

Deep Reinforcement Learning-Based Large-Scale Robot Exploration

Yuhong Cao^{ID}, *Graduate Student Member, IEEE*, Rui Zhao^{ID}, Yizhuo Wang^{ID}, Bairan Xiang^{ID}, and Guillaume Sartoretti^{ID}, *Member, IEEE*

Abstract—In this work, we propose a deep reinforcement learning (DRL) based reactive planner to solve large-scale Lidar-based autonomous robot exploration problems in 2D action space. Our DRL-based planner allows the agent to reactively plan its exploration path by making implicit predictions about unknown areas, based on a learned estimation of the underlying transition model of the environment. To this end, our approach relies on learned attention mechanisms for their powerful ability to capture long-term dependencies at different spatial scales to reason about the robot’s entire belief over known areas. Our approach relies on ground truth information (i.e., privileged learning) to guide the environment estimation during training, as well as on a graph rarefaction algorithm, which allows models trained in small-scale environments to scale to large-scale ones. Simulation results show that our model exhibits better exploration efficiency (12% in path length, 6% in makespan) and lower planning time (60%) than the state-of-the-art planners in a 130 m × 100 m benchmark scenario. We also validate our learned model on hardware.

Index Terms—View Planning for SLAM, reinforcement learning, motion and path planning.

I. INTRODUCTION

AUTONOMOUS exploration focuses on finding the shortest total exploration path to map an unknown environment (i.e., classify it into free and occupied areas). In this work, we consider autonomous exploration based on omnidirectional 3D Lidar with 2D action space (i.e., for a ground robot). There, recent developments of Lidar-based simultaneous localization and mapping (SLAM) can now near-perfectly mitigate state estimation error and yield high-quality maps on hundreds-meter scenarios [1]. Leveraging these advanced SLAM approaches, current autonomous exploration planners [2], [3], [4], [5], [6] are able to assume the localization and mapping are perfect and start tackling complex larger-scale environments towards real-life deployments. However, optimizing trajectories in large-scale environments is non-trivial, since the exploration task requires

Manuscript received 25 October 2023; accepted 29 February 2024. Date of publication 20 March 2024; date of current version 8 April 2024. This letter was recommended for publication by Associate Editor A. M. Metelli and Editor J. Kober upon evaluation of the reviewers’ comments. This work was supported in part by Temasek Laboratories (TL@NUS) under Grant TL/FS/2022/01, and in part by the Singapore Ministry of Education Academic Research Fund Tier 1. (*Corresponding author: Yuhong Cao.*)

The authors are with the Department of Mechanical Engineering, College of Design and Engineering, National University of Singapore, Singapore 117482 (e-mail: caoyuhong@u.nus.edu).

This letter has supplementary downloadable material available at <https://doi.org/10.1109/LRA.2024.3379804>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3379804

real-time replanning as the robot discovers unknown areas and updates its map, with planning horizons in the hundreds of meters.

Advanced large-scale planners often leverage hierarchical planning to decrease computational complexity while maintaining the fineness of local paths [2], [3], [4], [6], as a result, the state-of-the-art planner, TARE [4], is able to explore environments at the hundred-meters scale while keeping computing times under a second at each planning step. Nevertheless, these conventional planners share the same limitation: they try to optimize paths solely based on the partial belief (map) over the environment. However, even optimal long-term paths on the partial map can lead to suboptimal longer-term behaviors as the map is updated with new measurements that may contradict the current plan. At its core, the exploration problem is a partially observable Markov decision process (POMDP), where the transition model between true states (ground truth of the environment) and the robot belief (map) is hard to predict. For example, a robot cannot confirm that two close-by, incomplete corridors in its partial map are connected before fully exploring this portion of the environment. However, we note that humans in the same situations will often try to infer the complete structure of an environment from a partial map (i.e., guess and reconstruct that full corridor) to improve efficiency. Learning-based approaches such as supervised learning or model-based reinforcement learning may be an option to explicitly predict unknown areas from the agent’s partial map, allowing conventional planners to then plan paths over the predicted belief. However, since the underlying transition model in autonomous exploration is highly stochastic and thus very hard to estimate/learn, directly relying on such a learned prediction may not be a wise choice in practice.

We believe that a simple model-free deep reinforcement learning (DRL) approach can elegantly handle all these challenges, by implicitly estimating the transition model and learning an adaptive policy. We build upon recent attention-based policy neural network [7], [8], which have shown outstanding abilities at allowing the agent to reason about its entire belief at multiple spatial scales. In this work, we further propose to train an attention-based policy with the assistance of ground truth knowledge (i.e., privileged learning). It enables the model to draw out more potential of attention mechanism at implicitly predicting unknown areas from the partial robot belief, leading to a more stable training process and better performance after training. In particular, we train our model under the soft actor-critic (SAC) algorithm [9], [10], and let the critic network have access to

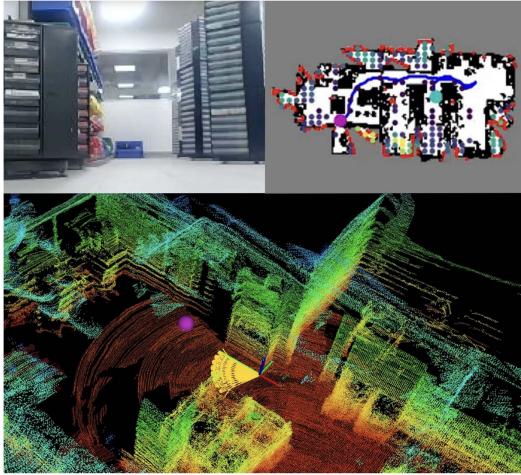


Fig. 1. Mobile ground robot exploring an indoor lab environment using our DRL-based planner. Top-left: view from the robot's onboard camera (*not used for mapping*). Top-right: agent's current belief (map) of the environment, and view of our neural network's inputs. Bottom: 3D point cloud (map) of the environment constructed by the robot. The axes system represents the current position and orientation of the robot, the purple ball the next waypoint output by our planner.

the ground truth of the environment to get precise long-term evaluations. We further propose a graph rarefaction algorithm, which enhances the model's capacity to capture longer-term temporal dependencies between areas, thus extending the use of trained policy from small- to large-scale environments. We compare our DRL-based planner with conventional planners in simplified small-scale environments, where it achieves 11% better exploration efficiency in terms of path length. We then test our planner in a high-fidelity 3D Gazebo simulation benchmark of a $130\text{ m} \times 100\text{ m}$ indoor office, where it achieves better exploration efficiency (12% in path length, 6% in makespan) and significantly lower computing times (60%) compared to the state-of-the-art exploration planner: TARE [4], [6]. Finally, we experimentally validate our planner on a ground robot in an $80\text{ m} \times 10\text{ m}$ indoor scenario involving cluttered furniture, highlighting its real-life applicability without any additional training (demonstrated in Fig. 1). To the best of our knowledge, our work is the first DRL planner that can effectively explore such large-scale environments.

II. RELATED WORKS

In this work, we focus on methods applicable to the problem setup where the information in the robot belief is evaluated through frontiers (i.e., boundary between known, free area and unknown area). There are another group of other methods based on the problem setup where the information is evaluated through information theory (e.g., mutual information) [11], [12], [13]. However, information theory-based methods require the robot belief to be a probabilistic occupancy map, making them not applicable in the setup of this work and related works discussed below.

Small-scale exploration planner: Frontier-based exploration planners have been shown to be very efficient in small-scale environments. Yamauchi et al. [14] first proposed to utilize

frontiers to drive exploration, where the *cost* (i.e., the path length from the current position to a frontier) is usually used to rank frontiers, with the lowest-costed frontier being chosen as the next one to be visited by the robot. More recent works [15], [16] further considered *utility* (i.e., the size of a frontier, quantifying the amount of information expected to be collected at that frontier) to improve exploration efficiency. Although the resulting paths are often short-sighted (i.e., optimize for shorter-term objectives, often at the cost of longer-term exploration ones), the above planners remain near-optimal in simple scenarios, where the number of frontiers is relatively low. In general, as the number of frontiers increases in larger and more cluttered environments, frontier-based planners suffer from such *myopic* decisions. Instead of evaluating a single frontier, [17] relied on a sampling-based strategy in a receding-horizon manner to optimize a long-term trajectory, thus achieving non-myopic decisions and significantly outperforming frontier-based planners in relatively complex (but still small-scale) environments while remaining executable in real-time.

Large-scale exploration planner: Optimizing long-term trajectories online in large-scale exploration problems is challenging since trajectories might be over 100 m in length, while the gap between individual waypoints should remain below 1 m to allow for fine detail mapping. Some advanced sampling strategies have been proposed to tackle larger-scale exploration [18], [19]. Selin et al. [2] found that, in large-scale environments, frontiers are sparse at the global scale but dense at the local level, and proposed a hierarchical framework that combined a global frontier-based planner and a local planner based on [17]. Following works [3], [4], [6], [20] drew on the experience of such hierarchical frameworks to perform path planning at multiple resolutions and further improve the exploration efficiency of planned paths, which significantly outperformed naive frontier- and sampling-based planners [3], [4], [6]. Notably, Cao et al. [4], [6] proposed to use a TSP-based global planner and a sampling-based local planner with a coverage constraint to trade off the planning fineness of nearby and distant areas, which currently exhibits state-of-the-art performance with 80% better exploration efficiency over other benchmark planners [3], [17], [21].

DRL-based exploration planner: While some works [22], [23] have relied on deep learning to explicitly predict the layout of the unknown (but only structured indoor) environments to assist conventional approaches, most learning-based approaches leverage deep reinforcement learning (DRL) to directly and reactively make decisions on movements. DRL-based planners are expected to achieve long-term exploration objectives by training a policy to maximize long-term *returns*. Most current DRL-based exploration planners [24], [25], [26], [27] rely on convolutional neural networks (CNNs) to select a waypoint from a set of locations, based on a visual representation of the agent's belief (e.g., map). Although DRL intuitively looks like a well-suited method for autonomous exploration, the performance of these DRL-based planners is only on par with naive frontier-based planners in small-scale environments. On the other hand, Chen et al. [28] proposed to utilize graph neural network to handle the localization error during exploration, however their approach can only work in environments without obstacles.

Notably, relying on learned attention over a graph representation of the agent's belief, our previous work ARIADNE [8] achieved remarkable improvements over frontier-based planners and one of these CNN-based DRL planners [25]. However, ARIADNE's advantage over TARE is marginal, and like other previous DRL methods, this DRL planner cannot scale to large environments.

III. PROBLEM STATEMENT

Let \mathcal{E} be a bounded environment, classified into free areas \mathcal{E}_f and occupied areas \mathcal{E}_o , where $\mathcal{E}_f \cup \mathcal{E}_o = \mathcal{E}$ and $\mathcal{E}_f \cap \mathcal{E}_o = \emptyset$. The robot maintains a map \mathcal{M} (i.e., robot belief) about \mathcal{E} , which is classified into free areas \mathcal{M}_f , occupied areas \mathcal{M}_o , and unknown areas \mathcal{M}_u , where $\mathcal{M}_f \cup \mathcal{M}_o \cup \mathcal{M}_u = \mathcal{M}$. Starting from an empty belief over the environment (i.e., $\mathcal{M} = \mathcal{M}_u$), the robot executes an exploration path ψ and updates its map/belief about the environment using measurements taken along the path from an onboard sensor (here, a LiDAR). The exploration task completes when \mathcal{M}_o is closed with respect to \mathcal{M}_f ,¹ and the objective is to minimize the cost $C(\psi)$ (e.g., the path length $L(\psi)$ or makespan $T(\psi)$) of the exploration path.

Considering the completed exploration path as a finite sequence of robot-visited positions $\psi = [p_1, p_2, \dots, p_n]$, where $p_i \in \mathcal{E}_f$ is the robot position at decision step i , the exploration problem is a partially observable Markov decision process (POMDP). This problem can be formulated as a tuple (S, A, T, R, Ω, O) , where S is a set of states, A is a set of actions, T is a set of conditional probabilities between states, R is a set of reward, Ω is a set of observations, O is a set of conditional probabilities between states and observations. The true state $s_i = (\mathcal{E}_i, \mathcal{M}_i, \psi_{1:i})$, $s_i \in S$ is not available to the agent during exploration; instead, the robot only has access to observation (i.e., its map/belief) $o_i = (\mathcal{M}_i, \psi_{1:i})$, $o_i \in \Omega$, depending on $O(o_i|s_i)$. At each decision step, the robot selects and takes action $a_i \sim \pi(\cdot|o_i)$, which indicates the next waypoint to visit. Upon reaching that next state s_{i+1} with $T(s_{i+1}|s_i, a_i)$, the agent receives a reward $r_i = -C(\psi_{t-1:t})$. The objective is to find an optimal policy π^* , which selects an action at each decision step that maximizes the long-term discounted return $\mathbb{E}(\sum_{t=i}^n r_t)$.

IV. METHODOLOGY

Solving a highly stochastic POMDP such as autonomous exploration is non-trivial, and thus we propose to rely on model-free DRL for its powerful ability to make implicit predictions based on past experiences. In this work, we rely on an attention-based neural network similar to [7], [8] for its proven ability to model multi-scale dependencies between areas. We further improve the model's performance by allowing the critic to train based on ground truth information, to help it precisely estimate returns and the underlying transition model. Guided by these improved predictions, our policy network can finally improve its ability to achieve long-term efficiency under partial observations.

¹In practice, the map is often considered complete when all frontiers have been removed, which also ensures the set of occupied areas is closed.

We first decrease the complexity of the problem by extracting a dense collision-free graph from the map. With this collision-free graph as input, we train an attention-based policy network offline, using the soft actor-critic (SAC) with discrete actions [9], [10]. Note that under the actor-critic framework, the critic network is only used during training to assist the policy network in estimating the state-action value. Therefore, we allow the critic network to access true states s , instead of partial observations o , to estimate the long-term impact of current actions more precisely, which in turn leads to improved overall performances. Moreover, benefiting from the nature of attention layers [29], [30], our policy network is able to generalize to arbitrary graphs and arbitrary scale environments. We further propose an algorithm to rarefy the dense collision-free graph, allowing our DRL-based planner to naturally scale to larger-scale environments without any additional training.

A. Sequential Decision-Making on a Graph

Consider the collision-free graph $G = (V, E)$ extracted from the map \mathcal{M} , where $V = (v_1, v_2, \dots, v_m)$, $\forall v_i = (x_i, y_i) \in \mathcal{M}_f$ is a set of nodes (one of the nodes is located at the robot current position p_t) and $E = ((v_1, v_2), \dots, (v_{m-1}, v_m))$ is a set of collision-free edges. In this work, nodes in V uniformly cover free areas \mathcal{M}_f . We find the k nearest neighbors of each node and check for collisions to construct the set of admissible edges E (i.e., edges between nodes that do not cross \mathcal{M}_o or \mathcal{M}_u). During exploration, this collision graph extends incrementally along with the update of free areas. Our DRL planner only makes decisions on which neighboring node will be the next waypoint (i.e., $\pi(a_i|o_t) = p(v_i|o_t)$, $(p_t, v_i) \in E$), toward which the robot then navigates. Note that, during training, the planner only makes decisions upon arriving at the previously selected waypoint, while at execution time, the planner can make decisions at a fixed frequency to be more reactive to map updates.

B. Policy Network

We first extract an *informative graph* G^* , extending the collision graph G , as the input of our policy network to allow for more efficient learning. The informative graph $G^* = (V^*, E)$ shares the same collision-free edge set as G , and each node $v_i^* = (v_i, u_i, g_i) \in V^*$ has two more properties than nodes in V : 1) the node's *utility* (denoted as u_i , quantifying observable frontiers within line of sight from location v_i) and 2) the *guidepost* (denoted as g_i , a binary value defining whether v_i has already been visited previously). The utility extracts information from the map to avoid the need for unnecessary learning of low-level pattern recognition, allowing the network to focus on modeling long-term dependencies between areas. The guidepost representing the path $\psi_{1:t}$ executed so far, which we empirically found to help speed up training by encouraging the robot to select unvisited nodes. The informative graph is normalized and used as input of our policy network, composed of an encoder and a decoder, as shown in Fig. 2, where the encoder learns to model the dependencies among nodes V^* and the decoder uses these dependencies to finally output the policy π over neighboring nodes.

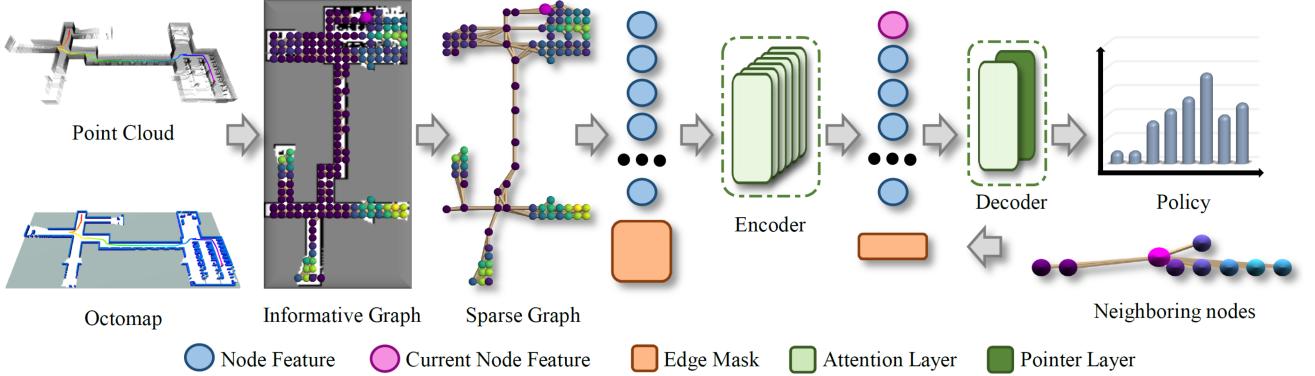


Fig. 2. Our proposed DRL-based planner. The robot first transforms its current map (point cloud) data to an occupancy grid, and then extracts and rarefies an informative graph from it. After that, this graph (node features and an adjacency edge mask) is fed to our attention-based network (consisting of an encoder and a decoder), which finally outputs a policy over which neighboring node should be the next waypoint.

Encoder: A graph structure is naturally aligned with the input format of an attention layer:

$$q_i = W^Q h_i^q, k_i = W^K h_i^{k,v}, v_i = W^V h_i^{k,v}, u_{ij} = \frac{q_i^T \cdot k_j}{\sqrt{d}},$$

$$w_{ij} = \begin{cases} \frac{e^{u_{ij}}}{\sum_{j=1}^n e^{u_{ij}}}, & M_{ij} = 0 \\ 0, & M_{ij} = 1 \end{cases}, h'_i = \sum_{j=1}^n w_{ij} v_j, 1 \leq i \leq m \quad (1)$$

where each $h_i \in \mathbb{R}^{d \times 1}$ is a d -dimension feature vector projected from node v_i^* , superscripts q and k, v denote the source of the query, key, and value respectively, $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$ are learnable matrices, and M , which serves as an edge mask, is a $m \times m$ adjacency matrix built from E ($M_{ij} = 0$, if $(i, j) \in E$, else $M_{ij} = 1$). Note that in each attention layer, each node in the informative graph is only allowed to access the features of its neighboring nodes. To allow the model to learn dependencies between non-neighboring nodes, we stack six attention layers in the encoder, each taking the output of its previous attention layer as input. The output of the encoder is a group of *node features*, each h'_i modeling dependencies between node v_i^* and all other nodes.

Decoder: Denoting the node feature corresponding to the robot's current position p_t as h^c , we first add a feature vector representing the global graph to it, by passing h^c as the query source and all node features h'_i as the key and value source to an attention layer, concatenating its output with h^c and projecting its back to a d -dimension feature h^{*c} . We then select its neighboring node features $h_i^n, \forall (p_t, v_i) \in E$ and input them as the key source and h^{*c} as the query source to a pointer layer [7], [8], [31], which directly outputs attention weights w as the policy $\pi(a_i|o_t) = p(v_i|o_t), (p_t, v_i) \in E$. By using this pointer network at the end of the decoder, the final policy's dimensions naturally match the number of neighboring nodes.

C. Training With Ground Truth

SAC: We train our attention-based policy network using the soft actor-critic (SAC) algorithm with discrete actions [9], [10], in the set of small-scale exploration environments provided by [25]. The goal of SAC is to balance the trade-off between

maximizing returns and policy entropy:

$$\pi^* = \operatorname{argmax} \mathbb{E} \left[\sum_{t=0}^T \gamma^t (r_t + \alpha \mathcal{H}(\pi(\cdot|o_t))) \right], \quad (2)$$

where \mathcal{H} denotes the entropy and α is a temperature parameter that tunes the importance of the entropy term versus the return. We use reward shaping to help tune the training: for each action a_t , in addition to the cost reward $r_c = -C(\psi_{t-1:t})$, we give an exploration reward r_e that quantifies the number of observed frontiers associated with a_t . Upon finishing exploration (i.e., once the utility of all nodes is zero), we also give the agent a fixed finishing reward r_f . Thus the reward used for training is $r_i = a \cdot r_c + b \cdot r_e + r_f$, where a and b are scaling parameters (in practice $a = 1/64$, $b = 1/50$, $r_f = 20$). SAC trains a critic network ϕ to estimate the soft action-state values $Q_\phi(o_t, a_i)$, using the critic loss: $J_Q(\phi) = \mathbb{E}_{o_t} [\frac{1}{2} (Q_\phi(o_t, a_t) - (r_t + \gamma \mathbb{E}_{o_{t+1}} [V(o_{t+1})]))^2]$, where $V(o_t) = \mathbb{E}_{a_i} [Q(o_t, a_i)] - \alpha \log(\pi(\cdot|o_t))$. The policy network θ is trained to output a policy that maximizes the expected state-action value, where the policy loss reads: $J_\pi(\theta) = \mathbb{E}_{(o_t, a_i)} [\alpha \log(\pi_\theta(a_i|o_t)) - Q_\phi(o_t, a_i)]$. The temperature parameter is auto-tuned during training and the temperature loss is calculated as: $J(\alpha) = \mathbb{E}_{a_t} [-\alpha (\log \pi_t(a_t|o_t) + \bar{\mathcal{H}})]$, where $\bar{\mathcal{H}}$ denotes the target entropy [9], [10].

Critic Network: The training of a model-free DRL agent on a POMDP involves the need to estimate state/state-action values from partial observations of the environment, where the model needs to implicitly learn to predict the transition model T and O , which is non-trivial in a stochastic problem such as autonomous exploration. In SAC, the learning of the policy network depends on accurate state-action value estimations from the critic network, where such coupling might further hamper the learning of optimal policies (e.g., from oscillations in the critic's training from high-variance gradients). In this context, recent works [8] tried to use standard SAC (i.e., identical observation inputs and nearly the same network structure for both actor and critic with just a different final layer to differentiate their output) but found that the critic network remained rather noisy due to the high randomness of the underlying transition model (shown as the

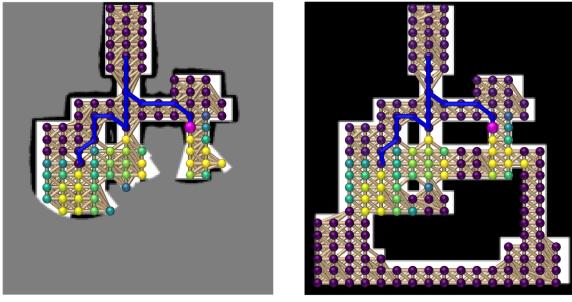


Fig. 3. Informative graph (left) and Ground-truth graph (right). The informative graph is the input of our policy network. The ground truth graph is the input of our critic network, which is only used during training to assist the learning of the policy. Nodes are color-coded based on their utility (dark purple to yellow, low to high). The blue trajectory is the path executed so far by the robot (light purple node).

gradient variance). During training, the critic loss remains high ($\sim 10\%$ of $Q(o_t, a_t)$), showing that the critic is struggling at learning the underlying long-term effects of actions/states, and thus providing less accurate estimations to train the actor.

To address this issue, in this work, we leverage the structure of the actor-critic framework, where the critic network is only used to assist policy network training and is not required after being trained, allowing us to take true states as the observations for the critic network (i.e., privileged learning). Fig. 3 shows a visual comparison between the inputs of the policy network and the critic network. Specifically, we keep the structure of the vanilla critic network but construct a *ground truth graph* $G' = (V', E')$, from s_t and o_t . G' is similar to the informative graph used as input of the policy network, but the node set V' covers the whole (ground truth) free area in \mathcal{E} , the property of each node $v'_i = (x_i, y_i, u_i, e_i)$ is slightly different: the unexplored feature e_i is a binary value indicating whether v'_i is in the already explored area by the robot. If a node v'_i lies in unexplored areas, we set its utility $u_i = -1$. By doing so, the critic network now only needs to tackle a more traditional MDP problem, i.e., estimating paths to cover an environment with fully known prior, allowing for more accurate estimations of state-action values (around 10 times lower state-action value loss and gradient variance). In practice, this translates into $\sim 10\%$ increased average per-step rewards compared to standard SAC (more details can be found in the supplemental material).

Training details: Our model is trained on the dungeon environments provided by [25], where each environment \mathcal{E} is a 640×480 grid world. The sensor range of our robot is set to 80 cells. We uniformly place 30×30 points in the environment and select all points in the explored free area \mathcal{M}_f to form the node set V and construct the information graph $G^* = (V^*, E^*)$, where the number of neighboring nodes $k = 20$. The exploration task is completed once there is no non-zero utility node in V^* (using a threshold to ignore nodes with $u_i \leq 5$ in practice). During training, we set the max episode length to 128 decision steps, the discount factor to $\gamma = 1$, the batch size to 64, and the episode buffer size to 2500. Training starts after the episode buffer collects more than 1000 step data. The target entropy is set to $0.01 \cdot \log(k)$. Training happens at the end

of each episode, and is composed of 8 iterations. We use the Adam optimizer with a learning rate of 10^{-5} for both policy and critic networks and 10^{-4} for the temperature auto-tuning. The target critic network updates every 64 training step. Our model is trained on a desktop equipped with an AMD Ryzen7 5700X CPU and an NVIDIA GeForce RTX 4080 GPU. We train our model utilizing Ray, a distributed framework for machine learning [32], and run 16 training environments simultaneously to accelerate data collection. The training needs approximately 3 days to fully converge. Our full code can be found at <https://github.com/marmotlab/large-scale-DRL-exploration>.

D. Graph Rarefaction for Large-Scale Exploration

Although our policy network is trained in small-scale environments only, our networks can naturally handle arbitrary graphs. However, the number of nodes needed to let G^* uniformly cover larger-scale environments will be significantly higher, even though while the graph structure would be dense, information would remain sparse (i.e., many frontiers are very far from each other, and separated by a large number of 0-utility nodes). To extend the use of our trained model to larger-scale environments, we propose a method to rarefy our informative graph, which lets the model capture longer-term dependencies more efficiently (i.e., frontiers far away can connect with each other through fewer edges). In short, we first classify all nodes with non-zero utility into multi groups according to neighboring relationship, and then find the shortest paths on G from the robot's position to each group through A* and check for line of sight at each waypoint in these paths, to find a minimal set of nodes $V^s \subset V^*$ that represents all information in the map (the pseudo-code can be found in the supplemental material). The result is a sparser information graph $G^s = (V^s, E^s)$, which is used as the policy network's input.

V. EXPERIMENT

A. Validation in Small-Scale Environments

We first test our trained model on a benchmark set of 100 small-scale simplified simulation environments (never seen during training). We compare our planner with baseline planners including 1) *nearest*: the robot always moves to the nearest frontier, 2) *utility*: the robot moves to the frontier which balances cost and utility, 3) *NBVP*: a sampling-based planner based on rapid random trees [17], and 4) *TARE Local*: the local level planner of TARE [4], [6]. 5) CNN: a DRL planner based on CNNs [25]. 6) ARiADNE: an ablation variant of our model trained without ground truth as in [8]).

In our tests, we consider exploring more than 99% (meaning minor error is tolerable) of the free area in the environment as completing the exploration. As shown in Table I, our DRL planner outperforms all baseline planners (11% better than TARE Local and Utility, 15% better than CNN and NBVP) in terms of average travel distance to complete the exploration. We first note that the vanilla model trained without ground truth already achieves shorter exploration paths than all other baseline planners, showing the efficiency of our attention-based policy

TABLE I
COMPARISONS WITH BASELINE PLANNERS IN SMALL-SCALE ENVIRONMENTS (IDENTICAL 100 SCENARIOS FOR EACH METHOD)

	Nearest	Utility	NBVP	TARE Local	CNN	ARiANDE	Ours
Distance (pixel)	1354(± 410)	1268(± 396)	1323(± 371)	1266(± 388)	1323(± 428)	1204(± 378)	1118(± 321)

The best value in each row is indicated in bold.

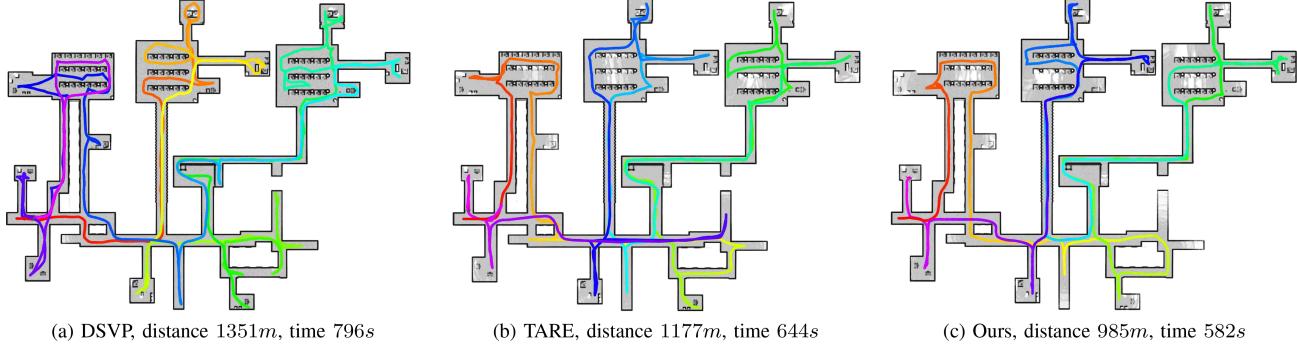


Fig. 4. Exploration paths comparisons in a large-scale 130 m \times 100 m indoor office simulation.

network to reason about dependencies of areas at multiple scales and thus making decisions that balance exploration and refining the map. We then note that our model trained with ground truth further improves over this vanilla ARiADNE model (7% better), verifying that training the critic network with ground truth can assist the policy network to estimate the transition model and long-term returns more precisely. Our results suggest that our planner can effectively predict the structure of unexplored areas, to make further non-myopic decisions and avoid redundant movements, by relying on past experiences in similarly structured indoor environments seen at training. We visualize attention weights of our attention layer to observe the insight learned mechanism (see our supplemental material), and we find that in some heads the planner focuses its attention over areas where connections might be missing, implying that the planner is predicting the structure of unknown areas from the belief of known areas.

B. Validation in Large-Scale Environments

We then test our large-scale exploration planner in a Gazebo simulation of a 130 m \times 100 m office with cluttered obstacles provided by [6]. Compared to our small-scale test environments where the sensor model and robot kinetic are highly simplified, in this large-scale test environment, the simulation considers real-world sensor model (a Velodyne 16-line 3D LiDAR) and real robot constraint (a four-wheel differential drive robot with max speed 2 m/s). Note that the model used for our large-scale tests is the same as the one for our small-scale tests, but using our graph rarefaction algorithm. We use Octomap [33] to classify free and occupied areas (i.e., define “explored” area). We set the resolution of Octomap to 0.4 m, the max map update range to 20 m, the gap of nodes to 2 m, the number of neighboring nodes in the dense collision-free graph to 5, the number of neighboring nodes in the sparse information graph to 10, the sparse radius to 12 m, and replan the robot’s path every 0.8 s. Although we

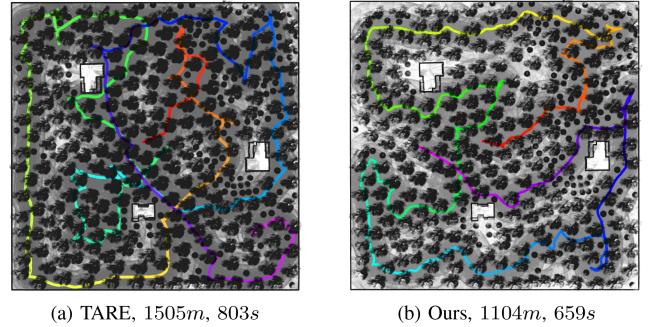


Fig. 5. Exploration paths comparisons in a large-scale 150 m \times 150 m outdoor forest simulation.

construct a 3D Octomap, the planner only considers frontiers at the ground level (i.e., planning is based on a 2D belief). All computations are conducted on a Intel i7-1270p CPU.

We compare our large-exploration planner with: 1) TARE [4], [6] (also the provider of this test environment), the state-of-the-art large-scale exploration planner which in general produces 80% increased exploration efficiency over benchmark planners such as [3], [17]; in TARE, a global planner is designed to find a global TSP path to visit all unexplored areas, and the start of the global path is taken as the destination for the local path planner discussed in the last subsection, 2) DSVP [34], which uses a rapid random tree-based local planner and a graph-based global planner, 3) Best human practice with full prior knowledge about the environment provided by [6] as the reference for optimal exploration path. The hyperparameters of TARE and DSVP were tuned by the original authors directly on benchmark environments. Note that TARE further considers 3D frontiers but still does path planning in 2D action space. We evaluate the performance of all planners from multiple metrics: 1) *Distance*: total exploration path length, 2) *Time*: total duration of the exploration task, 3) *Computing*: per-step planning time, and 4)

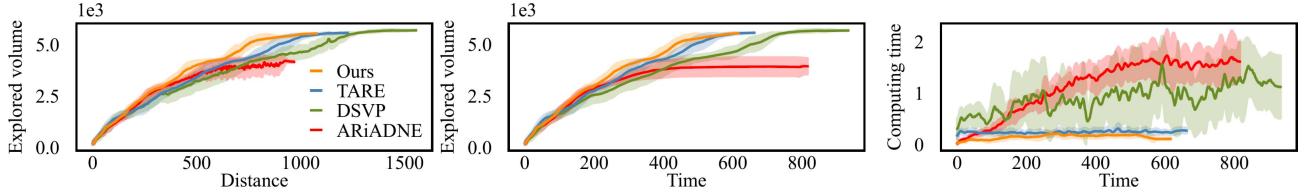


Fig. 6. Comparisons with baseline planners in the large-scale indoor benchmark [6] (10 runs each), including ARiADNE despite its inability to complete exploration.

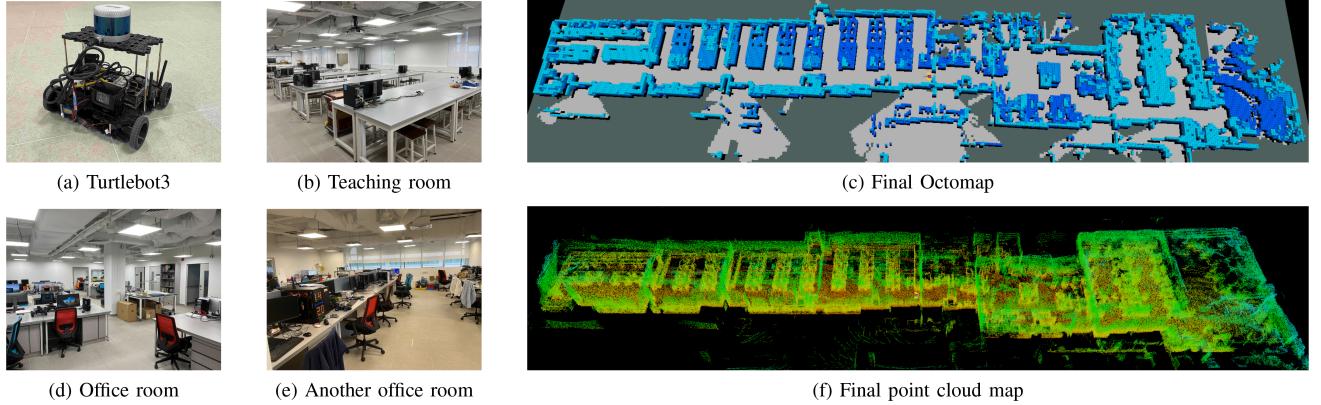


Fig. 7. Real world experiment in a $80 \text{ m} \times 10 \text{ m}$ laboratory with cluttered furniture and moving pedestrians.

TABLE II
COMPARISONS WITH BASELINE PLANNERS IN THE LARGE-SCALE INDOOR BENCHMARK [6] (10 RUNS EACH)

	DSVP	TARE	Ours	Human
Distance (m)	1462	1158	1020	879
Time (s)	870	634	590	520
Computing (s)	0.90	0.24	0.15	/
Efficiency (m^3/m)	3.91	4.82	5.42	6.27
Efficiency (m^3/s)	6.57	8.81	9.36	10.60

Efficiency: explored volume divided by the travel distance or makespan. Results are shown in Table II, Figs. 4 and 6.

We note that our DRL-based planner outperforms TARE in terms of exploration efficiency (12% in terms of distance efficiency and 6% in terms of time efficiency) and algorithm computing time (including frontier detection, informative graph update, and network inference time, 60% faster even though our planner is implemented in Python and TARE in C++). As shown in Fig. 4, the exploration path planned by our DRL planner exhibits fewer redundant movements than benchmark planners. Although TARE theoretically guarantees near-optimal paths to cover frontiers in its current belief, we believe the advanced performance of our DRL planners shows the key importance of predicting the potential structure of unknown areas, which can yield decisions that benefit longer-term exploration. Note that paths planned by TARE allow the robot to move at the fastest speed, as TARE explicitly considers the motion constraint of the robot. Therefore, our advantage in terms of time efficiency is smaller than distance efficiency.

Although our model was trained in indoor environments, we also test it in a $150 \text{ m} \times 150 \text{ m}$ cluttered outdoor forest Gazebo environment without further training (results are shown

TABLE III
COMPARISONS WITH BASELINE PLANNERS IN THE LARGE-SCALE OUTDOOR BENCHMARK [6] (5 RUNS EACH)

	DSVP	TARE	Ours	Human
Distance (m)	2002	1347	1174	1071
Time (s)	1053	707	686	549
Computing (s)	1.31	0.70	0.44	/
Efficiency (m^3/m)	21.13	29.37	32.63	36.05
Efficiency (m^3/s)	40.17	55.96	55.85	70.33

in Table III and Fig. 5). In this outdoor environment, the structure learned by our model (e.g., junctions and corridors) in indoor environments does not exist anymore, and we expect our planner not to be able to perform implicit predictions as well as in indoor environments. As expected, we observe our planner sometimes randomly chooses waypoints, implying that it cannot decide which action is optimal. However, in this cluttered outdoor environment, our DRL planner still achieves time efficiency on par with TARE and better distance efficiency.

C. Validation in Real World

We finally validate our large-scale planner in the real world, using a wheeled robot equipped with a 3D LiDAR to explore an $80 \text{ m} \times 10 \text{ m}$ indoor laboratory with cluttered furniture and moving pedestrians (see Fig. 7). The robot platform is a customized four-wheel Turtlebot3 with a max speed of 0.2 m/s . We use a Leishen C16 LiDAR with LOAM [1] to both get LiDAR odometry and mapping. For our DRL planner, compared to the parameters used in simulation, we set the resolution of Octomap to 0.2 m , and the gap of nodes to 0.8 m to better handle such a cluttered real-world indoor environment. We believe

the robot successfully and efficiently explores the lab in 12 minutes, highlighting the robust applicability of our planner in the real world. It should be noted that most previous DRL-based planners [24], [25], [26], [27], [28] are not validated on hardware in such a cluttered and large-scale environment. Compared to these planners that directly take the map as network input, where images from the real world may be very noisy and different from those used during training, we believe that extracting an informative graph helps the learned model avoid failure on never-seen real-world scenarios.

VI. CONCLUSION

In this work, we propose a DRL-based exploration framework, which relies on ground truth information to assist the training of an attention-based policy network. We show that our trained model can more precisely estimate the unexplored areas, which helps the planner better reason about its map/belief and make decisions that benefit long-term objectives. With a graph-rarefaction algorithm, our planner becomes the first DRL-based method applicable to large environments and significantly outperforms state-of-the-art conventional planners in a benchmark indoor simulation environment. We also validate our planner in the real world, highlighting its potential for real-life deployments on robot.

Future works will focus on extending our planner from single to multi-robot exploration, where the planner needs to further help robots achieve efficient cooperation. We are also interested in letting the robot take account of the robot's motion constraint in training.

REFERENCES

- [1] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst.*, 2014, pp. 1–9.
- [2] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3-D environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1699–1706, Apr. 2019.
- [3] 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.
- [4] C. Cao, H. Zhu, H. Choset, and J. Zhang, "TARE: A hierarchical framework for efficiently exploring complex 3D environments," in *Proc. Robot.: Sci. Syst.*, 2021, vol. 5, p. 2.
- [5] F. Yang, D.-H. Lee, J. Keller, and S. Scherer, "Graph-based topological exploration planning in large-scale 3D environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 12730–12736.
- [6] C. Cao, H. Zhu, Z. Ren, H. Choset, and J. Zhang, "Representation granularity enables time-efficient autonomous exploration in large, complex worlds," *Sci. Robot.*, vol. 8, no. 80, 2023, Art. no. eadf0970.
- [7] Y. Cao, Y. Wang, A. Vashisth, H. Fan, and G. A. Sartoretti, "CAtnipp: Context-aware attention-based network for informative path planning," in *Proc. Conf. Robot Learn.*, 2023, pp. 1928–1937.
- [8] Y. Cao, T. Hou, Y. Wang, X. Yi, and G. Sartoretti, "ARiADNE: A reinforcement learning approach using attention-based deep networks for exploration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 10219–10225.
- [9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
- [10] P. Christodoulou, "Soft actor-critic for discrete action settings," 2019, *arXiv:1910.07207*.
- [11] Z. Zhang, T. Henderson, S. Karaman, and V. Sze, "FSMI: Fast computation of shannon mutual information for information-theoretic mapping," *Int. J. Robot. Res.*, vol. 39, no. 9, pp. 1155–1177, 2020.
- [12] A. Asgharivaskasi, S. Koga, and N. Atanasov, "Active mapping via gradient ascent optimization of shannon mutual information over continuous se (3) trajectories," in *Proc. RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 12994–13001.
- [13] A. Asgharivaskasi and N. Atanasov, "Semantic octree mapping and shanon mutual information computation for robot exploration," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1910–1928, Jun. 2023.
- [14] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Automat. CIRA'97. Towards New Comput. Princ. Robot. Automat.*, 1997, pp. 146–151.
- [15] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *Proc. IEEE 41st Int. Symp. Robot. ROBOTIK 2010 (6th German Conf. Robot.)*, 2010, pp. 1–8.
- [16] M. Kulich, J. Faigl, and L. Přeučil, "On distance utility in the exploration task," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 4455–4460.
- [17] 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.
- [18] D. Duberg and P. Jensfelt, "UFOExplorer: Fast and scalable sampling-based exploration with a graph-based planning structure," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2487–2494, Apr. 2022.
- [19] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1500–1507, Apr. 2020.
- [20] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 779–786, Apr. 2021.
- [21] 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.
- [22] M. Luperto, L. Fochetta, and F. Amigoni, "Exploration of indoor environments through predicting the layout of partially observed rooms," in *Proc. 20th Int. Conf. Auton. Agents MultiAgent Syst.*, 2021, pp. 836–843.
- [23] L. Schmid, C. Ni, Y. Zhong, R. Siegwart, and O. Andersson, "Fast and compute-efficient sampling-based local exploration planning via distribution learning," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7810–7817, Jul. 2022.
- [24] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q.-H. Meng, "Deep reinforcement learning supervised autonomous exploration in office environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7548–7555.
- [25] 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, Art. no. 0396.
- [26] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2020.
- [27] Y. Xu et al., "Explore-bench: Data sets, metrics and evaluations for frontier-based and deep-reinforcement-learning-based autonomous exploration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 6225–6231.
- [28] F. Chen, J. D. Martin, Y. Huang, J. Wang, and B. Englot, "Autonomous exploration under uncertainty via deep reinforcement learning on graphs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6140–6147.
- [29] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30.
- [30] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, vol. 28.
- [31] Y. Cao, Z. Sun, and G. Sartoretti, "DAN: Decentralized attention-based neural network for the minmax multiple traveling salesman problem," in *Proc. Int. Symp. Distrib. Auton. Robotic Syst.*, 2022, pp. 202–215.
- [32] P. Moritz et al., "Ray: A distributed framework for emerging AI applications," in *Proc. Operating Syst. Des. Implementation*, 2018, pp. 561–577.
- [33] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, pp. 189–206, 2013.
- [34] 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.