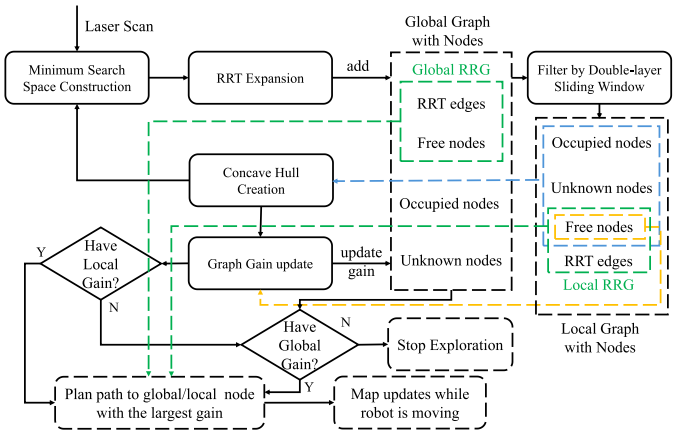


Fig. 3. The flowchart architecture of the proposed method.

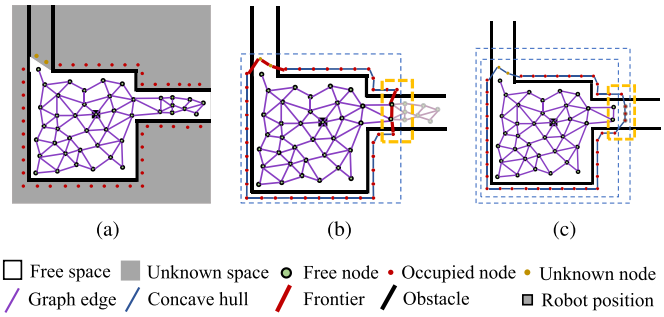


Fig. 4. (a) All nodes attempting to be extended during RRT extension are divided into three types "Free", "Occupied", and "Unknown". (b) A single-layer sliding window (blue dotted box) truncates the RRG. The truncation results in an undesired frontier because it is already explored (in the yellow box). (c) Nodes in RRT extensions are cropped using a double-layer sliding window, and concave hulls are constructed using these nodes.

Since map updates can only occur within the LiDAR observation range, the Graph-Based exploration path Planner (GBPlanner) [5] and Motion-primitive Based exploration path Planner (MBPlanner) [22] limit the expansion of the RRT by using a sliding window of the same size as the LiDAR observation range. The RRT can maintain efficient expansion in the sliding window without being affected by the expansion of the overall mapped region. Its exploration gain introduces a similarity function based on the NBVP method, using the Dynamic Time Warping method (DTW) to disfavor paths that greatly diverge from the currently estimated exploration direction. In addition, RRT nodes are interconnected to form a Rapidly-exploring Random Graph (RRG), which enables RRT to support multi-query path planning. On this basis, the Dual-Stage Viewpoint Planner (DSVP) [7] retains the viewpoints in the overlapping area between adjacent sliding windows to reduce the cost for the RRT to re-expand. During the RRT expansion, the DSVP method manually selects three potential regions and guides the sampling around them. Its variant [23] dynamically resizes the sliding window using an axis-aligned minimum bounding box. Subsequently, GBPlanner2 [6] uses Principal Component Analysis (PCA) to determine the bounding box orientation, which makes the bounding box better match the structure of the environment and further reduces the search space. However, the use of sliding windows makes the RRT lose probabilistic completeness, i.e., reachable frontiers exist, but the RRT constrained by the sliding window cannot be extended to the map frontiers, as shown in Fig. 2(b). Once the RRT cannot extend to the map frontiers, there is no unmapped space around the RRT nodes. Existing methods cannot get exploration gains around these nodes, so robots do not travel to these nodes, resulting in interrupted exploration. Therefore, there is a lack of an efficient and robust RRT-based exploration method that can avoid the interruption of exploration due to missed frontiers.

III. Methods

Fig. 3 shows the flowchart of our method. Our approach can be divided into three stages: minimum search space construction, concave hull creation, and graph gain update. The RRT is expanded in the minimal search space we construct, where the sampled nodes and the constructed RRT are stored. The stored sampled nodes are filtered by a double-layer sliding window and then these nodes are used to construct a concave hull. The free-state nodes inside the concave hull are calculated for their graph gain. The robot always drives to the node with the largest graph gain in the concave hull. If the graph gain of all nodes within the concave hull is zero, the robot turns to drive to the global node with the largest graph gain. If the graph gain of all global nodes is zero, the robot believes that the environment has been fully explored and then stops.

A. Concave Hull Creation

Generally, RRT retains only those nodes that are expanded in free space during expansion. For nodes expanded in obstacles or unknown areas, RRT discards them without any processing. However, we consider those discarded nodes, which we call expansion-failed nodes, to be as important as the retained nodes, which we call expansion-succeed nodes. These expansion-failed nodes can reflect information about the location of the obstacles and unknown areas, which are not reflected in the expansion-succeed nodes. An expansion-failed node indicates that RRT detects an obstacle or unknown area at the node location, where the RRT cannot expand further.

We further subdivide the expansion-failed and expansion-succeed nodes into "Free", "Occupied", and "Unknown" statues, as shown in Fig. 4(a). When the robot moves, we use a sliding window that moves with the robot to limit the number of nodes, and we only calculate the graph gains for the "Free" nodes inside the sliding window. Using a single-layer sliding window alone will create new undesired exploration frontiers at the truncation of the RRG (In Fig. 4(b) yellow box). We avoid this problem by using a double-layer sliding window. The size of the inner sliding window is the same as the LiDAR's observation range. Empirically, the spacing between the outer and inner layers is set to $2\,m$, which needs to be longer than the longest edge of the RRT. Nodes that are beyond the outer layer are discarded and the nodes in the inner layer are retained. In particular, the states of the nodes that are between the inner and outer layers are

considered as "Occupied" when constructing the concave hull, regardless of their previous states, as shown in Fig. 4(c).

We construct the concave hull with $O(nlogn)$ complexity using the Delaunay triangulation method [24], where n is the number of nodes involved in constructing the concave hull. In this way, those edges on the concave hull which only consist of "Free" or "Unknown" nodes are places where the RRT can extend outward in the future. Considering the passability of the robot, we set the maximum edge length of the concave hull as the robot's diameter. All nodes are down-sampled so that the node spacing is slightly larger than the robot's radius. In this way, we can ensure that the length of each edge of the constructed concave hull is larger than the robot's radius and smaller than the robot's diameter. Therefore, we redefine the *frontier* of the RRT search in this letter. Frontiers are no longer the boundaries between the free and unknown space in the map, but those edges on the concave hull that consist of "Free" or "Unknown" nodes.

B. Graph Gain Update

After finding all exploration frontiers it is necessary to calculate the exploration gain of the nodes to determine the priority of the nodes for future visits. In general, the exploration gain of a node v is affected by several factors, such as the volume of the unknown space around the node $VolumetricGain(v)$, the distance from the robot's current location v_0 to the node $D(v_0, v)$, the differences between nodes and current exploration directions $S(v)$, etc. We let $\{v_0, v_1, v_2, \ldots, v_n\}$ be the nodes that the robot passes along the RRT branch to node v_n. Existing exploration gain [5] of the node v_n is calculated as follows:

$$Gain(v_n) = e^{-\gamma_S S(v_n)} \sum_{i=1}^{n} VolumetricGain(v_i) e^{-\gamma_D D(v_{i-1}, v_i)} \tag{1}$$

where γ_S and γ_D are tunable factors greater than 0.

It can be seen that the exploration direction $e^{-\gamma_S S(v_n)}$ and distant $e^{-\gamma_D D(v_{i-1}, v_i)}$ are weight functions between (0,1), and the decisive factor is the volumetric gain. Nodes with large volumetric gain indicate that more unknown space can be explored and measured by visiting these nodes. The existing method considers the volume of the unknown region within the gain evaluation region around a node as the volumetric gain of this node. However, due to the incomplete extension of RRT, existing methods ignore those nodes that do not have unknown regions nearby but still need to be visited (e.g., the nodes in the box in Fig. 2. To avoid ignoring these nodes, we design the graph gain based on the concave hull we construct.

We consider the space outside the concave hull as the unknown space, and the graph gain counts the volume around the node beyond the concave hull. The unknown space here no longer represents the unmapped space, but the space uncovered by the concave hull constructed by RRT. The graph gain of the RRT node can be updated in two steps by checking the points in the polygon and finding the intersecting edges.

Firstly, the space within the gain evaluation region around the node N is discretized into grids. For each grid, we determine whether it is outside of the concave hull, as shown in Fig. 5(a).

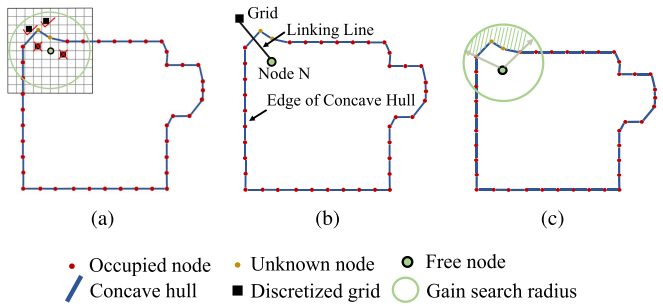


Fig. 5. (a) Step 1: Determine if the discrete grids are outside the concave hull. (b) Step 2: Traverse each edge of the concave hull to find the edge that intersects the linking line. (c) The graph gain is denoted as the volume of the shaded area within the gain evaluation region.

This is a point-in-polygon (PIP) problem that can be solved with $O(n)$ complexity by ray casting algorithm [25], where n is the number of polygon edges. For the grid outside the concave hull, we then look for which edge on the concave hull intersects the line linking this grid to the node N, as shown in Fig. 5(b). This is the problem of checking whether two line segments intersect, and it can be quickly solved by cross product, as shown in Algorithm. 1 (line 7). After finding the edge on the concave hull that intersects the linking line, we determine whether the edge can be traversed by checking the node label at both ends of the edge. If at least one of the nodes on the edge is "Free" or "Unknown", this grid is accumulated as graph gain (lines 8-9). The area of the graph gain counts is shown in the green shaded area in Fig. 5(c). The weight functions for the distance $e^{-\gamma_S S(v)}$ and exploration direction $e^{-\gamma_D D(v_i)}$ we keep consistent with the DSVP method. The exploration methods using volumetric gain tend to observe more space. Our method using graph gain tends to make the RRT cover all the drivable area. Therefore, for regions that are already mapped but not covered by the RRT, our method will make the RRT try to cover all already mapped regions again. In this case, the exploration efficiency of our method, which is generally considered to be the ratio of mapping volume to exploration time, will be lower than that of existing methods. In addition, since our work focuses on ground robots, the RRT extensions, concave hull construction, and graph gain are currently only applicable to 2D or slightly skewed environments.

C. Minimum Search Space Construction

The existing exploration methods only require the robot to drive to the node with the highest exploration gain, and the map frontier around the node is updated automatically by the SLAM module. And the update of our exploration frontiers depends on the future expansion of the RRT. Therefore, we need to guide the RRT to expand outward more efficiently.

At the time t, we set the robot's location to be l_t and the area observed by the LiDAR scan to s_t. The concave hull C_{t-1} constructed in the previous section can represent the local RRT coverage area at time $t-1$. Therefore, we regard the set difference of the latest LiDAR scan s_t and the concave hull

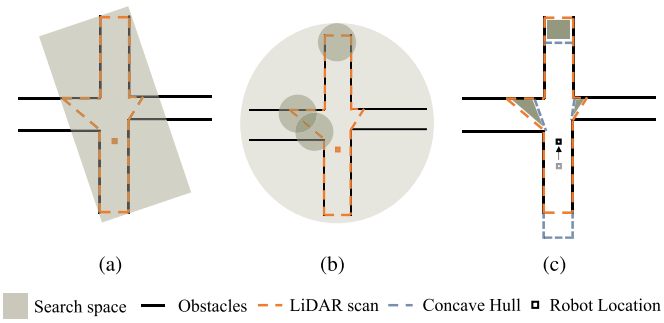


Fig. 6. Comparison schematic of representative RRT search spaces (a) is the oriented bounding box search space used by the GBPlanner2 method [6]. (b) is the search space of the DSVP method [7], in which three potential expansion regions are predicted. (c) is the search space of our method, which constructs by the set difference between the LiDAR's latest frame coverage range and the concave hull. The shade represents the sampling probability of the region. A darker shade indicates a higher probability of RRT search.

Algorithm 1: Graph Gain Update.

Input:
Concave Hull: C_t
RRT node: n
Output:
Graph gain: G_n

```
G_n = 0
for grid g in gain evaluation region around n do
    if g is outside concave hull C_t
        LinkingLine = (n, g)
        for edge e_i on the polygon C_t do
            (a, b) ← two endpoints of edge e_i
            if (n−b) × (a−b) * (g−b) × (a−b) ≤ 0
                if State(a) is "Free" or "Unknown" or
                  State(b) is "Free" or "Unknown"
                    G_N += 1
                end
            end
        end
    end
end
```

C_{t-1} as the minimum search space S_t of RRT expansion, i.e., $S_t = s_t - C_{t-1} = \{x \mid x \in s_t \, and \, x \notin C_{t-1}\}$. Fig. 6 visualizes the comparison of our minimum search space with that of the DSVP and GBPlanner2 methods. Compared to other methods, the sampling space generated by our method accurately outlines the newly explored region. The precise sampling space allows RRT to expand more efficiently.

To maintain the convenience of sampling, we generate the angles and ranges of the sampled nodes in the minimum search space in the polar coordinate system. However, it should be noted that the samples generated using polar coordinates do not satisfy a uniform distribution compared to uniform sampling in Cartesian coordinates. This difference can be neglected since the newly explored area is small.

IV. Experimental Results

A. Benchmark Environments

We evaluated our method, the GBPlanner2 method [6], and the DSVP method [7] in three simulated benchmark environments [26] (i.e., narrow indoor environment, multi-layer garage environment, and large scale tunnel environment) to validate the exploration performance of our method in different types of environments. The maximum speed of the vehicle is 3 m/s, and other parameters of our method and the GBPlanner2 method are consistent with the open-source parameters of the DSVP method. Each method was run 5 times in each environment and their average data (exploration efficiency ε, exploration volume V, exploration distance L, and the algorithm running time T) are shown in Table I, where exploration efficiency is the ratio of exploration volume to exploration time when exploration is completed and the algorithm running time is the total time, including sampling space construction time, RRT expansion time, exploration gain update time, and path planning time.

Our method and the DSVP method have similar exploration efficiency in all scenarios (the difference in efficiency is less than 7%), while the average efficiency of GBPlanner2 is only about 69% of our method. The main factor affecting exploration efficiency is the delay in frontier detection, i.e., lagging frontier detection can lead to inefficient exploration behaviors such as pausing or even turning back.

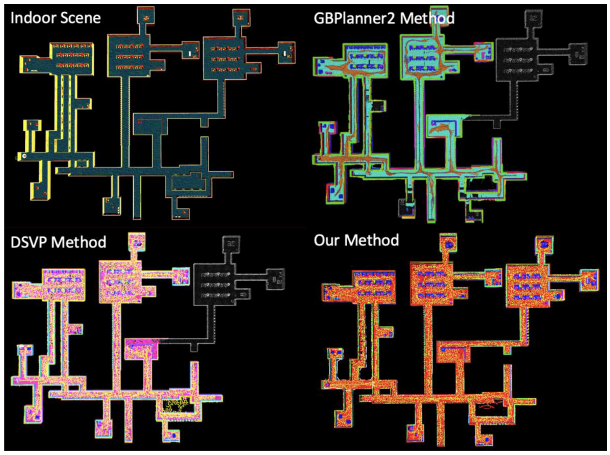
Both our method and DSVP method can complete the exploration with low frontier detection latency. Benefiting from the accurate division of the newly explored space and fast graph gain update, the average running time of our method is 74.8%, 86.9%, 65.5% of the DSVP method. Compared with our method and DSVP method, GBPlanner2 does not retain nodes in the overlapping area between adjacent sliding windows. Although it also has a short running time, it repeatedly expands in the overlapping region when all three methods expand the same number of nodes, which leads to a delay in frontier detection and thus reduces the exploration efficiency.

The robustness of frontier detection determines the exploration volume V and exploration distance L. Our method can completely explore the indoor environment while the DSVP and GBPlanner2 methods always fail, as shown in Fig. 7(a). Therefore, our method explores 8.6% and 33.1% more space than DSVP and GBPlanner2 methods, respectively, and has a correspondingly longer exploration distance. Both our method and DSVP method can fully explore the garage and tunnel environments with a short exploration distance, while GBPlanner2 may miss some areas or require a longer exploration distance. This indicates that our method also supports the robust and efficient exploration of multi-layer and large-scale scenes. We also present graphs of the instant variation of exploration volume and algorithm running time during indoor environment exploration, as shown in Fig. 7(b) and (c).

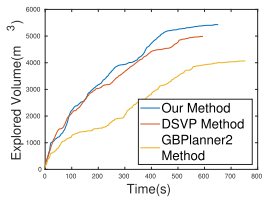
Specifically, we visualize the volumetric gain and graph gain in the intersection and room scenes by our and DSVP methods respectively, as shown in Fig. 8. In Fig. 8(a), the RRT is not fully extended as the robot passes the intersection from left to right, resulting in the insufficient covering of all areas of the

TABLE I
TABLE OF THE EFFICIENCY OF DIFFERENT METHODS IN THE THREE SCENARIOS

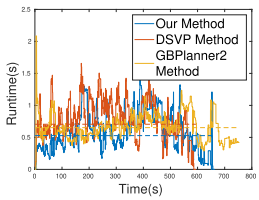
	Indoor				Garage				Tunnel			
	ε	V	L	T	ε	V	L	T	ε	V	L	T
GBPlanner2 Method	5.4	4067	1155	0.652	16.8	37264	4578	0.661	7.6	21967	6511	0.572
DSVP Method	8.4	4986	1484	0.705	22.7	42745	5556	0.710	10.3	21922	6463	0.701
Our Method	8.4	5415	1651	0.527	24.3	42687	5356	0.617	10.6	22206	6350	0.459



(a) Mapping results comparison



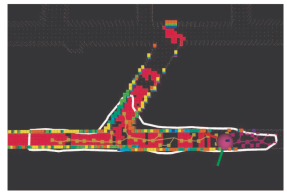
(b) Exploration volume comparison



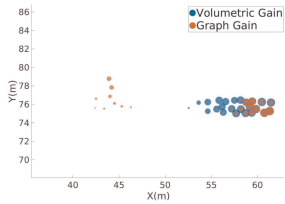
(c) Running time comparison

Fig. 7. Comparison results of the three methods in the indoor environment. (a) Top view of the indoor environment and the mapping results of GBPlanner2, DSVP, and our method, respectively. (b) The chart of the instant exploration efficiency of three methods, where the vertical axis is the exploration space volume and the horizontal axis is the exploration time. (c) The chart of the instant algorithm running time of three methods, where the vertical axis is the total running time and the horizontal axis is the exploration time.

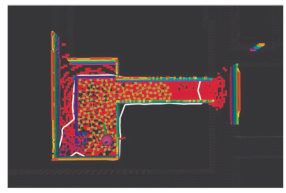
upward road. The node with the free-state in the concave hull is displayed. Due to being between the double-layer sliding window, some nodes, in the gap between the RRT and the concave hull on the left side of the intersection, temporarily changed from the free-state to the occupied-state and are therefore not displayed. Only the gain of the free-state nodes is calculated, which is visualized in Fig. 8(b). At this time, DSVP can only detect the gain information of the rightward road and mistakenly believes that there is no unknown region in the upward road. However, our method can robustly and accurately detect the gain information of the rightward and upward roads. In Fig. 8(c), the robot enters the room with a translucent obstacle (e.g., glass or railing), and the LiDAR can observe both the obstacle and the area behind the obstacle. RRT cannot bypass the obstacle and cover all areas in a short time, so the DSVP method cannot detect any map frontier, which leads to an interruption of exploration. In contrast, our method can still detect the exploration frontier, as shown in Fig. 8(d).



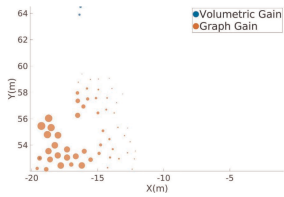
(a) Intersection scene



(b) Comparison of two gains

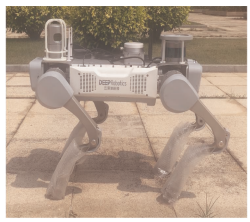


(c) Room scene

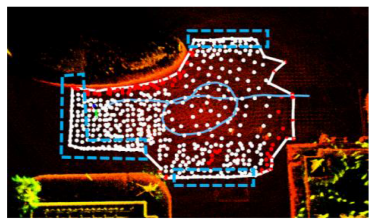


(d) Comparison of two gains

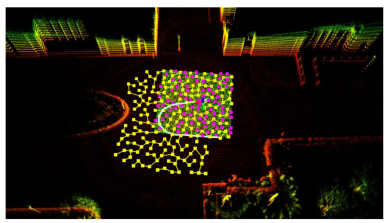
Fig. 8. Comparison of graph gain and volumetric gain in case of incomplete RRT expansion. In (a) and (c), the RRT is disturbed by intersections and railings, resulting in incomplete expansion. The volumetric gain and graph gain of each free-state node within the concave hull are plotted in (b) and (d). The center of the circle represents the node location in the exploration map and the radius of the circle indicates the value of the gain. In this case, the disturbed volumetric gain cannot be calculated as expected. However, our graph gain can detect the correct gain without being disturbed.



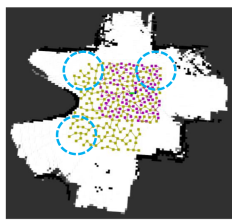
(a) Quadruped robot platform



(b) The mapping results of our method



(c) The mapping results of DSVP method



(d) Visualization of local occupancy map

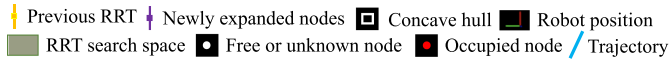


Fig. 9. (a) The RRT is restricted to extend within 5 m around the robot, while the mapping range is not restricted. (b) There is no unknown area around the nodes (blue circles indicate gain evaluation regions), so the DSVP method cannot find the map frontier. (c, d) Our method can find the exploration frontiers (in the blue dashed rectangle) based on the constructed concave hull and can robustly explore the surroundings.

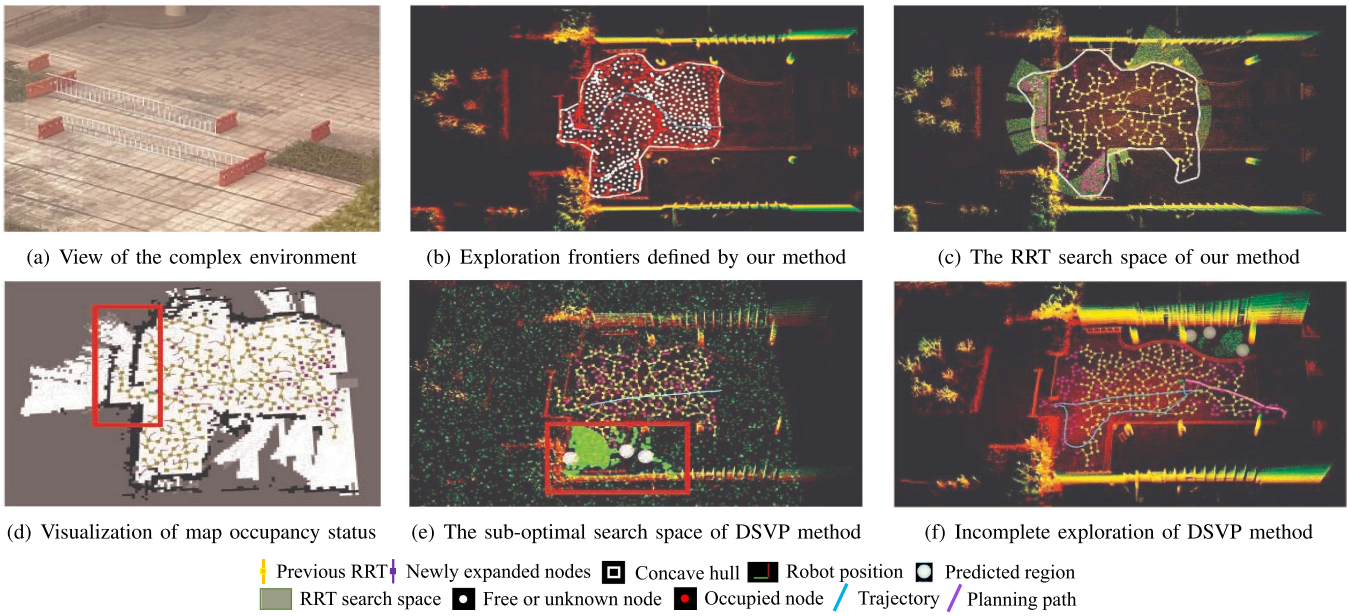


Fig. 10. (a) A narrow complex environment consisting of railings, where the RRT is difficult to expand. (b) When using our method, the robot drives to the exploration frontier between the railings. (c) Guided by the minimum search space (green area) we construct, the next expansion of RRT bypasses the railings. The concave hull is updated after the RRT extension. (d) There is no unknown space around the nodes in the red box, so the robot will not drive to these nodes when using the DSVP method. (e) RRT focuses on expanding in the regions predicted by the DSVP method and does not pay enough attention to other regions, which results in RRT not being able to bypass the railings. Each green dot represents one sampling of the RRT, and dense green areas indicate that the RRT has a higher probability of expanding in these areas. (f) The DSVP method is unable to explore the region behind the railings, thus leading to incomplete exploration.

B. *Outdoor Environment*

To validate the robustness of our method in the outdoor environments, we deployed our method and DSVP method on the quadruped robot, as shown in Fig. 9(a). The devices we used are a 32-line LiDAR on the head of the quadruped robot, a built-in IMU, and an onboard computer with an Intel i7 processor and 32 GB of RAM running ROS. Its maximum speed is 1 m/s.

We design two outdoor experiments to verify the robustness of our method. 1. **Rugged terrain environment**, which can only ensure accurate segmentation of the drivable area in a small range while mapping the surroundings. 2. **Complex environment**, which is complex and easy to miss frontiers.

As described in Fig. 2, the DSVP and GBPlanner2 methods must extend RRT to the boundary of the map in order to find the map frontier, so in the benchmark simulation environment, the robot's mapping range and terrain detection range are the same. However, it is difficult to precisely segment the drivable area for outdoor environment exploration, especially for areas far away from the robot. When the RRT expands in an inaccurately segmented area, the robot may try to drive to unreachable goals and thus waste a lot of time. An effective approach is to limit the expansion range of RRT so that it expands only within the accurately segmented area. This limitation on the terrain detection range can cause the DSVP and GBPlanner2 methods to fail to explore. The DSVP and GBPlanner2 methods solve this problem by limiting the mapping range to be equal to the RRT expansion range. Otherwise, it will abort because there is no unknown space around the node (Fig. 9(c) and (d)). However, limiting the mapping range will affect other tasks, such as the inability of the pose estimation module to match distant environmental features, and the inability of the segmentation and obstacle avoidance modules to observe distant vehicles and pedestrians in advance. In contrast, our method can build large-range maps while performing effective robust exploration over a small RRT extension range (Fig. 9(b)).

A narrow environment consisting of railings is shown in Fig. 10(a). Our method detects and drives to the exploration frontier between the railings (Fig. 10(b)) even when the RRT is unable to bypass the railings. During the robot's drive to the exploration frontier, the RRT expands within the set difference between the LiDAR's latest frame coverage range and the concave hull, effectively leading the RRT to bypass the railings and update the concave hull (Fig. 10(c)). In contrast, since the DSVP method has no volumetric gain at the nodes between the railings (the nodes in the red box in Fig. 10(d)), the robot cannot detect the map frontier nor does it drive toward those nodes between the railings. Then, according to the search space predicted by the DSVP method, the RRT focuses on the extension of the sub-optimal region, which makes the RRT not adequately cover the region behind the railings (Fig. 10(e)). Due to these factors, the DSVP method eventually fails to explore beyond the railings, which leads to the abortion of exploration (Fig. 10(f)).

V. Conclusion

In this letter, we propose an efficient and robust method for robot exploration. We construct concave hulls by RRT to represent the environmental information. The exploration frontier to be explored by the robot is defined as an edge on the concave hull consisting of free or unknown-state RRT nodes.

The graph gain of the nodes is designed as the volume around the nodes beyond the concave hull. The RRT future expansion region is constructed by the set difference between the latest LiDAR scan coverage and the concave hull. We evaluated our method against the state-of-the-art methods in the benchmark simulation environments and the outdoor environments. The results show that our method is more robust in detecting frontiers and exploring the environment while spending less computation.

REFERENCES

[1] S. M. LaValle et al., "Rapidly-exploring random trees: A new tool for path planning," Res. Rep. 9811, 1998.

[2] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1396–1402.

[3] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proc. Robot. Sci. Syst. VI*, 2010, pp. 267–274.

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

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

[6] M. Kulkarni et al., "Autonomous teamed exploration of subterranean environments using legged and aerial robots," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 3306–3313.

[7] H. Zhu, C. Cao, Y. Xia, S. Scherer, J. Zhang, and W. Wang, "DSVP: Dual-stage viewpoint planner for rapid exploration by dynamic expansion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 7623–7630.

[8] J. D. Gammell, S.S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 2997–3004.

[9] J. D. Gammell, S.S. Srinivasa, and T. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 3067–3074.

[10] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[11] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.

[12] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using RRT* based approaches: A survey and future directions," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 11, pp. 97–107, 2016.

[13] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[14] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 354–363, Jun. 2005.

[15] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Automat.*, 1997, pp. 146–151.

[16] M. Keidar and G. A. Kaminka, "Robot exploration with fast frontier detection: Theory and experiments," in *Proc. 11th Int. Conf. Auton. Agents Multiagent Syst.*, 2012, pp. 113–120.

[17] P. Quin, A. Alempijevic, G. Paul, and D. Liu, "Expanding wavefront frontier detection: An approach for efficiently detecting frontier cells," in *Proc. Australas. Conf. Robot. Automat.*, 2014.

[18] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1271–1278.

[19] Z. Sun, B. Wu, C.-Z. Xu, S. E. Sarma, J. Yang, and H. Kong, "Frontier detection and reachability analysis for efficient 2D graph-SLAM based active exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2051–2058.

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

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

[22] M. Dharmadhikari et al., "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 179–185.

[23] Z. Sun, B. Wu, C. Xu, and H. Kong, "Ada-detector: Adaptive frontier detector for rapid exploration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 3706–3712.

[24] A. Maus, "Delaunay triangulation and the convex hull of n points in expected linear time," *BIT Numer. Math.*, vol. 24, no. 2, pp. 151–163, 1984.

[25] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, "A characterization of ten hidden surface algorithms," *ACM Comput. Surv.*, vol. 6, no. 1, pp. 1–55, 1974.

[26] C. Cao et al., "Autonomous exploration development environment and the planning algorithms," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 8921–8928.