Zero-Shot Reinforcement Learning on Graphs for Autonomous Exploration Under Uncertainty

Fanfei Chen, Paul Szenher, Yewei Huang, Jinkun Wang, Tixiao Shan, Shi Bai and Brendan Englot

***Abstract*— This paper studies the problem of autonomous exploration under localization uncertainty for a mobile robot with 3D range sensing. We present a framework for self-learning a high-performance exploration policy in a single simulation environment, and transferring it to other environments, which may be physical or virtual. Recent work in transfer learning achieves encouraging performance by domain adaptation and domain randomization to expose an agent to scenarios that fill the inherent gaps in sim2sim and sim2real approaches. However, it is inefficient to train an agent in environments with randomized conditions to learn the important features of its current state. An agent can use domain knowledge provided by human experts to learn efficiently. We propose a novel approach that uses graph neural networks in conjunction with deep reinforcement learning, enabling decision-making over graphs containing relevant exploration information provided by human experts to predict a robot's optimal sensing action in belief space. The policy, which is trained only in a single simulation environment, offers a real-time, scalable, and transferable decision-making strategy, resulting in zero-shot transfer to other simulation environments and even real-world environments.**

I. Introduction

In this paper we consider the autonomous exploration of an unknown environment by a range-sensing mobile robot reliant upon simultaneous localization and mapping (SLAM). Many recent solutions to this variant of active SLAM adopt a utility function that manages the trade-off between exploration and place-revisiting, and have high computational complexity due to the need for forward-simulation of SLAM along candidate paths. Although learning-based exploration algorithms can estimate an optimal action with greatly reduced computation time, these algorithms can require extensive training across many representative environments, and are vulnerable to poor generalizability when transferring a learned policy to a new environment. Our recent prior work [1] proposed a generalized graph representation, which we termed the *exploration graph*, for learning-based exploration under uncertainty, compatible with 2D landmark-based SLAM, deep reinforcement learning (DRL) [2] and graph neural networks (GNNs). However, many complex real-world environments cannot be captured using landmarks, and this approach has limited applicability in such environments.

We propose a new structure for the exploration graph in this paper, intended to support pose SLAM with dense 3D observations; the graph incorporates only poses and frontiers as nodes, without landmarks, to provide more generalizability. With the feature vector for the GNN provided by human experts, the agent has access to parameters directly relevant to the active SLAM task. Therefore the learning agent in our framework, despite the complexity of its dense 3D observations, only needs to use a single training environment to learn, from graphs, how to manage the trade-off between exploration and place revisiting. Fig. 1 summarizes our approach. After training in a virtual environment, the robot will use its learned policy without fine-tuning to explore new unknown (real and virtual) environments containing obstacles of different sizes, shapes, and spatial arrangements.

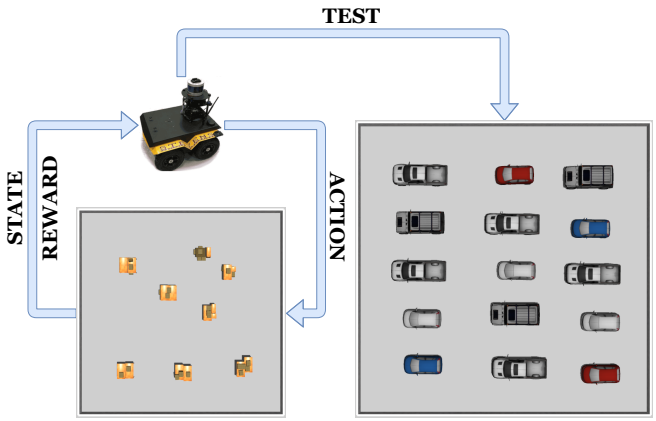


Fig. 1: **An illustration of our framework.** The robot is trained in a single representative simulation environment by DRL with GNNs. Then the GNN-based policy guides the robot to perform its exploration task in a testing environment of a different size, containing new objects, which may be real or virtual.

In this paper, we present a learning-based active SLAM framework with DRL on an optimized exploration graph which only needs a single virtual training environment to perform zero-shot RL transfer to other real and virtual environments[1]. We demonstrate that the GNN-based RL policy can be trained efficiently and transferred to new environments without parameter-tuning.

A. Related Work

Information-theoretic exploration methods focus on efficiently reducing the entropy in a mobile robot's evolving occupancy map. Specifically, [3] repeatedly selects the robot sensing action that maximizes the mutual information (MI) between the robot's range beams and the cells of its occupancy map. The Cauchy-Schwarz quadratic mutual

F. Chen, P. Szenher, Y. Huang, J. Wang and B. Englot are with the Department of Mechanical Engineering, Stevens Institute of Technology, USA, {fchen7, pszenher, yhuang85, jwang92, benglot}@stevens.edu

T. Shan is with the Computer Science & Artificial Intelligence Laboratory, Massachusetts Institute of Technology, USA, shant@mit.edu

S. Bai is with Wing, Alphabet Inc., baishi@wing.com

[1]Video attachment: https://youtu.be/62phOSf2HEg

information (CSQMI) has also been adopted, to reduce the computation time during sensing action selection [4]. [5] proposed to travel to the most informative map frontier predicted using Gaussian process (GP) occupancy maps.

Occupancy map entropy and a robot's localization uncertainty are both considered in active SLAM exploration algorithms. The entropy of the robot trajectory and its map are utilized in [6] to reduce both quantities. In [7], robot uncertainty during exploration dictates the particle weights applied in a particle filtering framework. Wang et al. [8], [9] proposed using *virtual landmarks* within an Expectation-Maximization (EM) inspired exploration algorithm to manage both robot uncertainty and its influence on occupancy map accuracy, using a novel utility function that explores while driving down the total uncertainty of a *virtual map*.

However, the high computational cost of the above state-of-the-art approaches limits their scalability. Alternatively, learning-based exploration methods can provide scalable, real-time decision making and near-optimal exploration policies. Without considering uncertainty, [10], [11] adopted GP modeling of MI and a subsequent Bayesian optimization active sampling approach to reduce the computational complexity of information-theoretic exploration. Furthermore, a supervised deep learning method [12] was proposed to predict the most informative sensing action given a local submap as input. DRL has also been used to select near-optimal, entropy-reducing sensing actions using 2D [13] and 3D [14] occupancy maps as input data. However, such learning-based methods need to capture a large variety of parameters to be applied successfully in complex environments. The training processes are intensive, and the learned policy is often challenging to transfer to a new environment.

Graphs can provide a generalized topological representation of a robot and its environment, and learning from graphs is an emerging research area. Combining graphs with neural network models [15] provides exceptional performance in many research fields. Graph Nets [16] are adopted to solve control problems by formulating a graph that represents the state of a dynamical system [17]. A robot navigation problem is solved in [18] with localization graphs and camera images. Combining graphs with DRL, Wang et al. [19] showed policies learned with GNNs are capable of solving control problems. To address the transfer of policies learned for mobile robot exploration under localization uncertainty, our prior work proposed adopting an *exploration graph* as a generalized representation of a robot's state [1], [20].

Transfer learning offers a mechanism for training robots safely in simulation environments, avoiding the expense of training a complex system completely in real-world environments. Bruce et al. [21] proposed an off-line interactive replay for real-world environment DRL training. In [22], domain randomization is adopted to fill the gap between simulated and real-world environments. Genc et al. [23] proposed zero-shot reinforcement learning using an attention mechanism to extract important features from camera images.

In this work we revisit the *exploration graph* as an appealing generalized representation of a robot's state, trajectory and environment, which, when combined with relevant features selected by human experts, allows learned exploration policies to be successfully transferred to new environments. The new measures described in the sections below specifically address how our DRL GNN framework can be applied, for the first time, to the exploration of environments populated with complex obstacles under dense 3D sensor observations. Successful training now relies on high-fidelity simulation, and accordingly, our approach offers a highly efficient training process to meet these new requirements.

B. Paper Organization

We first introduce a SLAM framework compatible with the proposed exploration graph in Section II. We then explain the framework for learning a robot exploration policy in Section III. In Section IV, we present the testing results of our robot exploration experiments, with conclusions provided in Section V.

II. Problem Formulation and Approach

A. Simultaneous Localization and Mapping Framework

We use the same SLAM framework as in [9] to support exploration. An illustration of the SLAM factor graph is shown in Fig. 2(a). There are two types of sequential factors; blue factors indicate the *odometry* measurements (ϕ^O) between two consecutive poses, and green factors represent *sequential scan matching* constraints (ϕ^{SSM}). These are provided by the iterative closest point (ICP) algorithm, using 3D LiDAR point clouds to estimate the relative transformation between consecutive poses. There are also two types of loop closure constraints. *Pose matching* (ϕ^{PM}), indicated by magenta in Fig. 2(a), provides a loop closure constraint when the point cloud from the current pose has been successfully matched with the point cloud from a previous pose. Finally, *segment matching* (ϕ^{SM}), colored by red in Fig. 2(a), achieves a loop closure when two poses observe the same segmented object in their respective point clouds (matched using descriptors, according to the approach of [24]).

The motion model and the measurement model of our SLAM framework are defined as:

$$\mathbf{x}_i = h_i(\mathbf{x}_{i-1}, \mathbf{u}_i) + \mathbf{w}_i, \quad \mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, Q_i), \tag{1}$$

$$\mathbf{z}_k = g_k(\mathbf{x}_{i_k}) + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, R_k), \tag{2}$$

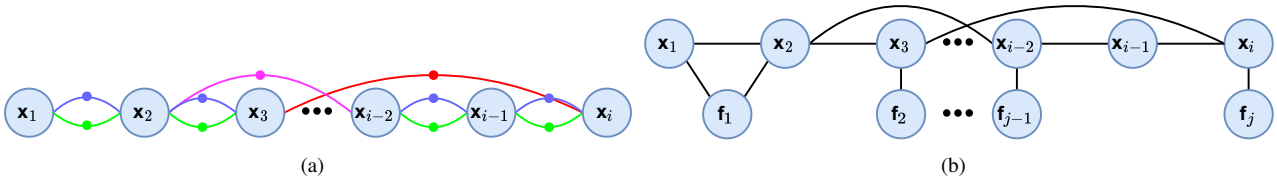
where $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^t$ are 6-DOF robot poses and $\mathcal{U} = \{\mathbf{u_i}\}_{i=1}^t$ is a given motion input. Then we define the factor graph:

$$\phi(\mathbf{x}) = \phi^0(\mathbf{x}_0) \prod_i \phi_i^{\text{O}}(\mathbf{x}_i) \prod_j \phi_j^{\text{SSM}}(\mathbf{x}_j) \qquad \text{(sequential)}$$

$$\prod_p \phi_p^{\text{PM}}(\mathbf{x}_p) \prod_q \phi_q^{\text{SM}}(\mathbf{x}_q). \qquad \text{(loop closures)}$$

The SLAM problem then becomes a nonlinear least-squares optimization problem on a factor graph. We use iSAM2 [25] to solve this problem.

We adopt the *virtual map* framework from the EM exploration algorithm of [8]. A virtual map is uniformly discretized at the same or lower resolution than the robot's



(a) (b)

Fig. 2: **An illustration of the SLAM factor graph and the exploration graph.** Left: The SLAM factor graph contains four different types of constraints: blue factors are provided by odometry measurements; green factors are obtained from sequential scan matching of two consecutive poses; red factors represent the loop closures provided by point cloud segment matching; magenta factors are loop closures generated by pose matching. Right: The corresponding exploration graph is shown, containing all poses from the SLAM factor graph, and waypoints representing map frontiers. If two poses have a constraint joining them in the SLAM factor graph, an edge will be assigned to connect these two poses. The current pose x_i is connected to the nearest frontier, and any frontiers whose paths achieve place revisiting are connected to the prior poses they revisit. All the edges in the exploration graph are weighted with Euclidean distances.

occupancy map, and contains *virtual landmarks*, $\tilde{\mathbf{l}}_k \in \tilde{\mathcal{L}}$, populating the map's cells. Each virtual landmark has a large initial covariance, which will be driven down by the robot as it observes the contents of the map cell. In this setting, the goal of exploration is to minimize the covariance of all *virtual landmarks*, which leads a robot to both explore efficiently and to produce an accurate map. The utility function of the current state is defined as follows:

$$U(\tilde{\mathcal{L}}) = \sum_{\tilde{\mathbf{l}}_k \in \tilde{\mathcal{L}}} \log\det(\Sigma_{\tilde{\mathbf{l}}_k}), \tag{3}$$

where $\Sigma_{\tilde{\mathbf{l}}_k}$ is the covariance matrix for virtual landmark $\tilde{\mathbf{l}}_k$.

B. Exploration Graph

We define the exploration graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as shown in Fig. 2(b). There are two types of vertices in $\mathcal{V}$. $\mathcal{X} \subset \mathcal{V}$ contains the robot pose history. Poses connected by constraints in the SLAM factor graph in Fig. 2(a) are also connected with edges in the exploration graph. Furthermore, the exploration graph includes exploration waypoints derived from map frontiers, $\mathcal{F} \subset \mathcal{V}$. These frontier nodes are extracted from the map's boundaries between free and unknown areas. The current pose x_t is connected with the *nearest* frontier to its location. If a frontier can provide place-revisiting through either pose matching or segment matching, we connect the previous poses associated with that loop closure to this frontier. All other frontiers, which are neither the nearest frontier to the current pose, nor achieve place-revisiting, are excluded from the exploration graph. All edges in the graph are weighted by their Euclidean distances.

Each vertex $\mathbf{n_i} \in \mathcal{V}$ has a feature vector

$$\mathbf{s}_i = [s_{i_1}, s_{i_2}, s_{i_3}, s_{i_4}],$$
$$s_{i_1} = \phi_{\mathrm{A}}(\Sigma_i), \tag{4}$$
$$s_{i_2} = \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2}, \tag{5}$$
$$s_{i_3} = \arctan2(y_i - y_t, x_i - x_t), \tag{6}$$
$$s_{i_4} = \begin{cases} 0 & \mathbf{n_i} = \mathbf{x_t} \\ 1 & \mathbf{n_i} \in \{\mathbf{f_n}\} \\ -1 & \text{otherwise} \end{cases}. \tag{7}$$

In Eq. (4), we adopt the A-Optimality [26] criterion as a metric to evaluate the uncertainty level of each node. The covariances of frontier vertexes $\mathcal{F}$ are extracted from the virtual map. We capture the geometric information of the current robot state using Eqs. (5) and (6), which contain the relative distance and orientation information between current pose x_t and the node n_i. The last feature s_{i_4} is used to indicate the identity of each node. The current pose is labeled as 0, all previous poses are -1, and frontiers are 1.

III. Algorithms and System Architecture

A. Graph Neural Networks

In this paper, we use two Graph U-Nets (g-U-Nets) [27] to serve as the policy network and the value network, respectively. Similar to U-Net [28], g-U-Nets have graph pooling layers to encode the input feature vectors of nodes in the input graph. Additionally, the graph unpooling layers are used to decode the graphs in the hidden layers to provide the output graphs. Besides the pooling and unpooling layers, each encoder and decoder has a Graph Convolutional Network (GCN) [29] layer to update the graph features. The depth of our g-U-Nets is 3 and the number of features for each hidden layer is 1000. A multilayer perceptron (MLP) output layer is adopted to provide the final output prediction. A dropout layer with a 0.5 dropout rate is placed between the output of our g-U-Nets and the MLP output layer.

B. Deep Reinforcement Learning

In our framework, the robot is solving a decision-making problem over exploration graphs using a learned policy from DRL. The candidate actions are paths from the current pose to the frontier nodes in the exploration graph. The exploration graph contains information about all past poses, their respective uncertainty, and how they relate to map frontiers. At each decision-making instant k, the state of our system is represented by an exploration graph $\mathcal{G}_k \in \mathsf{G}$. A reward $R_k \in \mathbb{R}$ is assigned to the selected action $\mathbf{f}_k \in \mathsf{F}_{\mathcal{G}_k}$. The overall decision-making process can be modeled as a Markov Decision Process $\langle \mathsf{G}, \mathsf{F}, \Pr, \gamma \rangle$ [30].

We consider the A2C [31] policy-based DRL algorithm in this paper, for which two separate g-U-Nets serve as the policy network and the value network. The loss function is defined as follows:

$$L_{\mathrm{A2C}}(\mathcal{D}) = \mathbf{E}_{\mathcal{B}\sim\mathcal{D}}[L^{(1)}_{\mathrm{A2C}} + \eta L^{(2)}_{\mathrm{A2C}}], \tag{8}$$
$$L^{(1)}_{\mathrm{A2C}} = [A(\mathcal{G}, \mathbf{f}) \log \pi(\mathbf{f}|\mathcal{G}) + \beta(A(\mathcal{G}, \mathbf{f}))]^2,$$
$$L^{(2)}_{\mathrm{A2C}} = \sum_{\mathbf{f}\in\mathsf{F}_{\mathcal{G}}} \pi(\mathbf{f}|\mathcal{G}) \log \pi(\mathbf{f}|\mathcal{G}),$$

Algorithm 1: Reward Function

input: Exploration graph $\mathcal{G}$, Frontier node $\mathbf{f}$
Calculate the raw reward (Eq. 9)
$\mathcal{R}^0_{\mathcal{G}} = U^0(\tilde{\mathcal{L}}) - U'_{\mathcal{U}}(\tilde{\mathcal{L}}') - \alpha C(\mathcal{U})$
Normalize the raw reward
$\mathcal{R}_{\mathcal{G}} = \{r_{\mathbf{f}'} = \mathcal{R}^0_{\mathcal{G}}\}$
$l = \min \mathcal{R}_{\mathcal{G}}$, $u = \max \mathcal{R}_{\mathcal{G}}$
$r_{\mathbf{f}} \leftarrow (r_{\mathbf{f}} - l)/(u - l)$
Compute projection based on nearest frontier
$\mathbf{f}_t = \texttt{nearest\_frontier}(\mathbf{x}_t)$
if $u = \texttt{raw\_reward}(\mathbf{f}_t)$ **then**
  return $r_{\mathbf{f}} - 1$ # $r(\mathcal{G}, \mathbf{f}) \in [-1, 0]$
end
return $2r_{\mathbf{f}} - 1$ # $r(\mathcal{G}, \mathbf{f}) \in [-1, 1]$

where the *advantage* function is defined as $A(\mathcal{G}, \mathbf{f}) = Q(\mathcal{G}, \mathbf{f}) - V(\mathcal{G})$ to evaluate the difference between the state value and the state-action value. $\beta \in \mathbb{R}$ is a coefficient for the loss of the value function. The entropy coefficient $\eta \in \mathbb{R}^+$ is used to weigh output entropy for encouraging exploration during training.

C. Reward Function

We use the utility function given in Eq. (3) from the EM exploration algorithm [8], whose behavior we wish to emulate, to compute the raw reward for actions that travel to each candidate frontier. The raw reward is defined as follows:

$$\mathcal{R}^0_{\mathcal{G}} = U^0(\tilde{\mathcal{L}}) - U'_{\mathcal{U}}(\tilde{\mathcal{L}}') - \alpha C(\mathcal{U}), \tag{9}$$

where $\mathcal{U}$ contains sequential actions to the selected frontier position. The output of cost-to-go function $C(\mathcal{U})$ is the travel distance weighed by coefficient α, expressing a preference for shorter paths. We then set a new range for these raw rewards by linear normalization. If the frontier selected by the EM algorithm is the *nearest* frontier associated with the current pose, the range of the reward is $[-1, 0]$. Otherwise, the range is $[-1, 1]$. The reward function is described in Alg. 1.

D. Computational Complexity

For the EM algorithm, the computational complexity of the decision-making process is $\mathcal{O}(N_{\text{actions}}(C_1 + C_2))$, where $C_1 = \mathcal{O}(n^3)$ bounds the complexity of the iSAM2 update (n is the number of poses), and $C_2 = \mathcal{O}(m)$ bounds the covariance update for virtual landmarks (m is the number of virtual landmarks) [8]. The computation time increases significantly in the size of the state-action space, limiting the framework's applicability. On the other hand, as shown in [1], the computation time for decision-making with our fully trained DRL GNN framework is nearly constant, allowing real-time performance across a wide range of problems.

IV. Experiments and Results

A. Experimental Setup

We assume that an unmanned ground vehicle (UGV) must explore an indoor environment populated with 3D obstacles, it relies upon 6 degree-of-freedom SLAM, and it is permitted to reason about exploration with respect to the ground plane, where belief space planning is performed. The simulation environments used in this work are built in Gazebo and explored using the Robot Operating System (ROS). A simulated Clearpath Jackal robot is equipped with wheel odometry and a VLP-16 3D LiDAR. We restrict the sensor range of our LiDAR to 3 meters to induce challenging pose uncertainty in our exploration comparison. We adopt the default noise settings in Gazebo for the simulated sensors, and we manually add Gaussian noise to robot translation and rotation actions of standard deviation 0.01m and 0.08°, respectively. We use the same SLAM framework employed in [9], with the GTSAM library [32]. Dijkstra's algorithm is used to generate a path to each frontier waypoint.

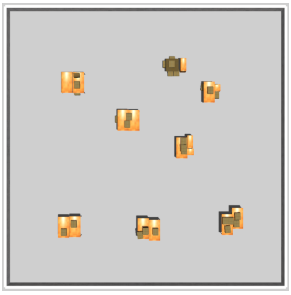


Fig. 3: The office-like Gazebo environment used for training.

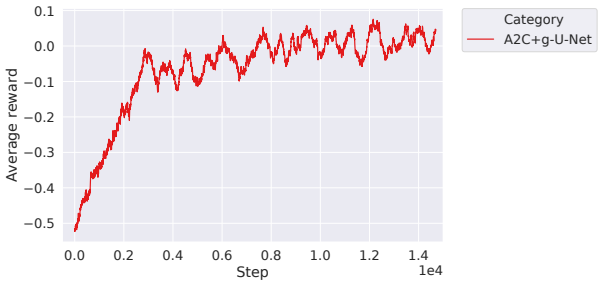


Fig. 4: The average reward during training.

A 2D map of virtual landmarks is maintained in the ground plane to provide the reward for each decision-making selection during the DRL policy training. The resolution of the map is 0.5m per cell and the standard deviation of the initial covariance of each virtual landmark is 0.2m. We terminate the exploration task once 85% of the environment's ground plane has been explored. Additionally, we keep track of the robot's volumetric sensor coverage of the environment using a separate 3D occupancy map of the workspace.

Our graph neural network models are trained using PyTorch Geometric [33]. The desktop used for policy training and simulation testing is equipped with an Intel i9 3.6Ghz CPU and an Nvidia Geforce Titan RTX GPU. For real-world exploration experiments (testing only), our algorithms run on a Dell Precision 3541 mobile workstation laptop which has an Intel Xeon CPU and an Nvidia Quadro P620 GPU.

B. Policy Training

To evaluate the transferability of our proposed approach, we only use one environment during training. The office-like

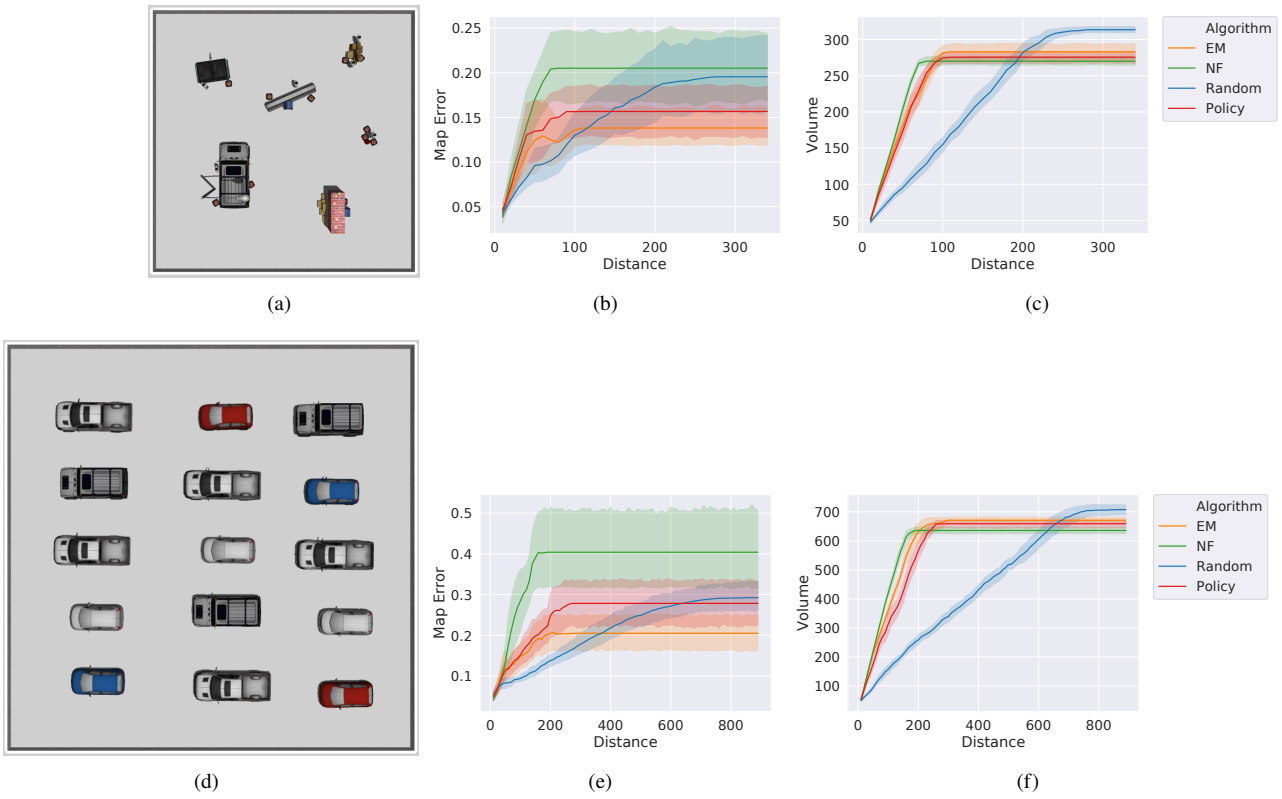


Fig. 5: Autonomous exploration results from our Gazebo simulation testing environments.

training environment is shown in Fig. 3. We randomly placed 8 objects in this environment. The robot has nine fixed initial locations in this environment from which it begins exploring.

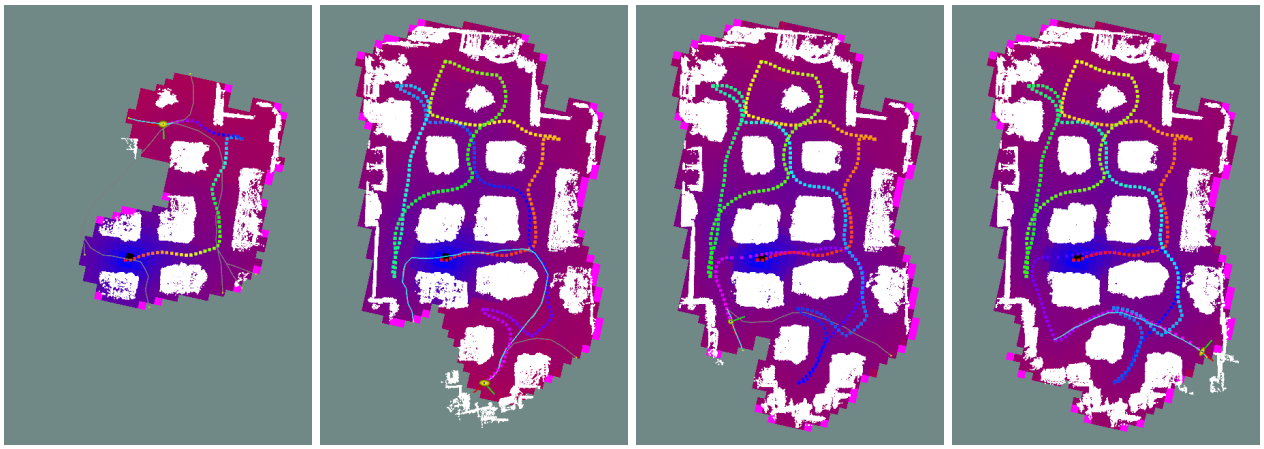
We apply 15,000 training episodes in total, which, motivated by the expense of our ROS/Gazebo simulation, is orders of magnitude fewer than in our prior work with 2D landmark-based SLAM [1]. We set the learning rate to 0.0001 and perform a policy update every 10 steps. The average reward obtained during training is shown in Fig. 4. Throughout this process, the state of the system is represented by the exploration graph, and the action space is comprised of the frontier nodes in the exploration graph.

C. Simulated Exploration Comparison

During the test phase, the trained policy takes exploration graphs as input data to support the prediction of the next frontier waypoint. There is no uncertainty propagation at the robot's decision-making step, permitting real-time computation. We use the learned policy trained in the office-like environment to test in two different environments. In Fig. 5(a), we build a "street environment" which has the same $20m \times 20m$ size as the training environment. However, this street environment contains fewer objects. Also, the size and the shape of these objects differ from the training environment. The exploration results are shown in Fig. 5(b) and 5(c). We compare the learned graph-based policy with (1) a nearest frontier (NF) approach [34], (2) the EM algorithm [8], [9], and (3) a random frontier selection approach over 10 exploration trials. For each trial, we always initialize the robot from the center of the testing environment. We compute the mean absolute error between the mapped 3D points associated with the estimated robot trajectories and the ground truth trajectories to determine the map error. We also compute the total 3D occupied volume mapped during each trial for the exploration efficiency comparison.

The EM algorithm achieves the lowest map error in the end, but has a slightly worse exploration efficiency than the NF approach. Although the NF approach is the most efficient method for covering an unknown environment, it offers the worst map accuracy during exploration because it achieves the fewest loop closures. The random method achieves the most loop closures during exploration, but its long travel distances generate a large accumulated error that cannot be completely eliminated by these loop closures. It also covers the environment very inefficiently. Our learned policy achieves a very similar exploration efficiency to the EM algorithm, and its map accuracy is surpassed only by EM algorithm, whose performance we seek to emulate.

The second simulation testing environment is shown in Fig. 5(d). In this "parking garage" environment, there are fifteen evenly spaced cars and the size of this environment is $30m \times 30m$, which is larger than the training environment. Like the first simulation environment, each exploration algorithm has 10 trials initialized from the center of the environment. The learned policy has a slightly worse exploration efficiency than the EM algorithm in this large environment, but it once again achieves the second lowest map error (again, second to the EM algorithm) compared with other exploration methods. All other algorithms have the same relative performance as in the "street" environment.



(a) (b) (c) (d)

Fig. 6: Ground-plane map and trajectory resulting from our graph-based policy's exploration of the Stevens ABS Engineering Center.

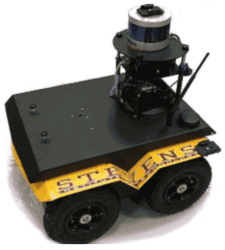


Fig. 7: The Clearpath Jackal UGV used in our experiments.

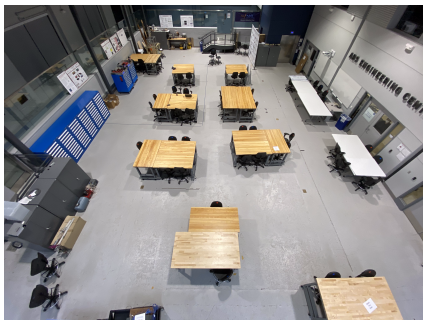


Fig. 8: Environment used for real-world deployment of our UGV.

D. Real-world Exploration

Our learned policy can also be transferred to a real-world environment. Our Clearpath Jackal UGV, shown in Fig. 7, is equipped with the same odometry and LiDAR sensing as the simulated UGV. We test our learned policy in the ABS Engineering Center at Stevens Institute of Technology shown in Fig. 8. In Fig. 6, we present an example of autonomous exploration using the learned, graph-based policy. The robot pose is represented by red-green axes, and the yellow ellipsoid indicates the uncertainty of the current pose. The cyan path is the path to the selected frontier, and gray paths are for other unselected frontiers. The cost map covering the ground plane represents the uncertainty of the current virtual map, where blue color indicates the lowest uncertainty. Magenta represents the high uncertainty of our virtual map prior. We terminate the exploration task if there are no frontiers detected on the current map. In Fig. 6(a), the robot chooses the nearest frontier to explore the environment, rather than the longer-distance path to obtain a loop closure, because this is the beginning of the exploration and a loop closure only reduces the uncertainty of a small portion of the current virtual map. In Fig. 6(b), the robot selects a longer-distance path to obtain a loop closure to reduce the uncertainty of the current virtual map. In Fig. 6(c), the uncertainty of the right bottom area on the map is reduced by taking the selected path. We present the final exploration result in Fig. 6(d). The overall exploration process is shown in our video attachment.

V. Conclusions

In this paper, we present a zero-shot transfer learning framework for mobile robot exploration under uncertainty that leverages an exploration graph as an efficient abstraction of a robot's state and environment. We have enhanced the DRL GNN framework developed in our prior work [1] so it can be applied, for the first time, to the exploration of environments populated with complex obstacles, perceived using dense 3D range observations. Successful training now depends on high-fidelity simulation, and accordingly, our approach offers a highly efficient training process to meet these requirements; the exploration policy is trained in a single virtual environment and is successfully transferred to both virtual and real environments containing different obstacle quantities, arrangements, and geometries. The exploration graph proposed in this work offers generality that is suitable for a wide variety of real-world exploration tasks, which we hope to study further in future work. Anticipated future versions of this system will adjust their exploration strategies to adapt to new environments. Also, although our policy is unlikely to exceed the EM algorithm's performance, because we adopt its utility function to assign rewards, we aim to enhance our reward function in future work to allow our framework to outperform the EM algorithm.

Acknowledgments

This research has been supported by the National Science Foundation, grant numbers IIS-1652064 and IIS-1723996.

References

[1] F. Chen, J. Wang, T. Shan, and B. Englot, "Autonomous Exploration Under Uncertainty via Deep Reinforcement Learning on Graphs," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6140–6147, 2020.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski and S. Petersen, "Human-level Control through Deep Reinforcement Learning," *Nature*, vol. 518(7540), pp. 529–533, 2015.

[3] B. J. Julian, S. Karaman and D. Rus, "On Mutual Information-Based Control of Range Sensing Robots for Mapping Applications," *The International Journal of Robotics Research*, vol. 33(10), pp. 1375–1392, 2014.

[4] B. Charrow, S. Liu, V. Kumar and N. Michael, "Information-theoretic Mapping using Cauchy-Schwarz Quadratic Mutual Information," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4791–4798, 2015.

[5] M. G. Jadidi, J. V. Miró and G. Dissanayake, "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots," *Autonomous Robots*, vol. 42, pp. 273–290, 2018.

[6] R. Valencia, J. V. Miró, G. Dissanayake and J. Andrade-Cetto, "Active Pose SLAM," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1885–1891, 2012.

[7] C. Stachniss, G. Grisetti and W. Burgard, "Information Gain-based Exploration using Rao-Blackwellized Particle Filters," *Proceedings of Robotics: Science and Systems*, pp. 65–72, 2005.

[8] J. Wang and B. Englot, "Autonomous Exploration with Expectation-Maximization," *Proceedings of the International Symposium on Robotics Research*, pp. 759–774, 2017.

[9] J. Wang, T. Shan and B. Englot, "Virtual Maps for Autonomous Exploration with Pose SLAM," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4899–4906, 2019.

[10] S. Bai, J. Wang, K. Doherty and B. Englot, "Inference-Enabled Information-Theoretic Exploration of Continuous Action Spaces," *Proceedings of the International Symposium on Robotics Research*, pp. 419–433, 2015.

[11] S. Bai, J. Wang, F. Chen and B. Englot, "Information-theoretic Exploration with Bayesian Optimization," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1816–1822, 2016.

[12] S. Bai, F. Chen and B. Englot, "Toward Autonomous Mapping and Exploration for Mobile Robots through Deep Supervised Learning," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2379–2384, 2017.

[13] 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(2), pp. 610–617, 2019.

[14] F. Chen, S. Bai, T. Shan and B. Englot, "Self-Learning Exploration and Mapping for Mobile Robots via Deep Reinforcement Learning," *Proceedings of the AIAA SciTech Forum*, 2019.

[15] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, vol. 20(1), pp. 61–80, 2009.

[16] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner and C. Gulcehre, "Relational Inductive Biases, Deep Learning, and Graph Networks," *arXiv preprint*, arXiv:1806.01261 [cs.LG], 2018.

[17] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell and P. Battaglia, "Graph Networks as Learnable Physics Engines for Inference and Control," *Proceedings of the 35th International Conference on Machine Learning*, pp. 4470–4479, 2018.

[18] K. Chen, J. P. de Vicente, G. Sepulveda, F. Xia, A. Soto, M. Vazquez, S. Savarese "A Behavioral Approach to Visual Navigation with Graph Localization Networks," *Proceedings of Robotics: Science and Systems*, 2019.

[19] T. Wang, R. Liao, J. Ba and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," *Proceedings of the International Conference on Learning Representations*, 2018.

[20] F. Chen, J. Wang, T. Shan, and B. Englot, "Autonomous Exploration Under Uncertainty via Graph Convolutional Networks," *Proceedings of the International Symposium on Robotics Research*, 2019.

[21] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, "One-Shot Reinforcement Learning for Robot Navigation with Interactive Replay," *Advances in Neural Information Processing Systems*, 2017.

[22] I. Mordatch, K. Lowrey, and E. Todorov, "Ensemble-CIO: Full-Body Dynamic Motion Planning that Transfers to Physical Humanoids," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5307–5314, 2015.

[23] S. Genc, S. Mallya, S. Bodapati, T. Sun and Y. Tao, "Zero-Shot Reinforcement Learning with Deep Attention Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, 2019.

[24] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart and C. Cadena, "SegMatch: Segment Based Place Recognition in 3D Point Clouds," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 5266- 5272, 2017.

[25] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard and F. Dellaert, "iSAM2: Incremental Smoothing and Mapping using the Bayes Tree," *The International Journal of Robotics Research*, vol. 31 (2), pp. 216–235, 2012.

[26] M. Kaess and F. Dellaert, "Covariance Recovery from a Square Root Information Matrix for Data Association," *Robotics and Autonomous Systems*, vol. 57(12), pp. 1198-1210, 2009.

[27] H. Gao and S. Ji, "Graph U-Nets," *Proceedings of the International Conference on Learning Representations*, 2019.

[28] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical Image Computing and Computer-assisted Intervention*, pp. 234-241, 2015.

[29] T. N. Kipf and M. Welling, "Semi-supervised Classification with Graph Convolutional Networks," *Proceedings of the International Conference on Learning Representations*, 2017.

[30] M. L. Puterman, "Model Formulation," *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, pp. 17–32, John Wiley & Sons, 1994.

[31] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. v. Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp and R. Tsing, "Starcraft II: A New Challenge for Reinforcement Learning," *arXiv preprint*, arXiv:1708.04782 [cs.LG], 2017.

[32] F. Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction," *Technical Report, Georgia Institute of Technology*, GT-RIM-CP&R-2012-002, 2012.

[33] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[34] B. Yamauchi, "A Frontier-based Approach for Autonomous Exploration," *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 146-151, 1997.