Autonomous Exploration Under Uncertainty via Graph Convolutional Networks

Fanfei Chen, Jinkun Wang, Tixiao Shan and Brendan Englot

Abstract We consider a mapping and exploration problem in which a range-sensing mobile robot is tasked with mapping the landmarks in an unknown environment efficiently in real-time. There are numerous state-of-the-art methods which consider the uncertainty of a robot's pose and/or the entropy and accuracy of its map when exploring an unknown environment. However, such methods typically use forward simulation to predict and select the best action based on the respective utility function. Therefore, the computation time of such methods is often costly, and may grow exponentially with the increasing dimension of the state space and action space, prohibiting real-time implementation. We propose a novel approach that uses a Graph Convolutional Network (GCN) to predict a robot's optimal action in belief space over a graph representation of candidate waypoints and landmarks. The learned exploration policy can provide an optimal or near-optimal exploratory action and maintain competitive coverage speed with improved computational efficiency.

1 Introduction

Mapping and exploration [1] of a priori unknown environments are crucial capabilities for mobile robot autonomy. Information-theoretic exploration methods were developed to guide robots to the unexplored areas of their environment that will contribute the largest quantities of new information to an occupancy map [2], [3], [4], [5], [6]. Meanwhile, simultaneous localization and mapping (SLAM) provides a way to evaluate map accuracy and manage localization error under noisy relative measurements. Consequently, SLAM-based exploration or *active SLAM* [7], [8], [9] has been applied successfully for mapping an unknown environment autonomously under robot landmark and pose uncertainty. However, the decision-making compo-

F. Chen, J. Wang, T. Shan and B. Englot are with the Department of Mechanical Engineering, Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ, USA 07030.
e-mail: {fchen7, jwang92, tshan3, benglot}@stevens.edu

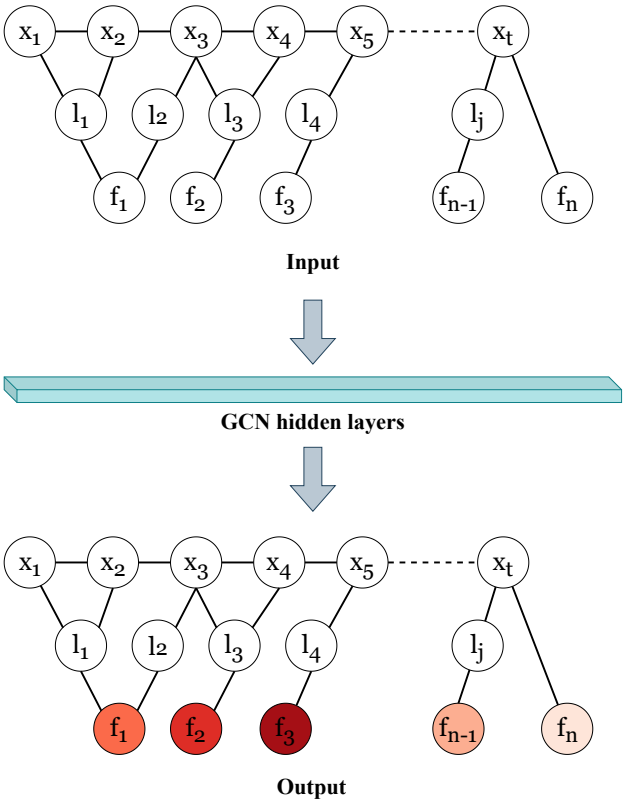


Fig. 1: An illustration of our use of a Graph Convolutional Network (GCN), in which the cyan-colored rectangular block represents its hidden layers. The original input data is the *exploration graph* associated with the current robot state, which reflects its pose history, observed landmarks, and candidate frontiers. The output graph has the same structure as the input graph, but one of the frontier nodes is selected to be the next goal location, which is represented by the darkest red color. The darkness of the red color indicates the predicted probability of optimality of the frontier.

nent of active SLAM exploration methods is time-consuming due to the need for forward simulation of future robot measurements, and prediction of the resulting map and pose uncertainty. This approach will ultimately fail for real-time decision-making with increasing dimensionality of the state space and the action space, due to the costly time complexity of such methods.

We propose a novel approach that uses an *exploration graph* to select the next sensing action via Graph Convolutional Networks (GCN) in real-time. The proposed graph captures the current robot state, its pose history, observed landmarks, and candidate frontiers. We use the Expectation-Maximization (EM) exploration algorithm [9] to generate training data with randomly generated maps and random initial robot

locations. Once trained, a mobile robot does not need to use forward simulation to compute the expected information gain of each action exhaustively. After a training phase, the GCN predicts optimal sensing actions based on the current exploration graph. Fig. 1 summarizes our approach. The nodes of the input graph have different attributes with the same size. The GCN provides an output graph which has the same structure as the input graph, with updated attributes. In our method, each node of the output graph has only one attribute; which takes on a value of one for the selected frontier node, and zero for all other nodes. Our method can provide an accurate and efficient strategy to explore an a priori unknown environment populated with sparse landmarks. For example, when an underwater robot explores a subsea environment, it needs an efficient strategy to build a map because it has a limited power supply, and a scarcity of subsea landmarks to support navigation. Meanwhile, communication constraints require onboard real-time computation for navigation decision-making.

The contributions of this paper are as follows. We use the proposed exploration graph to represent a SLAM-dependent robot's state space and action space using a generalized representation. Compared with typical metric maps, our graph only records the topological relationships between poses, observed landmarks, and candidate waypoints. Accordingly, the learned exploration policy will not be influenced as heavily by details such as the coloration of the environment, the type of feature descriptor used, or the orientation of obstacles in the workspace, and thus a smaller volume of training data is needed than would be required to learn from metric maps directly. Using this representation, we propose what is to our knowledge the first graph-learning-based active SLAM exploration approach, offering real-time viable decision-making during robot exploration under uncertainty.

1.1 Related Work

Information-theoretic exploration methods guide a robot to explore the unknown environment by repeatedly selecting the sensing action expected to be most informative [2], [3], [4]. It has also proven effective to use Gaussian process frontier maps as a predictive tool to support exploration [5], [6]. However, these methods do not consider a robot's localization uncertainty during the exploration process, which eventually leads to an inaccurate map.

Active SLAM exploration approaches have been developed to reduce both the uncertainty of a robot's pose and entropy of the map by considering the correlation between localization and information gain [7]. The approach proposed in [8] uses the particle filter to reduce the overall uncertainty for both maps and poses by capturing a trajectory's uncertainty using the particle weight. The Expectation-Maximization (EM) Exploration algorithm [9] introduces *virtual landmarks* to represent the uncertainty of unexplored regions, and a novel utility function for solving the exploration problem that seeks the most accurate map possible with every sensing action.

Learning-based exploration methods can offer reduced computation time and near-optimal exploration strategies. Bai et al. proposed a Gaussian process based

mutual information prediction method [10], and a subsequent Bayesian optimization active sampling approach [11], to speed up the prediction of mutual information throughout a robot's action space. Furthermore, [12] and [13] show that deep supervised learning and deep reinforcement learning [14] can produce high-quality solutions for the exploration problem by using local occupancy maps as input data. Moreover, global occupancy maps are used as input data to find the frontier associated with the next best view using deep reinforcement learning in [15]. However, an occupancy grid map has limitations as an input to represent the environment. For example, in some instances neural networks may be unable to perform reliably over a map with a rotation relative to the maps presented in training. Hence, such a framework may take a longer time to converge during the training phase, or even fail to converge, due to the size of the state space induced by an occupancy map.

Learning from graphs is an emerging research area. Graph Convolutional Networks (GCNs) [16] have been successfully applied in many fields such as social networks [17] and chemistry [18]. Spectral convolutions are introduced to operate the graph-based neural network model. Sanchez-Gonzalez et al. [19] proposed to solve the control problem by using graphs to represent physical models with Graph Nets [20]. For graph learning, the input data can be variable-size graphs, and the size of the output data scales with the number of nodes of the input graph.

By combining active-SLAM exploration with the graph-learning approach, we propose and develop the first graph-learning-based robot exploration framework. The proposed *exploration graph* structure offers a smaller state space for learning than an occupancy map and is a more generalized representation of a SLAM-dependent mobile robot's environment. By permitting real-time computation due to low query time complexity, a graph-learning approach is also scalable to high-dimensional state and action spaces.

1.2 Paper Organization

A formal definition of the problem is given in Section 2, including a brief discussion of the EM exploration algorithm, which generates the training data for our GCN framework, and an overview of our GCN model. Experimental results are presented in Section 3, with conclusions in Section 4.

2 Problem Formulation and Approach

2.1 Simultaneous Localization and Mapping Framework

We formulate graph-based SLAM as a least-squares problem. The definitions of the Gaussian robot motion model and the Gaussian measurement model are as follows:

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

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

where $\mathcal{X} = \{\mathbf{x}_i\}$ are robot poses, $\mathcal{L} = \{\mathbf{l}_j\}$ are landmarks and $\mathcal{U} = \{\mathbf{u_i}\}$ is a given motion input. i_k and j_k represent the ith pose and the jth landmark associated with the kth measurement. The estimate of the poses and the landmarks is obtained by solving the nonlinear least-squares equation:

$$\mathcal{X}^*, \mathcal{L}^* = \underset{\mathcal{X},\mathcal{L}}{\operatorname{argmin}} \sum_i \left\| \mathbf{x_i} - h_i(\mathbf{x_{i-1}}, \mathbf{u_i}) \right\|^2_{Q_i} + \sum_k \left\| \mathbf{z_k} - g_k(\mathbf{x_{i_k}}, \mathbf{l_{j_k}}) \right\|^2_{R_k}. \tag{3}$$

In the EM exploration algorithm used to produce our training data, this is solved using the GTSAM [21] implementation of iSAM2 [22] to construct a factor graph and perform incremental nonlinear least-squares smoothing. This graphical model-based inference approach provides Gaussian marginal distributions and Gaussian joint marginal distributions.

2.2 Expectation-Maximization Exploration Algorithm

The EM exploration algorithm, which repeatedly selects the exploratory sensing action that is expected to yield the most accurate map, offers an appealing balance between managing uncertainty and exploring efficiently [9]. *Virtual landmarks* are introduced as latent variables, and the exploration process is managed iteratively through expectation-maximization. The *virtual map*, a uniformly discretized grid map comprised of virtual landmarks in its cells (each characterized by an occupancy probability and error covariance), is estimated using the current SLAM estimate and measurements in the E-step. Then in the M-step, the optimal path is selected to minimize the uncertainty of the virtual landmarks, which are initialized with large covariance. Each cell in the initially unexplored map has a virtual landmark prior with a large initial covariance. We use A-optimality [23] for our uncertainty criterion:

$$\phi_{\mathrm{A}}(\Sigma) = \mathrm{tr}(\Sigma), \tag{4}$$

where Σ is a covariance matrix of the virtual landmarks. By calculating the trace of the covariance of all virtual landmarks, we can quantify the uncertainty of each virtual map. The definition of the cost function for the EM algorithm is as follows:

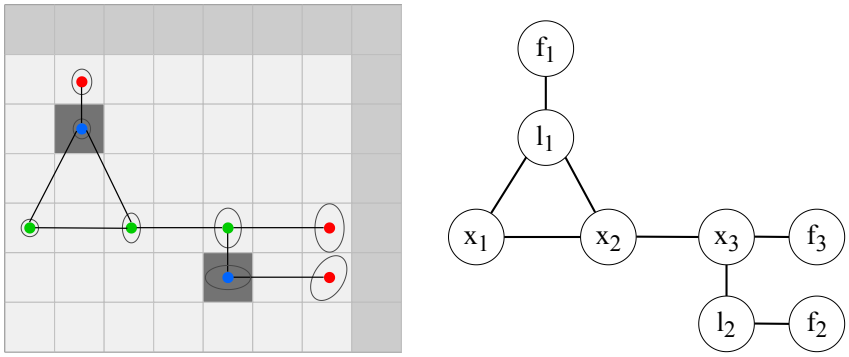
$$U \approx \sum_{\mathbf{v}_k \in \mathcal{V}} \phi_{\mathrm{A}}(\Sigma_{\mathbf{v}_k}), \tag{5}$$

where $\mathcal{V} = \{\mathbf{v}_k\}$ are virtual landmarks in a virtual map. All the virtual landmarks, which represent the possible real landmarks, are used to evaluate the performance of each potential action. The goal is to minimize map uncertainty with each decision-making step. Hence we define the action reward $r_{\mathbf{a}}$ to guide a robot accordingly:

$$r_{\mathbf{a}} = U_0 - U_{\mathbf{a}} - \alpha C_{\mathbf{a}}. \tag{6}$$

In Eq. (6), C indicates a cost-to-go to disfavor actions which have long travel distance and α is a weight for the cost-to-go. U_0 is the initial uncertainty and $U_{\mathbf{a}}$ is the uncertainty after taking the action $\mathbf{a}$.

2.3 Graph Convolutional Networks for Exploration



(a) (b)

Fig. 2: In the left figure, the grayscale color represents the probability of occupancy of a given map cell. The ellipses are estimation error covariances of each respective location on the *virtual map*. Robot poses, landmarks and frontiers are indicated by green dots, blue dots and red dots respectively. All landmark nodes and the current pose node are connected with their nearest frontiers. In the right figure, An input *exploration graph* is extracted from the current exploration state. Each edge in this graph is weighted with the Euclidean distance between the two vertices connected.

We define the *exploration graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{\{\mathbf{x}_i\}, \{\mathbf{l}_j\}, \{\mathbf{f}_n\}\}$. The edges $\mathcal{E}$ connecting pose to pose $\mathbf{x}_i$ — $\mathbf{x}_{i+1}$, and pose to landmark $\mathbf{x}_{i_k}$ — $\mathbf{l}_{j_k}$ represent odometry measurements and landmark measurements respectively. The frontier nodes $\mathbf{f}_n$ are sampled from the boundary cells between the free and unexplored space, and they are connected to the *nearest* node, either pose or landmark, in the graph. It is possible for multiple nodes to have the same frontier node. A simple example of extracting the exploration graph from a robot's current state is shown in Fig. 2.

Instead of explicitly computing the sensing action that maximizes the reward of Eq. (6) through an expensive evaluation of all the candidate frontiers, we employ a GCN to predict the optimal frontier with respect to (6). However, we use the

EM algorithm to generate training data. For each decision-making step, actions $\mathcal{A} = \{\mathbf{a}_n\}$ are generated for each frontier node with straight-line path planning, and the EM algorithm provides an action reward $\mathcal{R} = \{r_n\}$ for all frontiers via forward simulation. The reward $\mathcal{R}$ is normalized so that $r_n \in [0, 1]$. We require that the reward of an optimal frontier has to be larger than 0.95. If multiple frontiers are optimal at the same time, we will select one at random. The rewards associated with pose and landmark nodes are always 0 because we only select the robot's next action from among frontier nodes. Therefore, the training label y_{nodes} is defined as follows:

$$y_i = \begin{cases} 1 & r_i > 0.95 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

A multi-layer GCN is adopted to explore the environment. The layer-wise propagation rule of the GCN [16] is as follows:

$$H^{(l+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}), \tag{8}$$

with $\hat{A} = A + I$, where A is the adjacency matrix of the exploration graph $\mathcal{G}$ and $\hat{D}$ is the degree matrix of $\hat{A}$. $H^{(l)}$ represents the lth hidden layer of the GCN model with the activation function $\sigma(\cdot)$. $W^{(l)}$ is the weight matrix of the lth layer.

We use a three-layer GCN model. The definition of the input feature vector $\mathbf{s}_i = [s_{i_1}, s_{i_2}, s_{i_3}, s_{i_4}, s_{i_5}]$ for the node $\mathbf{n_i} \in \mathcal{V}$ is as follows:

$$s_{i_1} = \phi_{\text{A}}(\Sigma_i), \tag{9}$$

$$s_{i_2} = \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2}, \tag{10}$$

$$s_{i_3} = \arctan2(y_i - y_t, x_i - x_t), \tag{11}$$

$$s_{i_4} = p(m_i = 1), \tag{12}$$

$$s_{i_5} = \begin{cases} 0 & \mathbf{n_i} = \mathbf{x_t} \\ 1 & \mathbf{n_i} \in \{\mathbf{f_n}\} \\ -1 & \text{otherwise} \end{cases}. \tag{13}$$

In Eq. (9), we use the same A-Optimality criterion of Eq. (4) for providing a suitable feature to represent a node's uncertainty, derived from our virtual map. Relative position information to the current robot pose is provided by the second and the third features. The Euclidean distance to the current pose is the second feature in Eq. (10), where x and y coordinates represent the locations of the node and the current pose. The relative orientation to the current robot pose is the third feature as shown in Eq. (11). Map information is provided by the fourth feature, where m_i is the occupancy of the map cell associated with node n_i in Eq. (12). Since the robot needs to travel to the optimal frontier from its current pose, it is important to provide

indicator variables to denote the current pose and frontiers. In Eq. (13), we define the indicator of the current pose to be 0, all frontiers to be 1, and all other nodes -1.

For the GCN model, the definition of each hidden layer is as follows:

$$H^{(1)} = \sigma_1(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}SW^{(1)}), \tag{14}$$

$$H^{(2)} = \text{Dropout}(H^{(1)}), \tag{15}$$

$$H^{(3)} = \sigma_3(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(2)}W^{(3)}). \tag{16}$$

In Eq. (14), the size of weight matrix $W^{(1)}$ is 5×1000 because we want to upsample the number of features from 5 to 1000. The activation function σ_1 is a Rectified Linear Unit (ReLU). We add a dropout layer with 0.5 dropout rate as shown in Eq. (15). In the output layer, the size of the weight matrix $W^{(3)}$ in Eq. (16) is 1000×1 and the activation fuction σ_3 is a sigmoid function.

The cross-entropy loss function is defined as follows:

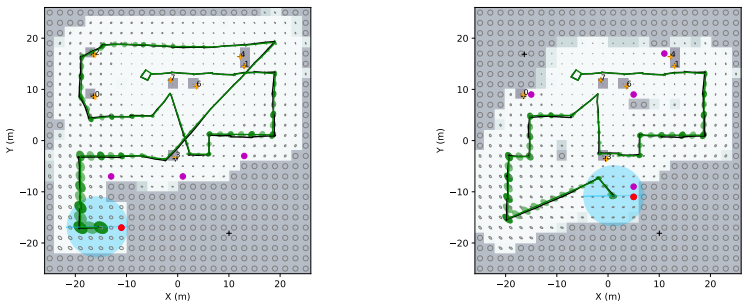
$$L = -(w_1 y \log\hat{y} + w_0(1-y)\log(1-\hat{y})), \tag{17}$$

where w_1 and w_0 are the weights for class 1 and class 0. These weights are used to balance the cost for imbalanced data. We empirically define $w_1 = 21$ and $w_0 = 1$. We use the Adam first-order gradient-based algorithm [24] when optimizing the parameters of the GCN model.

3 Experiments and Results

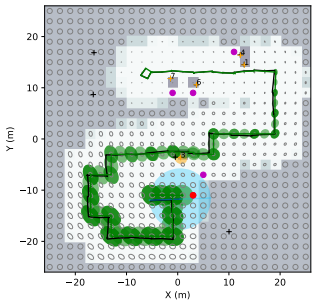
3.1 Experimental Setup

We trained the proposed GCN system over a two-dimensional landmark-based simulation environment, using 500 unique $40m \times 40m$ maps with randomly seeded landmarks. From the exploration sequences run over these maps, we selected 32 random exploration graphs as a training batch, which was repeated for 20 batches, with each batch undergoing 500 training epochs. All of this data was used to train our GCN policy - we generated new random environments excluded from this training dataset for the testing phase. In our experiments, we impose a *feature density* of 0.5 landmarks per m^2 on all training and testing maps. The simulated robot in our experiments is equipped with a $5m$ range sensor with a 360-degree field of view, and Gaussian white noise of standard deviation $0.02m$ in range and $0.5deg$ in bearing. Straight-line path planning is introduced to repeatedly lead the robot from its current location to the goal frontier location, by which it will first turn to face the goal, and then travel along a straight-line path to the goal. Although an occupancy grid is used to keep track of landmark locations, the point landmarks are assumed not to pose

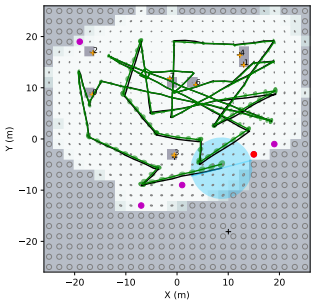


(a) GCN

(b) EM



(c) Nearest Frontier

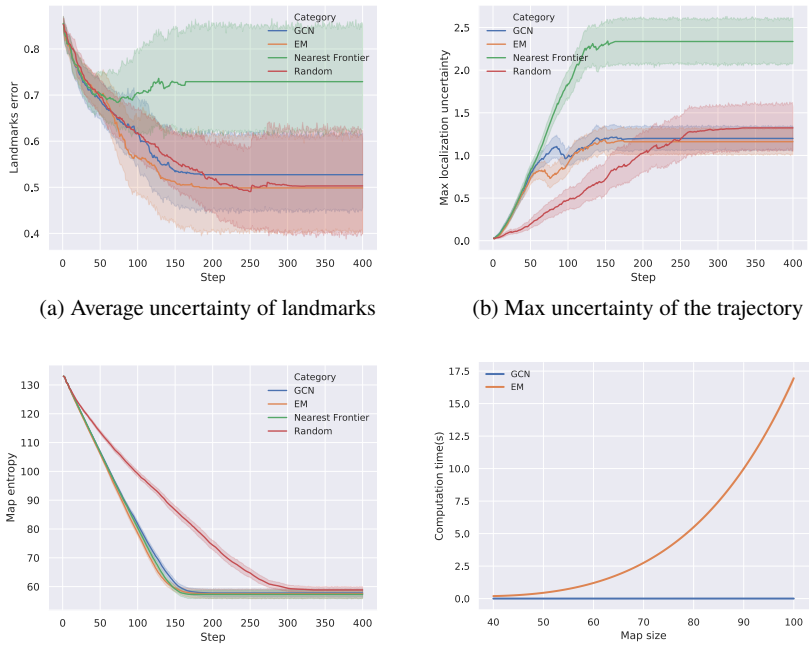


(d) Random

Fig. 3: Representative exploration trials across four different methods on the same $40m \times 40m$ map, over an equivalent number of decision-making instances.

collision hazards. During one time step of the simulation, the robot can rotate up to $180deg$, or translate forward up to $2m$; the standard deviation of Gaussian white process noise on the translation and rotation actions is $0.1m$ and $0.2deg$, respectively. The resolution of the virtual map is $2m$ (which is the size of all square map cells), and the initial error covariance for virtual landmarks is $1m^2$ in each dimension. The virtual landmarks of the EM algorithm are only used here to calculate the reward of Eq. 6, which serves as an indicator of map accuracy weighed against travel expense. The virtual map is otherwise excluded from iSAM2 factor graphs and from the exploration graph. For all the exploration experiments, the task will be terminated after 85% of a map has been explored.

The simulation environment is written in Python and C++, and our GCN model is trained using TensorFlow [25]. All of the test algorithms were run on a computer equipped with an Intel i9 3.6Ghz CPU and an Nvidia GeForce Titan RTX GPU.



(a) Average uncertainty of landmarks

(b) Max uncertainty of the trajectory

(c) Map entropy reduction

(d) Planning time

Fig. 4: The result of 50 exploration trials of each method, with the same randomly initialized landmarks and robot start locations, on $40m \times 40m$ maps (the first three metrics shown are plotted per time-step of the simulation).

3.2 Exploration Comparison

We compared the proposed GCN framework with (1) a nearest frontier approach, (2) the selection of a random frontier, and (3) the EM approach over 50 exploration trials. The test environments were randomly generated and excluded from the training data. For each trial, every approach uses the same random map and the same random initial robot location. We provide examples of the four competing exploration methods over the same random map with the same random initial location, in Fig. 3. The error ellipses (0.5 standard deviations) shown in all grid cells represent the uncertainty of the virtual landmarks. Derived from the robot poses that observed these cells, their covariance is a measure of map accuracy. These virtual landmarks are only used for reward calculation in Eq. 6 and they are not included in the SLAM factor graph or exploration graph. The blue circle is the sensing area, and the center of the blue circle is the current location of the robot. The orange error ellipses show the uncertainty of the real landmarks and the green error ellipses show the uncertainty of the previous robot poses. The magenta points are frontier nodes that are potential goal positions

for the robot, and the red point is the selected goal frontier. The cross symbols are the locations of the real landmarks. We extend the boundary of the map by $6m$ in each direction to ensure the robot can successfully obtain the reward around the boundary area, while frontier nodes are not allowed to be generated in these extended areas, and their observation does not contribute to the evaluation of reward.

We select three metrics to evaluate the relative performance of four exploration methods, as shown in Fig. 4. The average landmark uncertainty is a key element to evaluate the accuracy of the final map, and we only consider real landmarks in this comparison. The maximum uncertainty of the robot's trajectory is used for showing the accuracy of pose estimation along the trajectory. Map entropy reduction is used to evaluate the efficiency of robot exploration. The random method and the EM algorithm provide the lowest landmark uncertainty (Fig. 4a), but the random method is the most inefficient method for exploring the environment (Fig. 4c). In Fig. 3d, it is clear that the random method has a disorganized trajectory which has more chances to visit the previous landmarks to reduce uncertainty, and a much longer travel distance which is almost twice that of other methods (Fig. 4c). With respect to map accuracy, the GCN method is slightly worse than the EM algorithm (which was used to train the GCN) and the random method, but the uncertainty of those three methods is at a very similar level in Fig. 4a. In Fig. 4b, the GCN method and EM algorithm have very similar performance, which is slightly better than the Random method. The nearest frontier method achieves the same map entropy reduction performance as the GCN and EM approaches (Fig. 4c), but it achieves the worst performance for the uncertainty of the landmarks and the poses along the trajectories as shown in Fig. 4a and Fig. 4b, because this method only chooses the nearest frontier to explore the environment and never actively seeks a frontier which is near a previous landmark to reduce uncertainty, as shown in Fig. 3c. Overall, the GCN policy and the EM algorithm demonstrate the best performance in producing an accurate map of landmarks, and accurate pose estimates, using time-efficient trajectories.

3.3 Computation Time

In Section 3.2, both the GCN and EM exploration algorithms have similar satisfying exploration performance. However, the time complexity of the two methods is very different. The time complexity of the EM algorithm's decision-making is $\mathcal{O}(N_{action}(C_1 + C_2))$, where N_{action} is the number of the frontiers, C_1 is the cost of each iSAM2 predictive estimate over a candidate trajectory (a function of the number of mapped true landmarks and prior poses) and C_2 is the cost of the covariance update for virtual landmarks (a function of the number of prior poses, the virtual map resolution and the size of the region being explored). Therefore, the computational cost of the EM algorithm increases dramatically with increasing size of the state space (including the number of landmarks) and the action space, and it will ultimately lead to failure for complex and large-scale real-time exploration scenarios. The time complexity of decision-making for the GCN approach is $\mathcal{O}(n)$, where n

represents the number of nodes in a given exploration graph. This reflects the cost of matrix multiplication with a GPU, performed for each of the two GCN layers that require it. For the GCN approach, none of the N_{action} candidate actions need to be evaluated explicitly, hence the GCN approach remains robust to high-dimensional state and action spaces.

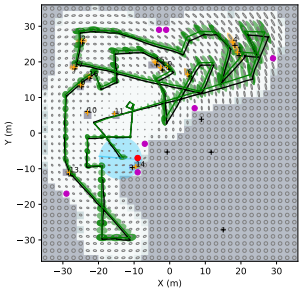
We test the computation time for exploration decision-making on four different sizes of maps, namely $40m \times 40m$, $60m \times 60m$, $80m \times 80m$ and $100m \times 100m$. Each size has the same feature density, which is 0.5 landmarks per m^2. The test result, to which a curve is fit using a sixth-order polynomial regression, is shown in Fig. 4d. The plot represents the mean of the computation time used for exploration decision-making (i.e., forward simulation). The GCN computation time is near zero, which is real-time viable. In contrast, the computation time of the EM algorithm grows sharply with increasing map size. In $100m \times 100m$ maps, the average computation time for the EM algorithm is $16.5s$, which is not acceptable for a real-time exploration task.

As mentioned above, the computation time shown in Fig. 4d captures only the decision-making component of exploration, the most costly step of the process, which entails selecting the optimal goal frontier from among the available frontier nodes. The computation time of other tasks, such as solving SLAM over the robot's true trajectory, updating the virtual landmarks as new measurements are collected, and generating frontier nodes from the grid map, is not constant, but is equivalent for both the EM and GCN algorithms (and these tasks are independent of the size of the robot's action space). The computation time of generating the exploration graph is not constant either, but consumes a small fraction of the overall computation time required for the GCN approach.

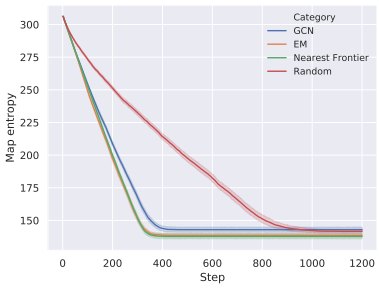
3.4 Scalability

For large-size maps with many landmarks, it is time-consuming to generate training data using the EM algorithm because of the poor scaling of its computational expense, as shown in Figure 4d. Our proposed method offers the additional prospect of *scalability in training*, by allowing a GCN trained on small maps to be queried over a large map. Larger maps accumulate a longer history of poses, more landmarks, and more frontiers, yielding larger exploration graphs, by which the state space and the action space expand as the map size increases. A robot is also likely to accumulate more map and pose estimation error when exploring larger maps. In this final experiment, we use a GCN trained on $40m \times 40m$ maps with 0.5 landmarks per m^2 feature density to explore larger maps which are $60m \times 60m$, $80m \times 80m$ and $100m \times 100m$ with the same feature density. Representative examples of GCN exploration over the three different map sizes are shown in Fig. 5, along with average entropy reduction over 50 trials across all competing algorithms.

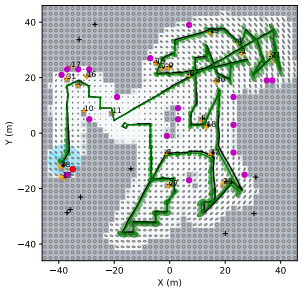
We show the map and pose uncertainty incurred by all four algorithms over different large-size maps in Fig. 6. The nearest frontier method has the worst performance over every map size. As the map size increases, the nearest frontier method accu-



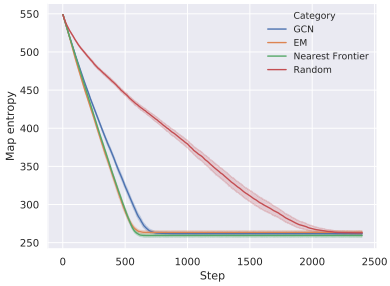
(a) GCN exploration on $60m \times 60m$ map



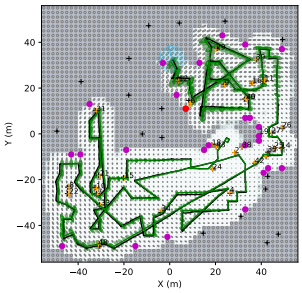
(b) Entropy reduction on $60m \times 60m$ maps



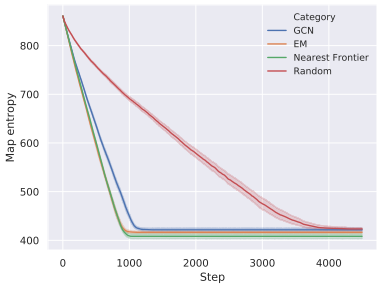
(c) GCN exploration on $80m \times 80m$ map



(d) Entropy reduction on $80m \times 80m$ maps



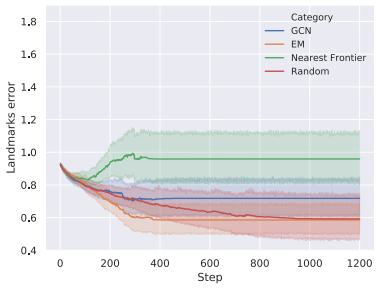
(e) GCN exploration on $100m \times 100m$ map



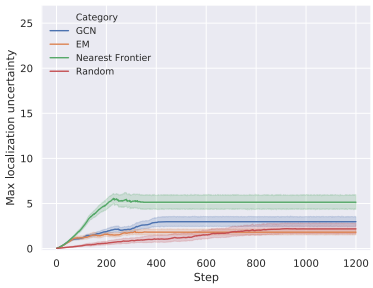
(f) Entropy reduction on $100m \times 100m$ maps

Fig. 5: (a)(c)(e): An illustration of GCN exploration over different map sizes. (b)(d)(f): The map entropy reduction results of 50 exploration trials on three different size maps. The GCN policy is trained on $40m \times 40m$ maps.

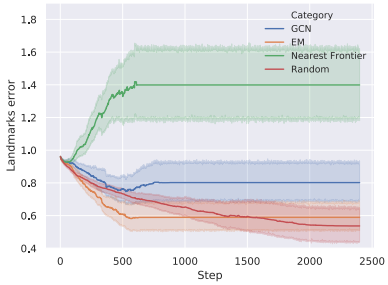
mulates more error during exploration. In contrast, the random method and the EM algorithm achieve similar performance even as the map size grows. However, although the random method and the EM algorithm have similar optimal performance with respect to curbing uncertainty, the random method produces the least efficient exploration trajectories as shown in Figs. 5b, 5d, and 5f. Here, the EM algorithm



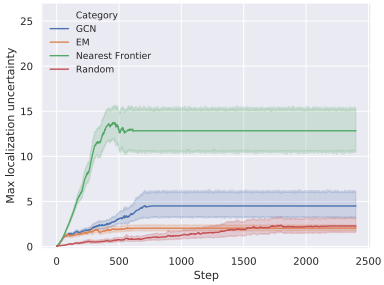
(a) Average uncertainty of landmarks on $60m \times 60m$ maps



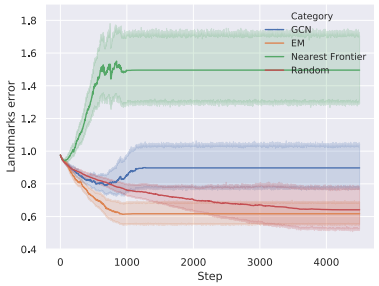
(b) Max uncertainty of the trajectory on $60m \times 60m$ maps



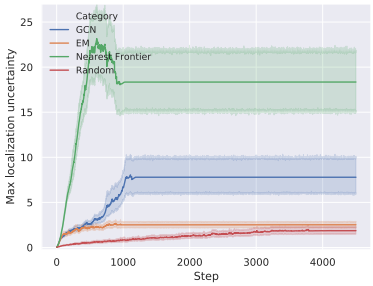
(c) Average uncertainty of landmarks on $80m \times 80m$ maps



(d) Max uncertainty of the trajectory on $80m \times 80m$ maps



(e) Average uncertainty of landmarks on $100m \times 100m$ maps



(f) Max uncertainty of the trajectory on $100m \times 100m$ maps

Fig. 6: Landmark and pose uncertainty management over 50 exploration trials on three different map sizes, with the same randomly initialized landmarks for every algorithm. The GCN policy is trained on $40m \times 40m$ maps.

is still an optimal method since it produces short trajectories with low-uncertainty landmark and pose estimates. The GCN approach is a near-optimal method compared with the EM algorithm. It still achieves low uncertainty of landmarks and

poses over the $60m \times 60m$ maps, but as the map size grows, the GCN framework gradually performs worse because the training data was generated from $40m \times 40m$ maps, and the larger maps' exploration graphs differ from those encountered in the training phase. However, the uncertainty of the landmark and pose estimates from the GCN approach is still lower than the nearest frontier method, although they have similarly competitive travel distance, as shown in Figs. 5b, 5d, and 5f. As we discussed in Section 3.3, the computation time of the EM algorithm is not real-time on these large maps, so the GCN framework offers a suitable alternative.

4 Conclusions and Future Work

In this paper, we proposed a novel graph learning-based exploration framework that uses a GCN to achieve real-time viable exploration decision-making, and provides an accurate and efficient exploration result. For other active-SLAM exploration algorithms, with increasing dimensionality of the robot's state space and action space, the procedure's computational complexity ceases to be real-time viable. The GCN framework estimates a robot's next optimal sensing action using an *exploration graph*, which is introduced to encapsulate the current state space and action space. Moreover, the proposed method is scalable to testing in higher-dimensional state-action spaces, even when trained in lower-dimensional state-action spaces. Looking ahead to future work, we wish to utilize a reduced graph structure to represent the current state-action space, ideally in which the constraints tied to a robot's pose history, which grows without bound, can be collapsed onto the landmarks only, which are finite in number. Moreover, in future work we will explore the viability of different graph learning models, and investigate the framework's extensibility to more complex real-world active SLAM scenarios.

Acknowledgements This research has been supported by the National Science Foundation, grant number IIS-1723996.

References

1. S. Thrun, W. Burgard and D. Fox. Exploration. *Probabilistic Robotics*, 569–605, MIT Press (2005)
2. B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael and V. Kumar. Information-theoretic Planning with Trajectory Optimization for Dense 3D Mapping. *Proceedings of Robotics: Science and Systems* (2015)
3. 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*, 4791–4798 (2015)
4. 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*, 33(10): 1375–1392 (2014)

5. M. G. Jadidi, J. V. Miró, R. Valencia and J. Andrade-Cetto. Exploration on Continuous Gaussian Process Frontier Maps. *Proceedings of the IEEE International Conference on Robotics and Automation*, 6077–6082 (2014)
6. M. G. Jadidi, J. V. Miró and G. Dissanayake. Mutual Information-based Exploration on Continuous Occupancy Maps. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 6086–6092 (2015)
7. 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*, 1885–1891 (2012)
8. C. Stachniss, G. Grisetti and W. Burgard. Information Gain-based Exploration using Rao-Blackwellized Particle Filters. *Proceedings of Robotics: Science and Systems*, 65–72 (2005)
9. J. Wang and B. Englot. Autonomous Exploration with Expectation-Maximization. *Proceedings of the International Symposium on Robotics Research* (2017)
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*, 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*, 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*, 2379–2384 (2017)
13. 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)
14. 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*, 0518(7540):529–533 (2015)
15. 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*, 4(2):610–617 (2019)
16. T. N. Kipf and M. Welling. Semi-supervised Classification with Graph Convolutional Networks. *Proceedings of the International Conference on Learning Representations* (2017)
17. W. Hamilton, Z. Ying and J. Leskovec. Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems*, 1024–1034 (2017)
18. D.K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik and R.P. Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. *Advances in Neural Information Processing Systems*, 2224–2232 (2015)
19. 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. *arXiv preprint*, arXiv:1806.01242 [cs.LG] (2018)
20. 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).
21. F. Dellaert. Factor Graphs and GTSAM: A Hands-on Introduction. *Technical Report, Georgia Institute of Technology*, GT-RIM-CP&R-2012-002 (2012).
22. 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*, 31(2):216–235 (2012)
23. M. Kaess and F. Dellaert. Covariance Recovery from a Square Root Information Matrix for Data Association. *Robotics and Autonomous Systems*, 57(12):1198–1210 (2009)
24. D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *Proceedings of the International Conference on Learning Representations* (2015)
25. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin and S. Ghemawat. Tensorflow: Large-scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint*, arXiv:1603.04467 [cs.DC] (2016)