Scalable and Expert-guided Reinforcement Learning-based Autonomous Robot Exploration

Yuhong Cao, Yizhuo Wang, Jingsong Liang, Guillaume Sartoretti

***Abstract*— This work further pushes the boundary of learning-based methods in autonomous robot exploration in terms of scalability and exploration efficiency. In particular, we focus on ground robot autonomous exploration with omnidirectional 3D-LiDAR. In this work, we present HEADER, an attention-based reinforcement learning approach with hierarchical graphs for efficient exploration in large-scale environments. HEADER follows existing conventional methods to construct hierarchical representations, but further designs a novel community-based algorithm to efficiently partition the dense local graph into the shape-adaptive global graph. Leveraging a guidepost design to incorporate the global reference path into local decisions, we further empower our attention-based model to better capture dependencies during exploration to produce near-optimal exploration behaviors. While existing learning approaches all train models with handcrafted but biased rewards, we propose a parameter-free privileged reward that strictly aligns with the exploration objective. In the experiments, HEADER demonstrates better scalability than most existing learning and non-learning methods, while achieving significant improvement in exploration efficiency over the state-of-the-art baselines.**

I. Introduction

In autonomous exploration considered in this work, a mobile ground robot is tasked with exploring and mapping an unknown environment as fast as possible. By planning and executing an exploration path, the robot classifies *unknown* areas into *free* or *obstacle* areas based on its accumulated sensor measurements. In this work, we focus on tasks where a ground robot is equipped with an omnidirectional 3D LiDAR to obtain long-range, low-noise, and dense point cloud measurements. Recent advancements in LiDAR odometry have enabled accurate and robust localization and mapping at the scale of thousands of meters [1]–[3], allowing recent planners to focus on exploring the environment without concerns about the mapping accuracy [4]–[9]. Despite this, few existing planners support exploration in large-scale and complex environments [5], [10], due to the complexity that comes with long-term, real-time path planning requirements. That is, to achieve efficient exploration, the planner must actively react to belief and map updates at a high frequency by re-evaluating the full partial belief and replanning a long-term, non-myopic exploration path.

Reasoning about such a large-scale map at full resolution and at a high frequency is an impossible mission. Therefore, most recent advanced large-scale exploration planners rely on hierarchical planning [4], [5], [8], [9], [11], which decouples the exploration path into a low-resolution global path with coarse inference on the full belief and a high-resolution local path with fine reasoning on the nearby local belief.

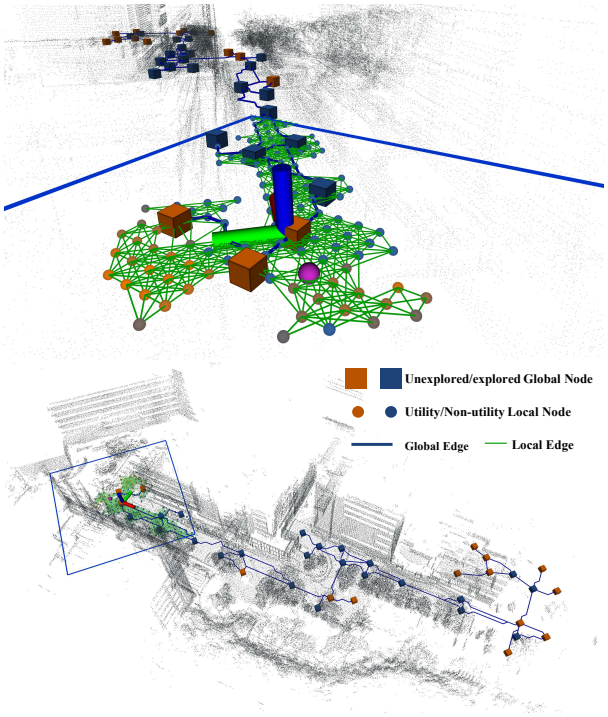


Fig. 1. An example hierarchical graph constructed by HEADER during exploring our campus. While dense local nodes keep expanding to cover all traversable areas, our planner adaptively and incrementally partitions these local nodes (bounded by the blue box as the local range) into global nodes to generate the global reference path. Based on the local graphs and the global path, our trained model selects a neighboring node as the next waypoint (denoted as the purple ball) to visit.

The insight of hierarchical planning is that fine reasoning is most effective in nearby areas, while less detailed inference is sufficient for utility far away. Given that the global path may change significantly after the local belief is updated, hierarchical planning offers a good balance between scalability and planning quality. We note that how sparse the global representation is determines the scalability of exploration planning [5], [10]. Existing methods [5], [9] often rely on a predefined global resolution, which makes them sensitive to this parameter: high resolutions will reduce planning efficiency, while low resolutions risk failing to fully cover the belief space.

Learning-based methods are intuitively promising to tackle exploration since they take advantage of predicting future states (i.e., unknown areas) based on previous experience, instead of only relying on current states (i.e., the current free areas) like conventional planners. However, existing

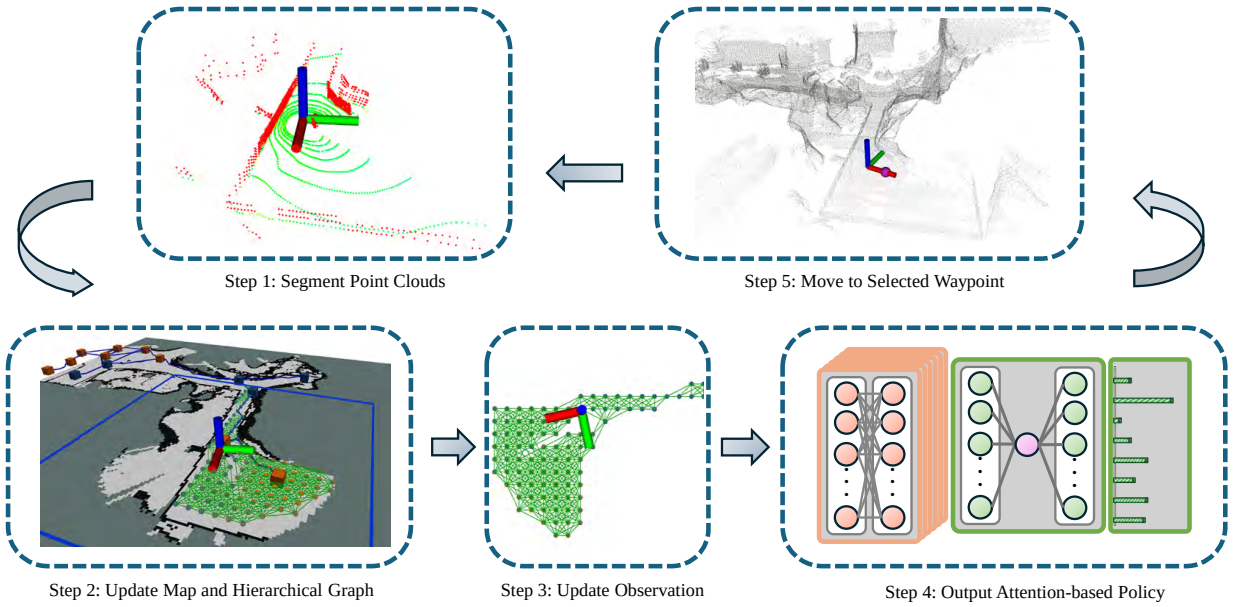


Fig. 2. The working flow of HEADER. We first classify the sensor scan into free and obstacle point clouds, which are used to build the current map of the environment. Then we construct the community-based hierarchical graph to generate the graph observation as the input of the trained neural network. After that, our attention-based decision model outputs the next waypoint for the robot to follow and get new measurements.

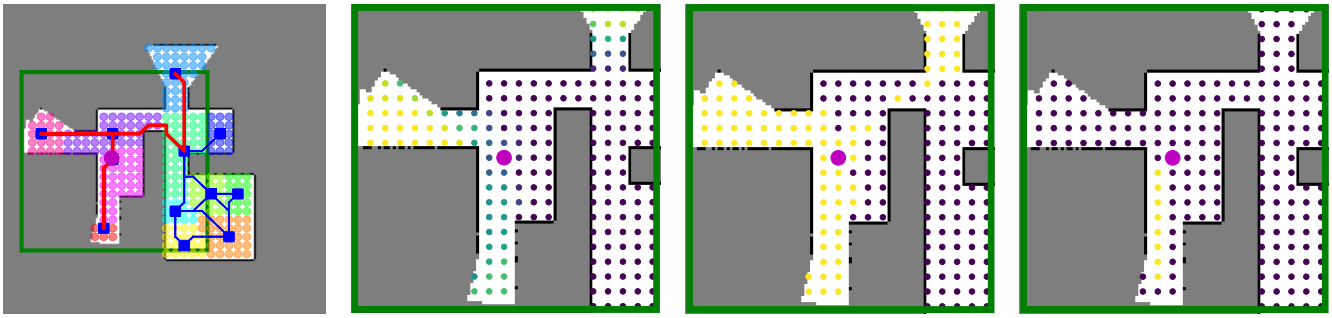
learning-based works have yet to explore how to integrate with hierarchical planning, since most existing learning-based planners still struggle to achieve good performance (i.e., exploration rate, makespan, and sim-to-real robustness) in small/local-scale exploration [12]–[16]. As a result, the scalability and exploration efficiency of existing learning-based planners are still far from matching the conventional state-of-the-art methods.

In this work, we further push the boundary of learning-based robot exploration by integrating our attention-based planner with hierarchical graphs and tackling the reward bias issue, making our planner the first learning-based approach that achieves better scalability and exploration performance than the state-of-the-art conventional planners. In particular, we introduce HEADER, which substantially extends our previous work [17], [18] by: (1) enabling powerful scalability through a novel global graph representation based on community detection [19]; (2) enhancing the model's long-term reasoning ability through guidepost features; (3) significantly improving its exploration performance through a privileged expert training paradigm. We first propose a global graph construction method to enable efficient global planning, which can incrementally generate shape-adaptive subregions for arbitrary structures without parameter tuning. Novelly, our method leverages community detection [19] to adaptively partition the dense graph into global subregions by identifying clusters of nodes with high intra-community connectivity and structural cohesiveness [20]. Then, we design guidepost features as part of the attention decision network's inputs. Our guidepost features contain a set of local and global reference paths. Those guideposts not only enhance the long-term reasoning performance in complex environments but also integrate global paths into our attention-based local decisions to scale up with hierarchical planning. Last but not least, we propose the privileged expert training paradigm to tackle the sparse reward problem. At each training step, by allowing an expert planner to access the full environment, we compute an optimal coverage path from the current position. The reward is calculated based on the Euclidean distance between the privileged expert action and the action planned by the network, which constantly provides dense feedback for actions in arbitrary environments.

Through these novel contributions, HEADER demonstrates state-of-the-art exploration efficiency (on average 20% higher than the second-best planner [5] in the benchmark tests) and scalability (completing exploration in large-scale environments where most recent open-sourced planners [6], [7], [9] fail). We validated HEADER in multiple outdoor and indoor simulation benchmarks, as well as on hardware in real-life scenarios, including a $300m \times 230m$ outdoor environment, where HEADER drives a wheeled robot to travel over 1.5 kilometers to explore over $60,000m^3$ of our campus.

II. Problem Formulation

Let $\mathcal{M} \subset \mathbb{R}^3$ be the current map/belief of the environment to be explored. Define $\mathcal{M}_{free} \subseteq \mathcal{M}$ as the known free/traversable space, $\mathcal{M}_{obs} \subset \mathcal{M}$ as the known obstacle/occupied space, and $\mathcal{M}_{unk} \subset \mathcal{M}$ as the unknown space. We then define $\mathcal{M}_{fron} \subset \mathcal{M}$ as the frontier, the boundary between free space and unknown space. At each decision step t, the ground mobile robot plans and visits the next waypoint $w_t \in \mathcal{M}_{free}$ in the traversable space to observe the environment. The newly perceived unknown space by an omnidirectional Lidar with a sensor range



(a) Global Graph (b) Node Utility (c) Local Guidepost (d) Global Guidepost

Fig. 3. An example of the community-based hierarchical graph constructed by HEADER. The purple dot indicates the robot's current position, while the green square outlines the local graph region. Small dots denote local nodes, and blue squares indicate global nodes. The blue paths denote the global edges, and the red path is the planned global path. In Fig. 3a, the colors of local nodes represent their community memberships. In Fig. 3b, the colors reflect node utilities. Fig. 3c and 3d highlight the selected local and global guideposts, respectively. The local guideposts denote paths to all utility nodes, and the global guideposts denote paths to the next global node.

d_{sensor}, will be classified as free space or occupied space based on the traversability. We denote the executed trajectory as $\psi = [p_0, p_1, ..., p_t], \forall p_i \in \mathcal{M}_{free}$, where p_i denotes the position visited by the robot. We consider the exploration tasks as completed when no frontiers are remaining, such that $\mathcal{M}_{fron} = \emptyset$. The objective is to minimize the cost of the executed path to complete the exploration:

$$\psi^* = \underset{\psi \in \Psi}{\operatorname{argmin}} \mathrm{C}(\psi), \text{ s.t. } \mathcal{M}_{fron} = \emptyset, \tag{1}$$

where ψ^* is the optimal exploration path, $\mathrm{C} : \psi \to \mathbb{R}^+$ maps a trajectory to its length, and Ψ is the space of all feasible exploration paths.

III. Methodology

In this section, we detail the working flow of HEADER (see Fig. 2) and the training paradigm. We first uniformly distribute candidate viewpoints in the free space and connect each viewpoint with its collision-free neighbors to construct a local graph that densely covers the current traversable space. We then run the community detection algorithm to incrementally and adaptively partition the local graph to construct the global graph (see Fig. 3a). Based on the global graph, we generate a global reference path by solving a TSP problem to visit all unexplored global nodes. After that, we formulate an informative graph that enhances the nearby local graph with frontier information (see Fig. 3b) and reference paths (see Fig. 3d and 3c). Finally, we input this informative graph into our attention-based network (see Fig. 4), which selects one of the neighboring viewpoints as the next waypoint to visit.

A. Community-based Hierarchical Graph

1) Local Graph: We first construct a *full dense graph* $G_d = (V_d, E_d)$ by incrementally and uniformly adding candidate viewpoints/waypoints at node resolution Δ_{node} in the free space. The node set is defined as $V_d = \{v_1, v_2, .., v_n\}, \forall v_i = (x_i, y_i) \in \mathcal{M}_{free}$. For each viewpoint v_i, we identify its neighboring viewpoints through line-of-sight checking (i.e., the distance between two viewpoints is within a threshold d_n and the straight line between them lies entirely in the free space). The neighboring connections formulate a set of collision-free edges $E_d = \{(v_1, v_2), ..., (v_{n-1}, v_n)\}$. At each decision step t, we define the *robot node* v_{cur} as the nearest node to the robot's current position p_t. Then we define a $d_{local} \times d_{local}$ sliding window $\mathcal{W}$ centered at v_{cur} and construct the *local graph* $G_l = (V_l, E_l)$, where the local node set $V_l \subseteq V_d$ contains all candidate nodes within $\mathcal{H}$, and the local edge set $E_l \subseteq E_d$ includes all edges between nodes in V_l.

2) Global Graph: Leveraging the Leiden algorithm [19], we adaptively partition the local graph G_l to incrementally construct a sparse *global graph* $G_g = (V_g, E_g)$, while ensuring full coverage of the robot belief (i.e., all local nodes remain reachable through the global graph). The Leiden algorithm aims to maximize the modularity of the partition, which is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} [A_{ij} - \beta \frac{k_i k_j}{2m}] \delta(c_i, c_j), \tag{2}$$

where A_{ij} is the adjacent matrix of the local graph, $k_i = \sum_j A_{ij}$ is the degree of node i, $m = \frac{1}{2} \sum_{i,j} A_{ij}$ is the total number of edges in the graph, β is a linear resolution parameter, c_i denotes the community that node i belongs to, $\delta(c_i, c_j)$ is an indicator function, which equals 1 if nodes i and j belong to the same community and 0 otherwise. Modularity evaluates whether the partition result is more structured than the random partition. High modularity indicates a community structure with dense intra-community and sparse inter-community connections.

Starting with an initial partition $\{C_1, C_2, ..., C_k\}$, for each node i, we move it from the current community to its neighboring community (i.e., the community that its neighboring node belong to and not identical to the current community), if modularity gain $\Delta Q > 0$. This node movement process continues until the modularity stops increasing. We then refine the community partition to ensure that nodes in each community are internally connected. We perform the connectivity check based on E_l to split communities that are

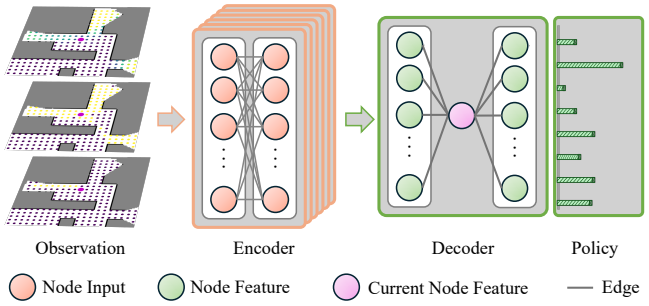


Fig. 4. The attention-based decision networks in HEADER. It takes the local graph, utility, and guideposts as input and outputs a probability distribution over neighboring nodes of the current robot position as the policy.

not fully internally connected. Subsequently, we merge the subcommunities with their neighbors if the modularity can be improved by doing so. To ensure consistency of the partition results, we fix the community assignments of existing nodes and only perform community assignments for newly added nodes at the current decision step. We also set a threshold based on $\mathcal{W}$ and Δ_{node} to limit the maximum number of nodes allowed in each community. To formulate G_g, for each community, we generate a global node $g_i \in V_g$ at the nearest local node to the center of the community and find its path and cost to neighboring global nodes by running A* [21] on the local graph. We term the global node corresponding to the robot node's community as the *current global node* g_{cur} and relocate its position to the same position as the robot node v_{cur}.

3) Incorporating Frontiers: Now we incorporate frontier information into the hierarchical graph. For each local node v_i, we compute its *utility* u_i, which represents the number of observable frontiers from the position v_i located within a utility range $d_{utility}$ (smaller than the sensor range). We classify the local nodes into *utility nodes* and *non-utility nodes*, depending on whether their utility value satisfies $u_i > 0$. A global node is classified as an *unexplored global node* or an *explored global node*, depending on whether it contains at least one utility node.

4) Reference Paths: Moreover, we compute two types of reference paths: a *global reference path* P_g and *local reference paths* P_l for the following decision-making. The global reference path is the shortest path to visit all unexplored global nodes starting from g_{cur}. We obtain this path by solving a TSP problem using OR-Tools [22]. Each local reference path corresponds to the shortest path from v_{cur} to a utility node. We obtain these paths using Dijkstra's algorithm [23]. In all cases, the initial portion of the current optimal exploration path lies within the set of these reference paths.

B. Attention-based Decision

See the details of our attention-based model in [17], [18]

C. Training with Privileged Expert Reward

1) Privileged Expert-guided Reward: We first introduce our privileged expert planning algorithm, which is probabilistically complete to produce (near-)optimal coverage paths as demonstrations for training the policy network. We grant the planner access to the ground-truth environment, making it a privileged expert. By doing so, the partially observable exploration problem is transformed into a fully observable sensor coverage problem, which is easier to tackle as it eliminates the need to handle the uncertainty that lies in the unknown environment. We modify the frontiers coverage algorithm proposed in [5] to plan a path that guarantees to cover all free space. Let $\mathcal{M}^*_{free}$ be the *ground truth free space*, $\mathcal{M}^*_{obs}$ the *ground truth obstacle space*. We denote the boundary between ground truth free space $\mathcal{M}^*_{free}$ and the unexplored obstacle space $\mathcal{M}^*_{obs} \setminus \mathcal{M}_{obs}$ as the *privileged frontiers* $\mathcal{M}^*_{fron}$. We note that finding the shortest path to cover the entire ground truth free space is equivalent to covering all privileged frontiers. To this end, we first sample a set of viewpoints that collectively observe all frontiers, and then solve a TSP to compute the shortest path that visits each viewpoint. We iterate this process multiple times and select the path with the minimum travel distance as the privileged expert path. To maintain the consistency between the expert action space and the learned policy space, similar to the dense graph G_d, we construct a *privileged graph* $G^* = (V^*, E^*)$, where $V^* \supseteq V_d, E^* \supseteq E_d$ as the planning space. Note that there is no hierarchical representation since we aim to ensure the optimality of the planned path for training.

At each decision step t, we compute the privileged expert path ψ^*_t and take the next waypoint in ψ^*_t as the expert waypoint w^*_t. We formulate the privileged expert reward r_t based on the normalized Euclidean distance d_t between the expert waypoint w^*_t and the waypoint w_t selected by the learned policy π_θ:

$$d_t = ||w^*_t - w_t||. \tag{3}$$

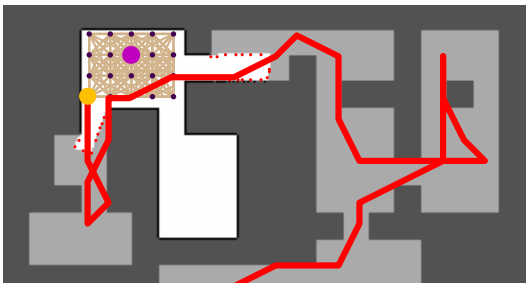


Fig. 5. Illustration of the privileged expert path. The white area denotes the explored free space, while the light gray region indicates unexplored free space. The red trajectory represents the privileged expert path planned using ground-truth information. The yellow dot marks expert path's first point as the expert waypoint. The light purple dot indicates the robot's current position, and the dark purple dots represent the candidate waypoints. The privileged reward is based on the distance between the selected waypoint and the expert waypoint.

We define the reward r_t as:

$$r_t = -\frac{e^{d_t/2d_n} - 1}{e - 1}, r_t \in [-1, 0] \tag{4}$$

where d_n is the neighboring threshold to rescale d_t between $[0, 1]$. The exponential term further encourages the policy to remain close to the expert's action. Compared to prior DRL-based methods that rely on carefully tuned, biased rewards [12]–[17], our privileged expert reward is both simpler and consistent with the true objective.

We train our policy network with soft actor critic, more details could be found in [17], [18].

IV. Experiments

We compare HEADER against state-of-the-art exploration planners in large-scale, realistic Gazebo simulations, covering both indoor and outdoor benchmark scenarios of up to $330m \times 250m$. Finally, we deploy HEADER on physical hardware in three real-world environments, again including both indoor and outdoor scenarios of up to $300m \times 230m$. Note that we use an identical trained model in all the experiments.

A. Comparison Analysis

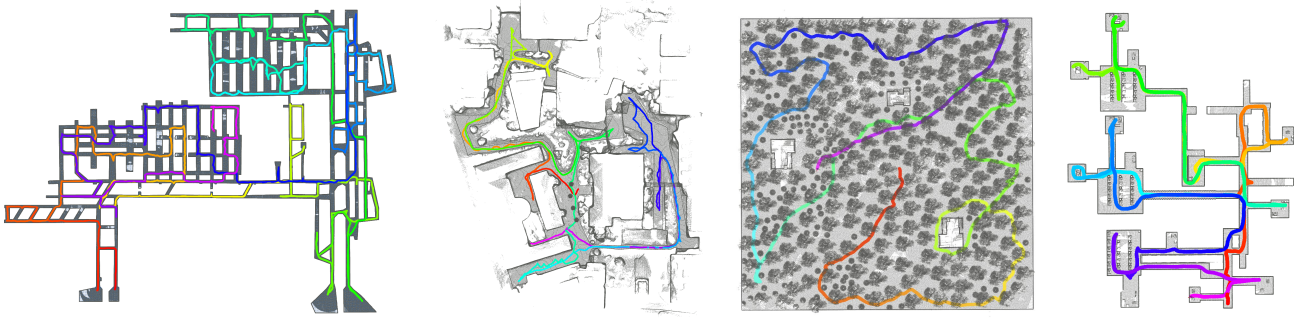
We integrate HEADER into the robot operating system (ROS) to test and compare it with the state-of-the-art open-sourced exploration planners, including: (1)**TARE** [5], (2) **DSVP** [11], (3) **GBP** [6], (4) **FAEL** [7], (5) **HPHS** [9], (6) **ARiADNE** [18]. Note that ARiADNE here is integrated with a graph rarefaction algorithm as in [18] to enhance performance in large-scale exploration, and all graph settings (e.g., node resolution and neighboring threshold) are the same as those in HEADER. We test the exploration performance of HEADER and the baseline planners in four exploration benchmark environments proposed in [24]: a $340m \times 340m$ campus environment, a $130m \times 100m$ indoor corridors environment, a $150m \times 150m$ forest environment, and a $330m \times 250m$ tunnel network environment. In these benchmarks, the test platform is a four-wheeled differential-drive robot equipped with a 16-channel 3D LiDAR, and the max speed is $2m/s$. We select these benchmarks since they are the largest and most realistic testing environments, to the best of our knowledge. In the campus environment, we block off certain areas, as the current implementation of HEADER cannot handle multilayer environments, where free space may exist above or below occupied regions. All the test runs on an ASUS mini PC with Intel i7-12700H CPUs (which is also used in our hardware validation). Note that GPU is not necessary for HEADER after training.

For different environments, we only tune two parameters for HEADER: the node resolution Δ_{node}, which ranges from $1.2m$ to $2.8m$, and the planning frequency, which ranges from $1Hz$ to $2.5Hz$. The baseline planners typically have more parameters to tune to get the best performance for each environment. We made our best effort to tune them to improve their performance, except TARE and DSVP, which are tuned by their authors. To enable the usage of HEADER in environments with non-flat ground, we develop a terrain segmentation tool based on a point cloud analysis module proposed in [24] and Octomap [25], which classifies voxels as traversable or occupied. The classified voxels are then projected to formulate an evaluation map as the input of HEADER.

We run HEADER, ARiADNE, TARE, DSVP 10 times and HPHS, FAEL, GBP 5 times in each environment. We also analyze and visualize some key results in Fig. 7. Among all the baselines, TARE demonstrates the best scalability, making it one of the only two methods capable of completing the exploration task in all four environments.

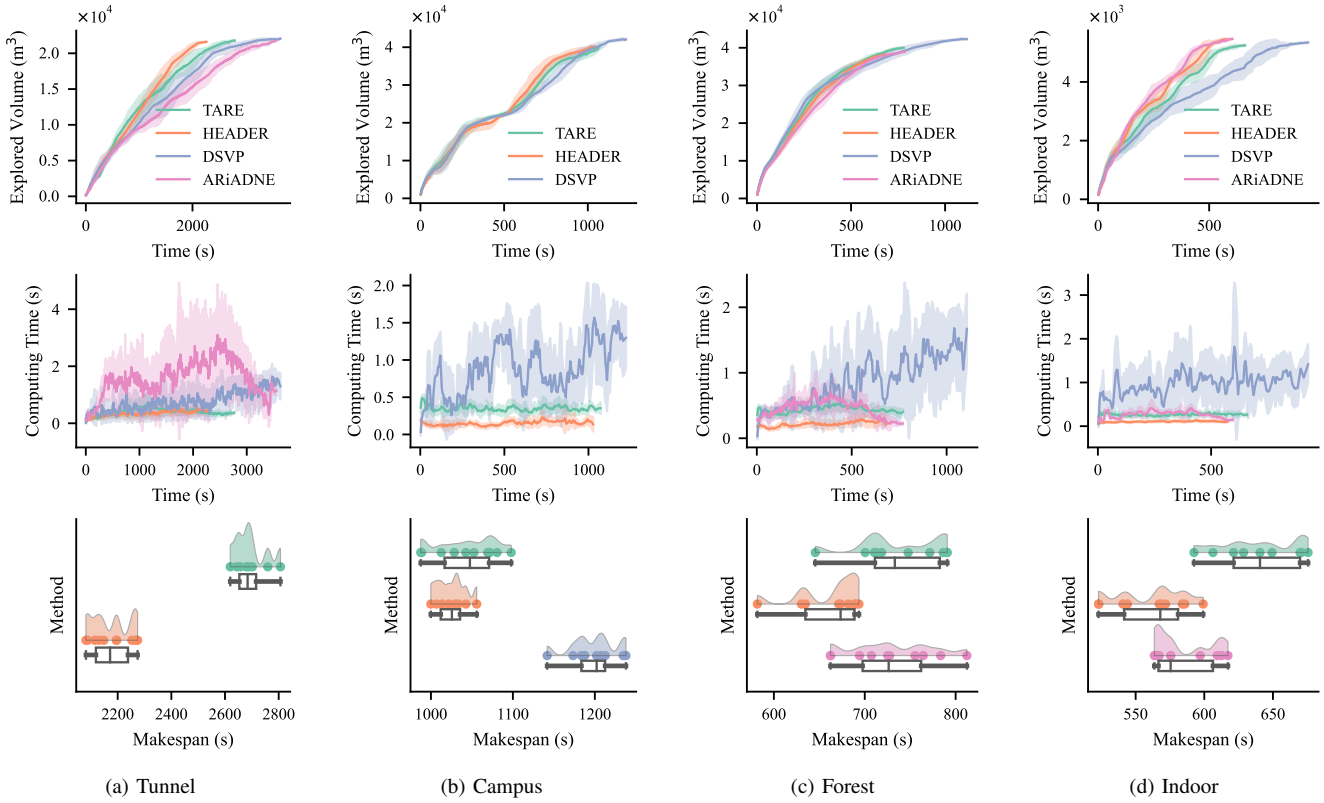
We note that HEADER consistently outperforms both TARE and ARiADNE across all tested environments. Upon closer examination, we attribute HEADER's superior performance to three key aspects:

1) The learned local decisions better avoid backtrack behaviors: **HEADER is able to reason about critical unknown areas, enabling it to make decisions that effectively avoid redundant backtracking behavior**. Conventional methods, by contrast, often suffer from inefficient detour exploration paths due to lack of ability to infer information beyond known areas. Moreover, HEADER's advantage over ARiADNE remains evident in high-fidelity simulations and



(a) Tunnel (b) Campus (c) Forest (d) Indoor

Fig. 6. Demonstration of the exploration trajectories output by HEADER in both indoor and outdoor Gazebo simulations. The Trajectory is color-mapped to represent the robot's movement over time.



(a) Tunnel (b) Campus (c) Forest (d) Indoor

Fig. 7. Performance comparison between HEADER and baseline methods across 10 runs per method in each environment. The first row shows the explored volume as a function of time, reflecting exploration efficiency. The second row presents the per-decision computation time, indicating computational cost during exploration. The third row illustrates the distribution of makespan values of all runs. Overall, HEADER achieves consistently higher exploration efficiency and lower computation time, and its makespan is statistically lower than that of all baselines.

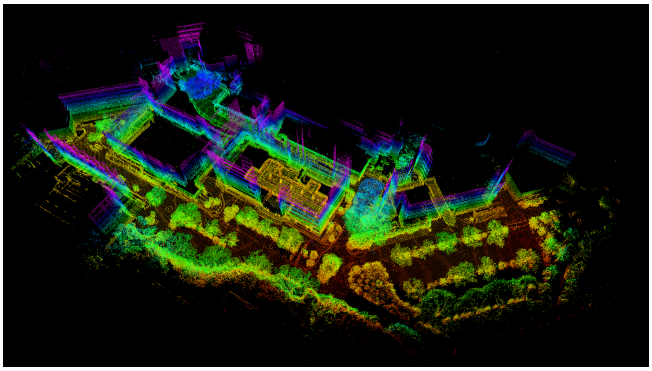
becomes even more significant in complex scenarios.

2) The community-based graph is a more efficient global representation: At the global level, **our community-based approach enables the construction of a sparser graph representation, offering improved scalability without compromising the coverage of explored areas**. The global partitioning depends on the structure of the constructed local graph, allowing the shape of each global partition to be adaptively and flexibly adjusted. Meanwhile, the computation is typically light since the complexity of our community partition is $\mathcal{O}(n+m)$, where n denotes the number of local nodes and m denotes the number of local edges.

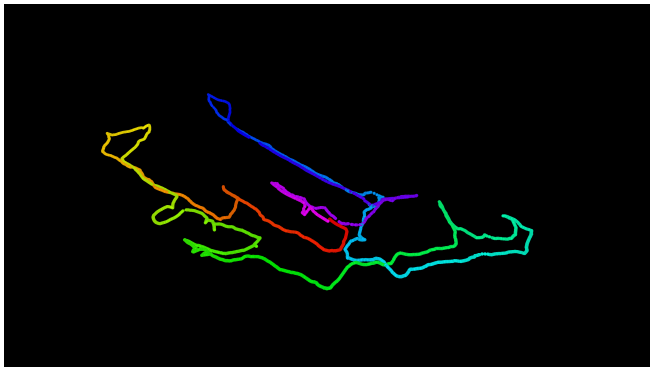
3) Joint decisions allow more adaptive exploration behaviors: Existing methods typically require the robot to strictly follow the planned global path, which is coarse and may be misleading in certain scenarios. In contrast, **HEADER treats the global path as a reference rather than a constraint, allowing the local planner to make joint decisions**. This enables HEADER to adaptively determine whether to follow the global guidance or deviate from it when necessary. As a result, HEADER demonstrates robustness in scenarios where the global path is suboptimal or potentially misleading for local exploration.

B. Hardware Validation

We validate HEADER on two wheeled robots: an Agilex Scout-mini equipped with an Ouster OS0-32 LiDAR, and a Yuhesen FW-mini equipped with two mid360 LiDARs. We use FastLIO2 [3] to get the odometry and mapping. We set the max speed to $1m/s$, sensor range to $8m$, and replanning frequency to $1Hz$ for all of our tests. We run HEADER in three real-world scenarios: a $200m \times 130m$ indoor teaching building environment, a $75m \times 60m$ outdoor garden environment, and a $300m \times 230m$ outdoor campus environment. We set the map resolution $\Delta_{map} = 0.4m$, the node resolution $\Delta_{node} = 0.8m$ for teaching building and garden environments, and $\Delta_{map} = 1m, \Delta_{node} = 2m$ for the campus environment. We note that these parameters were all guessed beforehand without tuning during the experiment. Since our implementation has not considered negative obstacles, we manually block stairs in the building and garden environments. In these real-world tests, HEADER reproduces similar performance as in the simulation: successfully exploring the full environment, keeping low computing time, and achieving high exploration efficiency. We believe this hardware validation further demonstrates HEADER's generalizability to unseen environments, as well as its robust sim-to-real transferability.

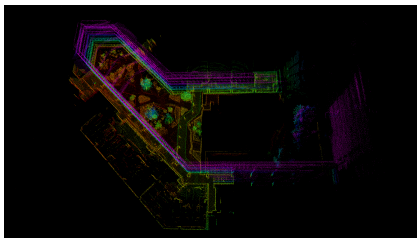


(a) Campus Explored Map

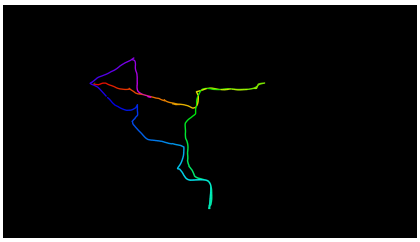


(b) Campus Exploration Trajectory

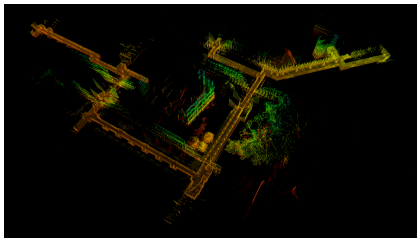
(c) Hardware setup



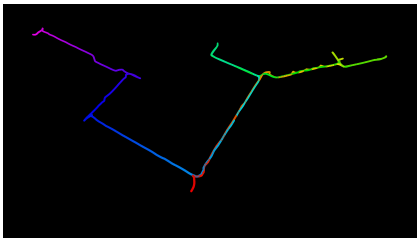
(d) Garden Map



(e) Garden Trajectory



(f) Building Map



(g) Building Trajectory

Fig. 8. Validations of HEADER in real-world scenarios with a wheeled robot. The trajectory is color-mapped to represent the robot's movement over time. The campus environment is around $300\,m \times 230\,m$, garden $75\,m \times 60\,m$, and building $200\,m \times 130\,m$.

V. Conclusion

In this work, we propose HEADER, a hierarchical learning-based planner for autonomous robot exploration. We first introduce an efficient global representation using community detection. We then develop an attention-based neural network that leverages both global reference paths and local observations to make joint decisions for local movements. Furthermore, we design a privileged expert reward to train the network, enabling the model to better reason about critical unknown areas by imitating a privileged expert while still exploring the policy space. As a result, HEADER achieves state-of-the-art performance in terms of scalability and exploration efficiency, as demonstrated in both high-fidelity simulations and real-world hardware experiments.

Future works will focus on extending HEADER from single to multi-robot exploration, where robots needs to achieve efficient cooperation in both local and global level. We are interested in extend HEADER to handle sensors with a limited filed-of-view (e.g., cameras), where the heading of the robot should be considered.

References

[1] J. Zhang, S. Singh *et al.*, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[2] 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 *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[3] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

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

[5] C. Cao, H. Zhu, H. Choset, and J. Zhang, "Tare: A hierarchical framework for efficiently exploring complex 3d environments," in *Robotics: Science and Systems*, 2021.

[6] 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.

[7] J. Huang, B. Zhou, Z. Fan, Y. Zhu, Y. Jie, L. Li, and H. Cheng, "Fael: Fast autonomous exploration for large-scale environments with a mobile robot," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1667–1674, 2023.

[8] O. Peltzer, A. Bouman, S.-K. Kim, R. Senanayake, J. Ott, H. Delecki, M. Sobue, M. Kochenderfer, M. Schwager, J. Burdick *et al.*, "Fig-op: Exploring large-scale unknown environments on a fixed time budget," *arXiv preprint arXiv:2203.06316*, 2022.

[9] S. Long, Y. Li, C. Wu, B. Xu, and W. Fan, "Hphs: Hierarchical planning based on hybrid frontier sampling for unknown environments exploration," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 056–12 063.

[10] C. Cao, H. Zhu, Z. Ren, H. Choset, and J. Zhang, "Representation granularity enables time-efficient autonomous exploration in large, complex worlds," *Science Robotics*, vol. 8, no. 80, p. eadf0970, 2023.

[11] 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 *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7623–7630.

[12] F. Chen, S. Bai, T. Shan, and B. Englot, "Self-learning exploration and mapping for mobile robots via deep reinforcement learning," in *Aiaa scitech 2019 forum*, 2019, p. 0396.

[13] F. Chen, J. D. Martin, Y. Huang, J. Wang, and B. Englot, "Autonomous exploration under uncertainty via deep reinforcement learning on graphs," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6140–6147.

[14] Y. Xu, J. Yu, J. Tang, J. Qiu, J. Wang, Y. Shen, Y. Wang, and H. Yang, "Explore-bench: Data sets, metrics and evaluations for frontier-based and deep-reinforcement-learning-based autonomous exploration," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6225–6231.

[15] S. Zhu, J. Zhou, A. Chen, M. Bai, J. Chen, and J. Xu, "Maexp: A generic platform for rl-based multi-agent exploration," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5155–5161.

[16] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, 2019.

[17] Y. Cao, T. Hou, Y. Wang, X. Yi, and G. Sartoretti, "Ariadne: A reinforcement learning approach using attention-based deep networks for exploration," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 219–10 225.

[18] Y. Cao, R. Zhao, Y. Wang, B. Xiang, and G. Sartoretti, "Deep reinforcement learning-based large-scale robot exploration," *IEEE Robotics and Automation Letters*, 2024.

[19] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[20] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[22] L. Perron and V. Furnon, "Or-tools," Google. [Online]. Available: https://developers.google.com/optimization/

[23] E. DIJKSTRA, "A note on two problems in connexion with graphs." *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[24] C. Cao, H. Zhu, F. Yang, Y. Xia, H. Choset, J. Oh, and J. Zhang, "Autonomous exploration development environment and the planning algorithms," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8921–8928.

[25] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.