applied sciences

MDPI

Article

Autonomous Exploration of Mobile Robots via Deep Reinforcement Learning Based on Spatiotemporal Information on Graph

Zhiwen Zhang , Chenghao Shi, Pengming Zhu, Zhiwen Zeng * and Hui Zhang

Robotics Research Center, College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China; zhangzhiwen@nudt.edu.cn (Z.Z.); shichenghao17@nudt.edu.cn (C.S.); zhupengming@nudt.edu.cn (P.Z.); zhanghui_nudt@126.com (H.Z.)
* Correspondence: zengzhiwen@nudt.edu.cn

Abstract: In this paper, we address the problem of autonomous exploration in unknown environments for ground mobile robots with deep reinforcement learning (DRL). To effectively explore unknown environments, we construct an exploration graph considering historical trajectories, frontier waypoints, landmarks, and obstacles. Meanwhile, to take full advantage of the spatiotemporal feature and historical information in the autonomous exploration task, we propose a novel network called Spatiotemporal Neural Network on Graph (Graph-STNN). Specifically, the proposed Graph-STNN extracts the spatial feature using graph convolutional network (GCN) and the temporal feature using temporal convolutional network (TCN). Then, gated recurrent unit (GRU) is performed to synthesize the spatial feature, the temporal feature, and the historical state information into the current state feature. Combined with DRL, our Graph-STNN helps estimation of the optimal target point through extracted hybrid features. The simulation experiment shows that our approach is more effective than the GCN-based approach and the information entropy-based approach. Moreover, Graph-STNN also performs better generalization ability than GCN-based, information entropy-based, and random methods. Finally, we validate our approach on the simulation platform Stage with the actual robot model.

Keywords: autonomous exploration; deep reinforcement learning; spatiotemporal information; graph convolutional network; temporal convolutional network; gated recurrent unit; robots

Citation: Zhang, Z.; Shi, C.; Zhu, P.; Zeng, Z.; Zhang, H. Autonomous Exploration of Mobile Robots via Deep Reinforcement Learning Based on Spatiotemporal Information on Graph. *Appl. Sci.* **2021**, *11*, 8299. https://doi.org/10.3390/app11188299

Academic Editor: Marina Paolanti

Received: 22 July 2021
Accepted: 1 September 2021
Published: 7 September 2021

1. Introduction

The prior information about the environment (usually a map) is always helpful for robots performing specific tasks, e.g., service, security, rescue, etc. To explore the unknown environments, the manually remote operation of the mobile robots to gather information is probably not a good option due to the possible poor communication link [1]. On the contrary, an autonomous scheme by selecting a target point to navigate in the unknown region is more popular. By repetitively selecting and navigating to the candidate point, the robot would cover the whole region [2,3]. However, this method requires expert human experience to choose the suitable point and thus is less adaptable to different situations. Moreover, with the increasing dimensionality of the state space and the action space, the computational burden of the multi-objective optimization problem will increase rapidly. Besides, these methods do not take the localization uncertainty into account during the exploration process, which may eventually lead to an inaccurate map [4].

There are also some deep learning-based methods. Typically, neural networks are used to map the sensor data or the global occupancy maps built by Simultaneous Localization and Mapping (SLAM) into the actions of robots [5–7]. However, such frameworks require significant computational resources and may take a longer time to converge during the training phase, even fail to converge. Some works aim to combine the advantages of

human experts and deep learning. For instance, the deep learning methods blend the learning scheme with the traditional planning methods helps to obtain the optional target point [8–10]. The common problem in learning-based methods is poor generalizability in new unknown surroundings [8].

Recently, Chen et al. [4] firstly represented the environment as a graph and proposed a new method to autonomously navigate around the undetected area based on graph neural networks (GNNs). Compared with raw data from sensors or the real-time map, the graph reduces the computational effort and better captures the structural information about the environment. In addition, due to the dynamically changed exploration process, the hyperparameters of end-to-end deep learning are hard to tune. More recently, Chen et al. [8] further designed a generalized exploration framework combining the GNNs with deep reinforcement learning (DRL) to learn a more robust policy. GNNs are used to enable network reasoning about the surroundings, and the extracted feature is then used to help DRL policy determine.

Although GNNs combined with DRL show great potential for solving the autonomous exploration problem for mobile robots, there are still some problems. The above-mentioned works did not consider obstacle avoidance, which may cause serious damage. Additionally, the characteristics of the continuous state transition and the incrementally constructed exploration graph are underutilized. Moreover, to promote the effectiveness of the exploration process, the information of the interaction structure and the historical trajectory of the mobile robot are supposed to be fully utilized. To address the limitations of the traditional methods, the learning-based methods, and the GCN-based methods, we propose a novel approach combined with graph neural networks and DRL with the aim of improving the effectiveness, robustness, and generalizability.

The main contributions of this paper are as follows: First, more general cases with obstacle avoidance are considered by embedding the obstacle information into the exploration graph. Second, to the best of our knowledge, the spatiotemporal information is the first time to introduce to the autonomous exploration problem, which helps improve the efficiency of exploration by learning the potential graph representation on incremental graphs. Third, the state encoder that comprehensively utilizes the spatial, temporal, and historical state features is proposed to extract global information from the continuous changes of graphs to improve the accuracy and robustness. Fourth, in DRL, we consider both separately state features and action features to improve the converge performance. The source code of Graph-STNN is publicly available from Github (https://github.com/zhangzw-nudt/Graph-STNN_Exploration (accessed on 22 July 2021)).

In sum, we make four key claims: Our approach (i) chooses more reasonable target points by taking into obstacle information; (ii) can better adapt to the exploration problem using temporal and spatial information and lead to more accurate exploration; (iii) has better robustness and generalizability to new environments against state-of-the-art approaches; (iv) converges faster and obtains more rewards. These claims are backed up by the paper and our experimental evaluation.

The rest of the paper is organized as follows. Section 2 reviews the work related to our approach. Section 3 introduces the details of our approach. Section 4 evaluates the performance of our approach in the 2D simulation environment. Finally, we conclude the paper in Section 5.

2. Related Work

Existing robotic autonomous exploration methods can be divided into two categories: the traditional approaches and the machine learning-based approaches.

The standard pipeline for traditional approaches is based on a frontier-based exploration strategy proposed by Yamauchi et al. [11]. They find the frontier, the junction of known open areas and unknown areas on the map. The points on the frontier are served as frontier waypoints, from which the robot selects the best one as a local target and navigates to the target point autonomously by path planning. For a frontier-based exploration

strategy, many approaches mainly focus on environmental uncertainty evaluation and optimal target point selection. Makarenko et al. [12] proposed a utility function considering information gain, distance cost, and covariance matrix-based localization effect, and use the function to evaluate the candidate points of the robot. Kaufman et al. [13,14] proposed to compute the expected entropy gain of an occupancy grid map and further applied it to select the target points. Carrillo et al. [15] proposed Shannon entropy for map uncertainty and Rényi entropy for robot pose uncertainty to estimate the cost of candidate points. Bai et al. [16] assessed the candidate points using mutual information obtained by a Bayesian optimization algorithm, which improved computational efficiency and significantly reduced the map entropy. The frontier waypoints, however, are not the only choice. Gonzalez et al. [17] proposed to use the points with the most observation information as the local goal. However, it results in a larger search space, which increases the computational burden. Traditional approaches have achieved positive results in some specific scenarios. However, the common problem of such approaches is hard to balance exploration effectiveness and computational burden as the exploration area increases. Moreover, these methods highly depend on human-designed experts' experience. While the environment is changing, the algorithm needs to be redesigned.

The learning-based approaches employ machine learning to replace the expert algorithm for target selection. Li et al. [9] measured the quality of mapping using Shannon entropy and defined the action space on a grid map. The candidates in the action space are evaluated by DRL using a fully convolutional Q-network. Niroui et al. [10] combined traditional boundary-based exploration methods with DRL to choose the best target points. Some other learning frameworks focus on learning the exploration actions in an end-to-end fashion. For instance, Tai et al. [5] employed CNN-based DRL to learn the exploration strategy, which is obtained end-to-end from RGBD camera data. Furthermore, Bai et al. [6] directly used a neural network to learn the mapping from the maps created during exploration to actions. Tai et al. [7] used the distances in ten directions, the targets, and previous actions as the inputs to obtain the robot's velocities. Usually, these learning-based methods may afford high calculation burdens and cannot be generalized to new environments with different obstacles, scales, etc.

Recently, GNN shows powerful capabilities in solving problems in non-Euclidean spaces [18]. Since the exploration problem is closely related to the spatial structure of the environment, Chen et al. [8] firstly proposed a generalized graph representation for mobile robot exploration and solved the exploration problem by using Deep Q-Learning (DQN) [19] with graph convolutional network (GCN) [20] as the policy and/or value networks. In addition, recurrent neural networks (RNNs) have achieved great success in processing sequential data [21], such as trajectory prediction [22,23] and speaker identification [24]. Inspired by them, based on the exploration graph constructed by a more general environment with obstacles, we propose a novel network to extract spatiotemporal information to generate a more effective exploration strategy.

3. Our Approach

We propose a novel Graph-STNN incorporated with DRL to address the autonomous exploration of mobile robots. The overall framework of our approach is shown in Figure 1. In the beginning, we calculate the frontier waypoints and take the optimal one as the candidate. Then, the robot updates the observed environment while moving to the targets by path planning until the whole map is covered.

The problem is first formulated in Section 3.1. Then we model the environment as an exploration graph in Section 3.2. In Section 3.3, we detail how the novel Graph-STNN generates the best target point. Next, Graph-STNN performs as a policy network trained with DRL, as described in Section 3.4.

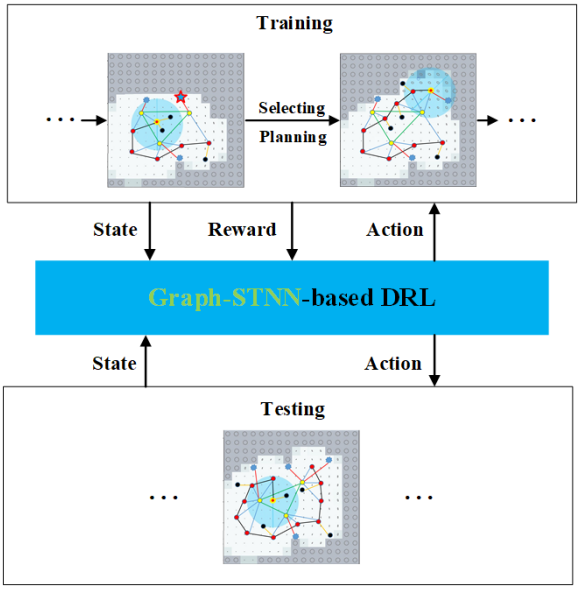


Figure 1. The framework of our approach. The grid environment is modeled as an exploration graph consisting of historical trajectories, frontier waypoints, landmarks, and obstacles shown in red, blue, yellow, and black. The red pentagram represents the currently selected target point, and the exploration graph represents the current state. In the training phase, the robot optimizes the network parameters to maximize the desired reward of the whole exploration progress. Then in the testing phase, our approach evaluates the frontier waypoints and selects the optimal one as the target point.

3.1. Problem Formulation

The exploration environments are two-dimensional grid maps, in which each historical trajectory, frontier waypoint, landmark, and obstacle corresponds to a grid. Then, four-type grids serve as the nodes in the exploration graph described in Section 3.2. Inspired by reference [8], to represent the uncertainty and occupancy probability of every grid, we construct the raster virtual map by Expectation-Maximization (EM) exploration algorithm [25], where each grid represents a virtual landmark with an initial identical error covariance matrix. Meanwhile, A-optimality metric [26] is used to measure the uncertainty of each virtual landmark v_i. Then, the uncertainty U of the whole map is calculated by:

$$U = \sum_{v_i \in V} \mathrm{tr}(\Sigma_{v_i}), \tag{1}$$

where V represents the set of all virtual landmarks, and $\mathrm{tr}(\Sigma_{v_i})$ represents the trace of each virtual landmark's error covariance matrix Σ_{v_i}.

To measure the robot exploration speed, we take the convergence speed. Specifically, we average the exploration rate for 50 tests at each moment and plot the average exploration rate over time.

In addition, to evaluate the exploration algorithm, we define the following exploration efficiency:

$$\eta = \frac{\Delta U}{T}, \tag{2}$$

where ΔU is the uncertainty reduction in time T. Therefore, η is used to measure the reduction of map uncertainty per unit time.

Finally, to compare the mapping accuracy of different methods, we define the following trajectory error:

$$e = \frac{n_a}{n_g} \sqrt{\max_{v_i \in V_P} tr(\Sigma_{v_i})}, \tag{3}$$

where, n_a is the total number of exploration steps, n_g is the total number of grids in the map, and V_P is the trajectory nodes. Furthermore, similarly, we take the mean value of the trajectory error for 50 identical test scenarios.

Our approach adopts the frontier-based exploration scheme with path planning. The exploration problem can be viewed as a sequential decision-making problem, whereby the robot learns to act optimally from changes to the environment that result from our selection of frontier nodes. Thus, our goal is to choose the best target point to maximize η, meanwhile, minimize U and e.

3.2. Exploration Graph

The observed information changes incrementally during the exploration progress, leading to the increasing dimension of state and action spaces. Compared with learning from occupancy map or sensor data, the graph offers a smaller state space, a more generalized representation of a SLAM-dependent mobile robot's environment [4]. Moreover, the graph-learning approach is more adaptable to dynamic environments, in which the input is variable-size graphs, and the size of the output data adjusts to the number of nodes of the input graph.

For these reasons, we represent the graph topology of the exploration problem and model the exploration process as a generalized exploration graph, as shown in Figure 2. We denote the exploration graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes and $\mathcal{E}$ the set of edges.

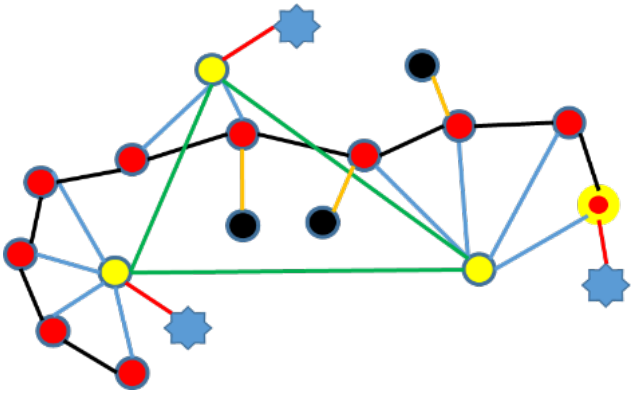


Figure 2. Extracting the exploration graph. The historical trajectories, frontier waypoints, landmarks, and obstacles are shown in red, blue, yellow, and black, respectively. The current position in trajectories is stressed with the yellow border , and the frontier waypoints are served as the candidate target points.

There are four types of nodes in $\mathcal{V}$: historical trajectories (red), frontier waypoints (blue), landmarks (yellow), and obstacles (black), where the obstacle node is firstly introduced in this paper, and the frontier waypoint nodes are performed as the candidate points. The edges $\mathcal{E}$ between the nodes are constructed as follows: (i) adjacent trajectory nodes are connected to each other, (ii) each trajectory is connected to the nearest landmark, (iii) each frontier waypoint is connected to the nearest landmark or the current position of the robot, (iv) the landmarks are fully connected, (v) each obstacle is connected to the nearest trajectory. After establishing the exploration graph, the robot can obtain the local target point to guide the exploration of unknown areas from candidates by network reasoning about the information on the graph.

As a start point of node feature aggregation, we initialize the node feature as:

$$\mathbf{x}_i = [d, \varphi, H_\gamma, H_\beta, \phi], \tag{4}$$

where d is the distance of the node and the current position of robot, φ the angle between the heading of robot and direction from the robot to the node, H_γ the Shannon entropy, H_β

the Renyi entropy, ϕ the flag. The correspondence between flag ϕ and nodes are shown in Table 1.

Table 1. The correspondence between flag ϕ and nodes.

ϕ	Nodes
2	frontier waypoint
1	landmark
0	robot's current position
−1	historical trajectory
−2	obstacle

The Shannon entropy and Renyi entropy [15] are used to measure the grid uncertainty and the pose uncertainty, respectively. The Shannon entropy is defined as:

$$H_\gamma \approx -\sum_{ij}(P(m_{ij})\log(P(m_{ij})) + (1-P(m_{ij}))\log(1-P(m_{ij}))), \tag{5}$$

where m_{ij} is the Bernoulli random variable. $m_{ij} = 0$ indicates that the grid is free, and $m_{ij} = 1$ means grid occupied. $P(m_{ij} = 1)$ is the probability of grid being occupied, which is abbreviated as $P(m_{ij})$. The Renyi entropy is defined as:

$$\begin{aligned} H_\beta &= \frac{1}{1-\beta}\log_2(\sum_{i=1}^{n} P_i{}^\beta) \\ &= \frac{1}{1-\beta}\log_2(P(m_{ij})^\beta + (1-P(m_{ij}))^\beta), \end{aligned} \tag{6}$$

where $\beta = 1 + \frac{1}{tr(\Sigma_{g_{ij}})}$, in which $\Sigma_{g_{ij}}$ denotes the error covariance matrix of the grid g_{ij} and P_i the probability associated with the random variable m_{ij}.

Then, we use a multi-layer perceptron (MLP) [27,28] to map the initial feature into a high dimensional vector as the input of the following network.

$$\mathbf{F} = \mathrm{MLP}(\mathbf{X}), \tag{7}$$

where $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^\mathsf{T}$, $\mathrm{MLP}(\cdot)$ is a multi-layer perceptron, and N the number of the nodes.

3.3. Graph-STNN

In this section, we propose a novel Graph-STNN, which extracts multiple features from the exploration graph to help choose the best frontier waypoints for robots to map the unknown environment effectively. The overview of our approach is shown in Figure 3, in which three different encoders are designed to extract information from the incremental exploration graph. Specifically, we design a spatial encoder to extract spatial features from the neighbor nodes and a temporal encoder to extract the temporal features from the historical robot trajectories. Besides, we design a state encoder to embed the spatiotemporal information and the historical state into the current state feature.

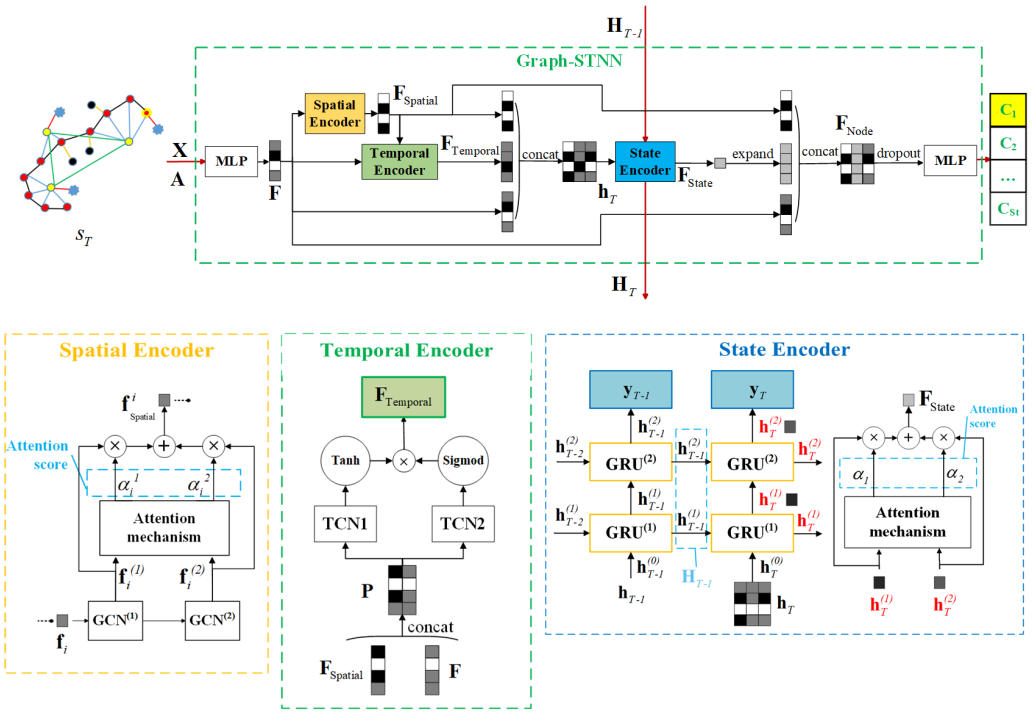


Figure 3. The structure of Spatiotemporal Neural Network on Graph (Graph-STNN). The current state is represented as the exploration graph s_T. All frontier waypoints can be represented as $C_1 \sim C_{st}$, in which st is the number of frontier waypoints in the current state. The inputs of the network are an adjacency matrix $\mathbf{A}$ and an original node feature matrix $\mathbf{X}$ of the exploration graph. First, the original features encoded by a multi-layer perceptron (MLP), the spatial features extracted by the spatial encoder, and the temporal features extracted by the temporal encoder are concatenated together. Then, the concatenated feature is input to the state encoder to obtain the state feature, which is the same for each node. Finally, the state feature combined with the original features and spatial features is input to a liner layer after dropout, and the output is the value of each frontier waypoint. The frontier waypoint with the largest value is the local target point.

3.3.1. Spatial Encoder

The spatial encoder is designed to encode the structural information around the frontier waypoints. As GCN has achieved great success in graph structure data processing, we use GCN to deal with the spatial interaction. Specifically, we use a two-layer GCN to construct the spatial encoder. We denote the l-layer node features as $\mathbf{F}^{(l)} = [\mathbf{f}_1^{(l)}, \cdots, \mathbf{f}_N^{(l)}]^\mathsf{T}$, the node aggregation at layer l is then calculated as:

$$\mathbf{F}^{(l+1)} = \sigma_s(\mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}\mathbf{F}^{(l)}\mathbf{W}_S^{(l)}), \tag{8}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\mathbf{I}$ is the unit matrix, $\mathbf{A}$ the adjacency matrix of exploration graph, $\mathbf{D}$ the degree matrix, $\mathbf{W}_S^{(l)}$ the learnable weight matrix of layer l, and $\sigma_s(\cdot)$ the ReLU function. The input of the encoder is $\mathbf{F}^{(0)} = \mathbf{F} = [\mathbf{f}_1, \cdots, \mathbf{f}_N]^\mathsf{T}$. Then, we use a two-layer GCN. The features of adjacent layers are combined by using the attention mechanism. The output spatial feature of node i can be represented as:

$$\mathbf{f}_{\text{Spatial}}^i = \alpha_i^1 * \mathbf{f}_i^{(1)} + \alpha_i^2 * \mathbf{f}_i^{(2)}, \tag{9}$$

where

$$\alpha_i^l = \frac{\exp(\mathbf{W}_1\mathbf{f}_i^{(l)} + \mathbf{b}_1)}{\sum_{l=1}^{2}\exp(\mathbf{W}_1\mathbf{f}_i^{(l)} + \mathbf{b}_1)}, \tag{10}$$

$\exp(\cdot)$ is the expedition function, and $\mathbf{W}_1$ and $\mathbf{b}_1$ are the parameters of attention mechanism. The spatial feature matrix is $\mathbf{F}_{\text{Spatial}} = [\mathbf{f}^1_{\text{Spatial}}, \cdots, \mathbf{f}^N_{\text{Spatial}}]^{\mathsf{T}}$, which consists of all nodes' spatial feature.

3.3.2. Temporal Encoder

The temporal encoder is designed to encode the temporal information of the exploration process from the position change of the robot in the map. Thus, we only consider the historical trajectory nodes in incremental graphs.

For the temporal encoder, we use a gated temporal convolutional network (gated TCN) [29]. Different from the original temporal convolutional network (TCN) [30], gated TCN uses a gating mechanism to improve the ability of information extraction. Compared with recurrent neural networks (RNNs) [21], gated TCN deals with parallel input, which reduces memory expense and amounts of parameters. Therefore, our temporal encoder consists of two same TCNs with different gates, i.e., Tanh and Sigmod function. We concatenate the initial features with the spatial features as:

$$\mathbf{P}^{'} = \text{concat}(\mathbf{F}, \mathbf{F}_{\text{Spatial}}). \tag{11}$$

The input of our temporal encoder is $\mathbf{P}$, which consists of the trajectory nodes' features in $\mathbf{P}^{'}$. We denote the t-th node in $\mathbf{P}$ as $\mathbf{p}_t$. The update of $\mathbf{p}_t^{(l)}$ at layer l in TCN is represented as:

$$\mathbf{p}_t^{(l+1)} = \mathcal{F}_{d_l}(\mathbf{P}^{(l)})|_{\mathbf{p}_t^{(l)}} = \sum_{k=1}^{K} f_k \mathbf{p}^{(l)}_{t-(K-k)d_l}, \tag{12}$$

where K is the one-dimensional convolution kernel size, d_l the expansion factor controlling the jump connection of layer l, $f_1 \sim f_K$ the learnable parameters of convolution kernel. Denote the output of these two TCNs as $\mathbf{P}_1$ and $\mathbf{P}_2$, respectively. The final temporal feature is calculated as:

$$\mathbf{F}_{\text{Temporal}} = \sigma_h(\mathbf{P}_1) \odot \sigma_g(\mathbf{P}_2), \tag{13}$$

where σ_h is Tanh function, σ_g Sigmod function, and $\odot$ the element-wise product.

3.3.3. State Encoder

Although the spatiotemporal information can be obtained, we still expect to obtain the global information to effectively accomplish the exploration task. Hence, a state encoder is proposed to further aggregate the global and historical environment information.

A similar idea of separating considerations of state and action is recently proposed by Wang et al. [31]. They use two MLPs at the end of the feature aggregation to extract the state value and the state-action value, respectively. Different from them, we explicitly divide the node feature into a state feature and an action feature. Notice that the state feature is independent of the robot's actions.

We construct the state encoder based on gated recurrent unit (GRU) [32].

We concatenate the initial feature, the spatial feature, and the temporal feature together as the input of the state encoder.

$$\mathbf{h}_T = \text{concat}(\mathbf{F}, \mathbf{F}_{\text{Spatial}}, \mathbf{F}_{\text{Temporal}}). \tag{14}$$

We then use a two-layer GRU to extract the state feature. The structure of GRU is shown in Figure 4, and the convolution process of layer l in the GRU is calculated by:

$$\mathbf{h}_T^{(l)} = \mathbf{z}_T^{(l)} \odot \mathbf{h}_{T-1}^{(l)} + (\mathbf{1} - \mathbf{z}_T^{(l)}) \odot \mathbf{h}_T^{'(l)}, \tag{15}$$

where $\mathbf{z}_T^{(l)}$ is the update gate defined as:

$$\mathbf{z}_T^{(l)} = \sigma_g(\mathbf{W}_z^{(l)}\mathbf{h}_T^{(l-1)} + \mathbf{U}_z^{(l)}\mathbf{h}_{T-1}^{(l)}), \tag{16}$$

and $\mathbf{h}_T^{'(l)}$ is the current memory information defined as:

$$\mathbf{h}_T^{'(l)} = \sigma_h(\mathbf{W}_h^{(l)}\mathbf{h}_T^{(l-1)} + \mathbf{r}_T^{(l)} \odot \mathbf{U}_h^{(l)}\mathbf{h}_{T-1}^{(l)}), \tag{17}$$

and $\mathbf{r}_T^{(l)}$ is the reset gate defined as:

$$\mathbf{r}_T^{(l)} = \sigma_g(\mathbf{W}_r^{(l)}\mathbf{h}_T^{(l-1)} + \mathbf{U}_r^{(l)}\mathbf{h}_{T-1}^{(l)}), \tag{18}$$

where $\odot$ is the element-wise product, $\mathbf{h}_{T-1}^{(l)}$ the hidden vector passed down from the layer l, $\mathbf{h}_T^{(l)}$ the hidden vector passing to the next layer and the output, $\mathbf{h}_T^{(0)} = \mathbf{h}_T$, and $\mathbf{W}_z^{(l)}, \mathbf{U}_z^{(l)}, \mathbf{W}_r^{(l)}, \mathbf{U}_r^{(l)}, \mathbf{W}_h^{(l)}, \mathbf{U}_h^{(l)}$ the learnable parameters in layer l of GRU. Three components work together and construct the state feature. The update gate passes the information from the previous step as well as the valuable information from the current input to the future. The reset gate determines the part of the previous message to be retained based on the current input and the output of the previous step. The current memory information consists of the current input information and the part of the previous information to be retained as determined by the reset gate.

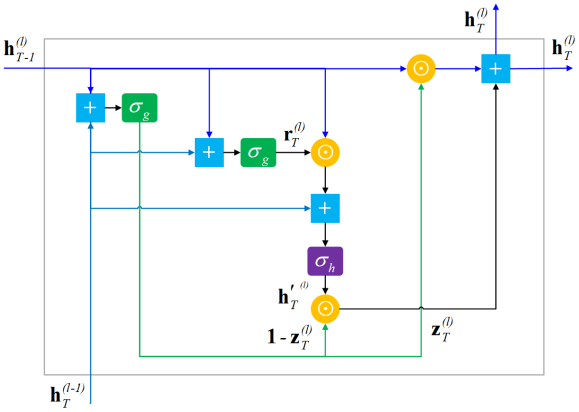


Figure 4. The structure of gated recurrent units (GRU). It consists of a reset gate and an update gate.

Similar to the spatial encoder, we also use an additional attention layer to further aggregate the two layers' output to the final vector, which can be represented by:

$$\mathbf{F}_{\text{State}} = \alpha_1 * \mathbf{h}_T^{(1)} + \alpha_2 * \mathbf{h}_T^{(2)}, \tag{19}$$

where

$$\alpha_l = \frac{\exp(\mathbf{W}_2\mathbf{h}_T^{(l)} + \mathbf{b}_2)}{\sum_{l=1}^{2}\exp(\mathbf{W}_2\mathbf{h}_T^{(l)} + \mathbf{b}_2)}, \tag{20}$$

$\mathbf{h}_T$ and $\mathbf{H}_{T-1} = [\mathbf{h}_{T-1}^{(1)}, \mathbf{h}_{T-1}^{(2)}]$ are the input of our state encoder in current time T; $\mathbf{h}_T^{(l)}$ is the output of the layer l of GRU at T, $\mathbf{W}_2$ and $\mathbf{b}_2$ are the learnable parameters of the attention mechanism.

To reduce the risk of overfitting and improve the exploration ability, we perform dropout to each node feature, concatenated by state feature, spatial features, and temporal feature. The dropout rate decreases gradually with the number of exploration steps. Once the node feature is calculated, the final values of each candidate target point are then

determined by an additional MLP. The candidate point with the most significant value is selected as the target point.

3.4. Policy Training with DRL

The proposed Graph-STNN is performed as the value network, which helps to estimate optimal action through combining NoisyNet DQN [33]. Moreover, a multi-step strategy is used to estimate the value function by taking longer trajectories that contain more observations of multi steps' rewards, which can be represented as:

$$Q(s_t, a_t, \theta, \varsigma) = \sum_{k=0}^{m-1} \gamma^k r_{t+k} + \gamma^m \mathrm{Q}(s_{t+m}, a_{t+m}, \theta, \varsigma), \tag{21}$$

where $Q(s_t, a_t, \theta, \varsigma)$ represents the value of taking an action a_t in the current state s_t; ς represents the Gaussian noise added in the final MLP; γ is the discount factor; m is the step to future observation, which is an adjustable parameter. θ is the learnable parameter, which is soft updated as:

$$\theta = \theta^- + \alpha \frac{\partial L}{\partial \theta}|_{\theta=\theta^-}, \tag{22}$$

where L is the loss function, α the soft update coefficient.

The state transition process in DRL is shown in Figure 5, in which the exploration sequence consists of series of exploration graphs. Besides, each exploration graph represents a state, consisting of the adjacency matrix and the feature matrix. In each step, the robot selects the best frontier waypoint as the local target according to the evaluation of Graph-STNN. By using A* algorithm [34], the robot is able to plan a path and navigate to the target. Then, the exploration graph is refreshed as well as the state is updating to s_{t+1} with the environment reward r_t. The robot repeats this process until finishing exploring the whole environment. Eventually, the goal is to maximize the total rewards obtained after the iteration.

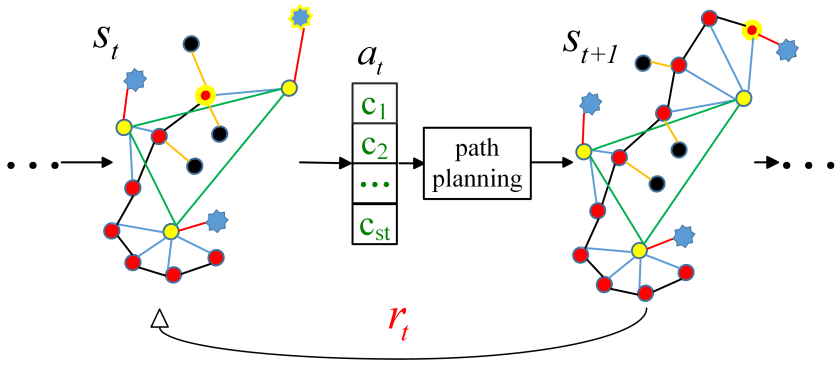


Figure 5. Extracting the exploration sequence. The exploration sequence consists of a series of exploration graphs, and each exploration graph represents a state. In the current state s_t, if the robot takes the action a_t to the next state s_{t+1}, it will obtain the immediate reward r_t.

Since state transition in each episode is sequentially related in our approach, the number of steps to future observation is the total step of the whole exploration progress. Thus, we update the network once after each episode, and the policy network is trained with the loss function as follows:

$$L = \frac{\sum_{i=1}^{T-1} \frac{1}{2} [\max_{a_i} Q(s_i, a_i, \theta, \varsigma) - \sum_{j=i}^{T-1} \gamma^{j-i} r_j]^2}{T-1}, \tag{23}$$

where T denotes the number of states for the entire training episode.

The environment reward is used to evaluate the action taken. The larger rewards obtained from the environment, the stronger tendency of the robot to adopt the strategy in the future. Furthermore, the ultimate goal of the robot is to maximize the desired reward of the whole exploration progress. Thus, to guide the selection of target points that can encourage efficient exploration, the reward function is defined as:

$$R = -\omega_1 * R_1 - \omega_2 * R_2 - \omega_3 * R_3 - \omega_4 * R_4 - \omega_5 * R_5 + \omega_6 * R_6, \quad (24)$$

where $R_1 = \Delta U$ is the variation of the whole map's uncertainty calculated by Equation (1) from the current state to the next state; R_2 and R_3 are the robot's angle change and the path length, according to the local path plan result by using A* algorithm; $R_4 = e_{\text{next}}$ is the maximum trajectory error calculated by Equation (3) after changing to the next state; R_5 is the total number of exploration steps from the beginning to the next state; R_6 is the environmental exploration rate (the ratio of the known area to the entire map area); $\omega_1 \sim \omega_6$ are constants larger than zero.

4. Experimental Evaluation

In this section, we use a graph-based SLAM method called GTSAM [35] to build the map of the environment during exploration.

Our experiments are designed to support our claims that our approach (i) chooses more reasonable target points by taking into obstacle information; (ii) can better adapt to the exploration problem using temporal and spatial information and lead to more accurate exploration; (iii) has better robustness and generalizability to new environments against state-of-the-art approaches; (iv) converges faster and obtains more rewards.

4.1. Experimental Setup

The exploration environment is a two-dimensional grid map with a random distribution of landmarks (number proportional to the map area) and obstacles (number proportional to the map edge length). It is assumed that the landmarks can be recognized once they enter the detection range of the robot. In addition, the horizontal view of the robot is set to 360 degrees, and the robot measurement range is set to 5 m. Assume the robot is able to rotate from −180 degrees to 180 degrees, and the noise in the simulation is Gaussian noise. The grid size is 2 m × 2 m, and the initial occupancy probability of each grid is 0.5. In this setting, the density of landmarks in the map is 3 landmarks per 100 grids.

Each time a new environment is reset, the position of landmarks and obstacles, and the initial position of the robot in the map are generated randomly. The robot has to avoid obstacles and landmarks while traveling to the target. First, we select the five points closest to the robot and the three points closest to each landmark from frontier waypoints as candidates. Then, the robot evaluates the candidates by our algorithm and selects the best one among them. Then the robot uses the A* algorithm to plan a collision-free path to achieve the goal. The exploration is finished while 90% of the area is checked by the robot. It is important to note that the models with obstacle avoidance and path planning module are closer to the actual situation.

An example exploration environment is shown in Figure 6, in which the frontier waypoints serve as candidate target points. The robot continually changes the local target and navigates to the local destination while updating the observed environment information until covering the whole map. The size of the map is 20 m × 20 m for training and 20 m × 20 m, 30 m × 30 m, and 40 m × 40 m for testing.

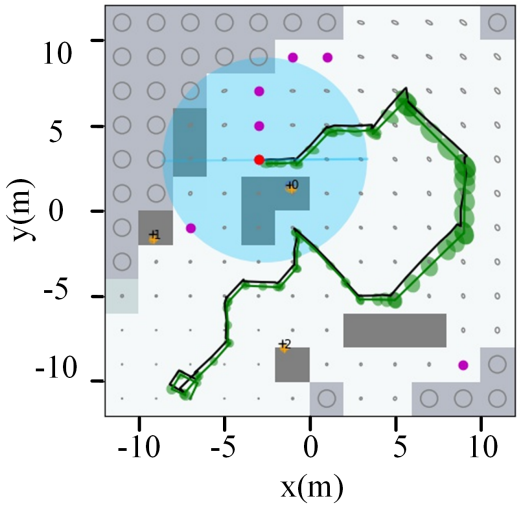


Figure 6. An example of our simulation environment, in which the exploration range is 20 m × 20 m, the light blue circular area the robot detection range, the light gray grid the unknown area, the gray grid the obstacle grid, the ellipse in the grid the virtual landmark uncertainty, the black plus the actual position of the landmark in the map, the yellow plus the estimated landmark position by the robot, the black trajectory the robot's actual trajectory, the green trajectory the robot's estimated trajectory, the green ellipse the robot's pose uncertainty, the purple point the candidate frontier waypoints (action space), and the red point the selected frontier waypoint (action).

4.2. Results and Analysis

In Graph-STNN, the first MLP only maps the initial feature into high-dimensional space, so we use one fully connected layer for reducing computational complexity. Similarly, the output MLP maps the hybrid features into the evaluation of each frontier. It is constructed from one linear layer with Gaussian noise (adding uncertainty to training and improving training efficiency) and one fully connected layer. In recurrent neural networks, the addition layer increases the level of abstraction of input observations over time [36]. However, the increase in layers will lead to an exponential growth in time and memory overhead and a sharp decline in convergence effect and efficiency. Therefore, combined with our actual computing capacity, we design the state encoder as a two-layer GRU. Finally, to determine the structure of the spatial encoder, we test one to three layers of GCN, and the training performance is shown in Figure 7. The test results are shown in Figure 8. As we see, Graph-STNN with a two-layer GCN has the best performance with the fast convergence, the minimum trajectory error, and the highest efficiency.

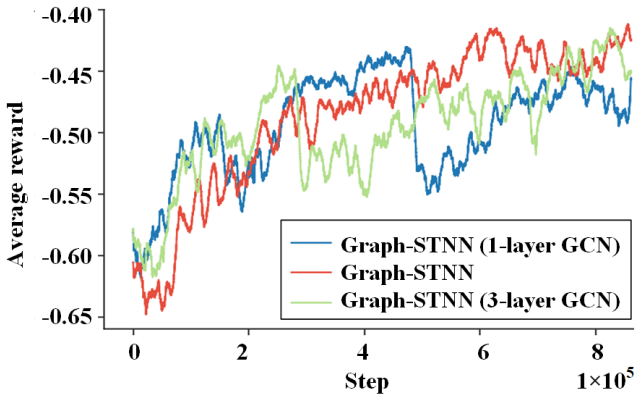


Figure 7. The training performance of Graph-STNN with different-layer GCN in the spatial encoder in our setting with the size of 20 m × 20 m.

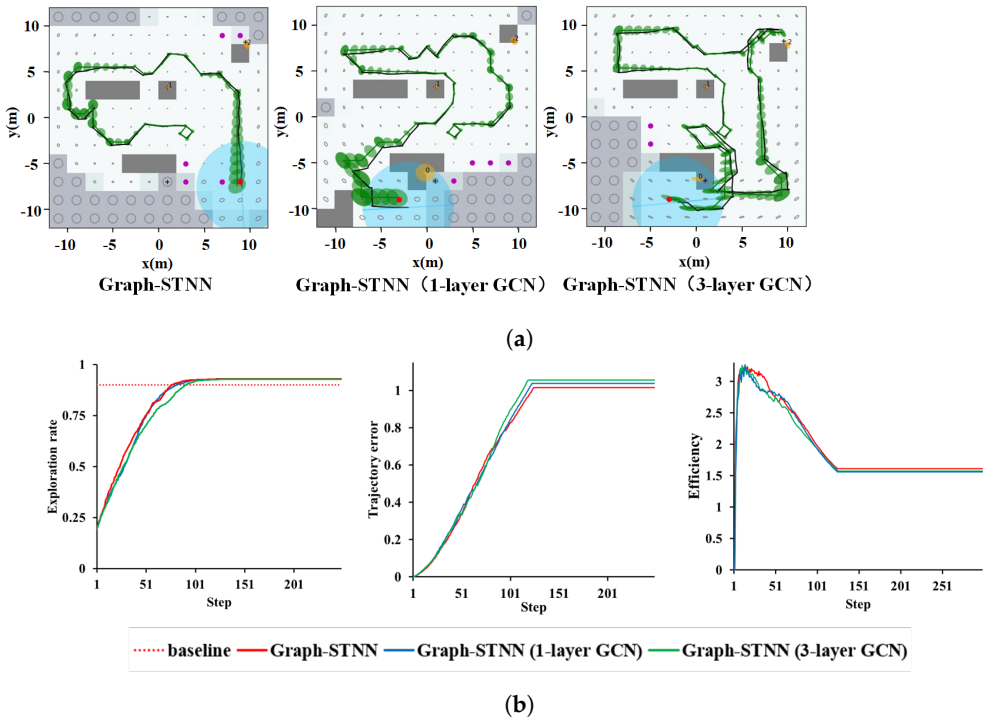


Figure 8. The test results of Graph-STNN with different-layer GCN in the spatial encoder of 50 exploration trials, with the same randomly initialized landmarks and robot start locations, on 20 m × 20 m maps (the results shown are plotted per time-step of the simulation). (**a**) One of the test result of 50 exploration trials. (**b**) Average exploration rate (left), trajectory error (middle), and exploration efficiency (right).

The entropy-based method [15] and the uncertainty-based methods [25] are the most popular traditional methods in this field, which perform well in autonomous exploration. Very recently, the learning-based method proposed in reference [8] achieved great progress on exploration performance, which is also inspired us. Therefore, to verify our approach, we compare our methods with them. Specifically, we train the GCN network proposed in reference [8] in the same experimental setting (with obstacles) as a comparison, which is denoted as GCN-obs for shorting. In addition, we also test the original GCN network of [8] in our experimental environment, which is trained in the environment without obstacles and denoted as GCN-no_obs. The following figure shows the convergence of rewards between Graph-STNN and the GCN-obs under the same reward function settings. The rewards are averaged every ten thousand steps. As shown in Figure 9, the proposed Graph-STNN is able to converge more quickly.

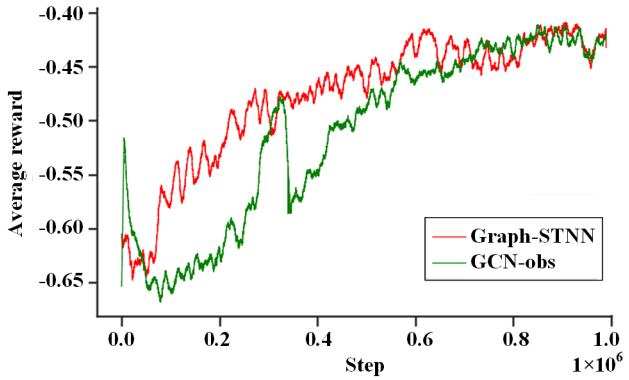
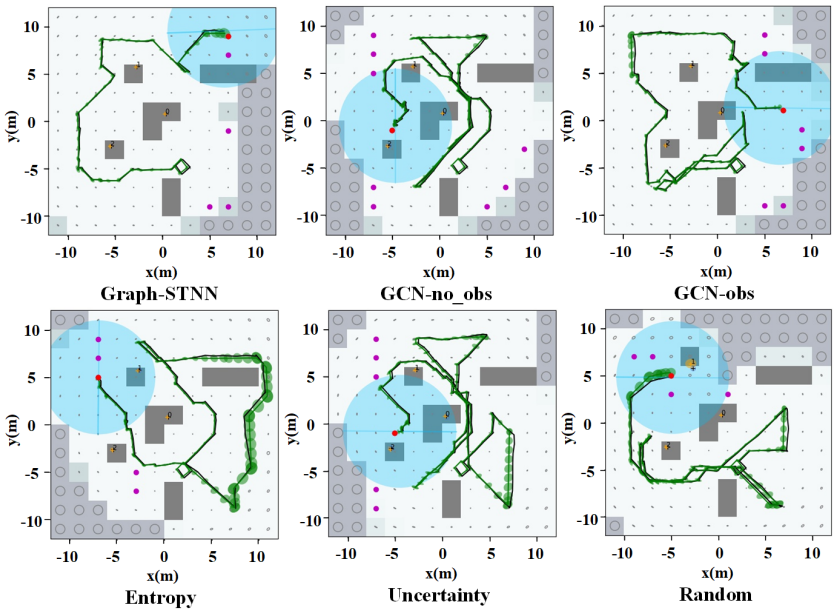
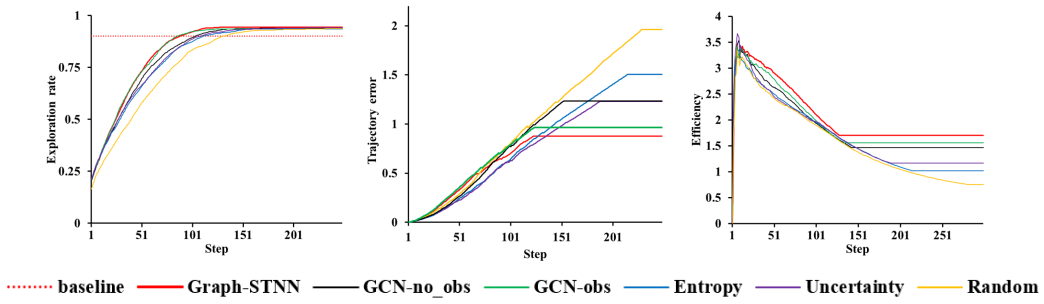


Figure 9. The training performance of Graph-STNN and GCN-obs in our setting with the size of 20 m × 20 m.

We test the trained strategies in randomly generated maps with sizes 20 m × 20 m and 40 m × 40 m. We test 50 episodes, and the results of the comparison are shown in Figures 10 and 11. To evaluate the performance, we use three metrics: the average exploration speed, e (the average trajectory error, which measures the mapping accuracy), and η (the average exploration efficiency, which measures the reduction of map uncertainty per unit time). The experimental value of e and η of different methods in 20 m × 20 m and 40 m × 40 m environments are shown in Table 2.

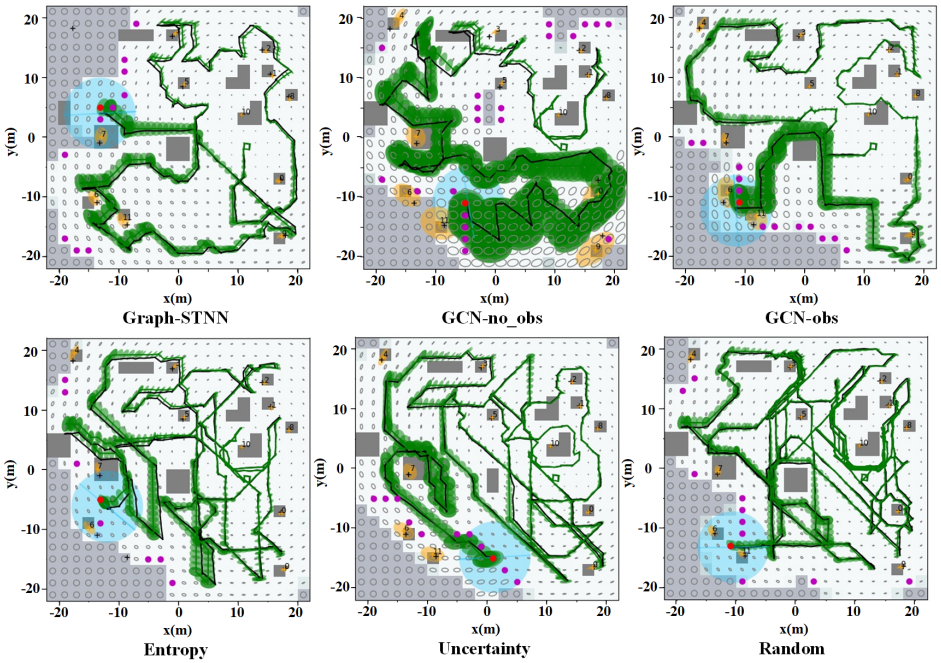


(**a**)

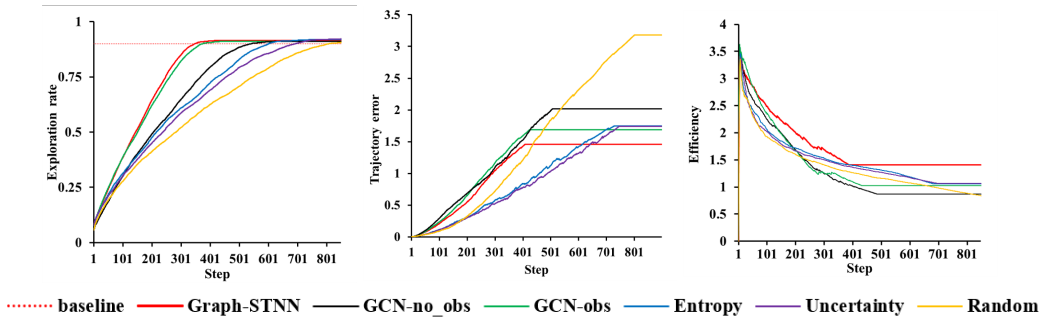


(**b**)

Figure 10. The results of 50 exploration trials of Graph-STNN, GCN-obs, GCN-no_obs, entropy-based, uncertainty-based and random methods , with the same randomly initialized landmarks and robot start locations, on 20 m × 20 m maps. (**a**) One of the test result of 50 exploration trials. (**b**) Average exploration rate (left), trajectory error (middle), and exploration efficiency (right).



(**a**)



(**b**)

Figure 11. The results of 50 exploration trials of Graph-STNN, GCN-obs, GCN-no_obs, entropy-based, uncertainty-based and random methods , with the same randomly initialized landmarks and robot start locations, on 40 m × 40 m maps (the models are trained in 20 m × 20 m grid maps, and the results shown are plotted per time-step of the simulation). (**a**) One of the test result of 50 exploration trials. (**b**) Average exploration rate (left), trajectory error (middle), and exploration efficiency (right).

Table 2. The average trajectory error and average exploration efficiency of different methods with the size of 20 m × 20 m and 40 m × 40 m.

Method	e		η	
	20 m × 20 m	40 m × 40 m	20 m × 20 m	40 m × 40 m
Graph-STNN	0.8761	1.4607	1.7051	1.4032
GCN-no_obs	1.2328	2.0179	1.4630	0.8719
GCN-obs	0.9665	1.6894	1.5573	1.0270
Entropy	1.5038	1.7454	1.0219	1.0571
Uncertainty	1.2304	1.7406	1.1676	1.0651
Random	1.9614	3.1803	0.7536	0.7779

The result shows that Graph-STNN gets the shortest path, the smallest trajectory error, and the largest exploration efficiency. Besides, the Random method has the worst performance in all three metrics. In addition, by introducing Uncertainty in reference [25] and Entropy in reference [15], we can see that the performance is better than the Random method. However, both of them are still not good enough.

Intuitively, it is evident in Figure 11 that Graph-STNN with obstacle information consideration has a better performance compared with the model without that (GCN-no_obs), which is closer to the actual situation. Because we take the obstacles as nodes in the graph, which enriches graphical representation. Meanwhile, by adding path planning, the approach can learn the path changes due to obstacle avoidance, which supports our claim (i): our approach chooses more reasonable target points by taking into obstacle information.

Furthermore, it can be seen in Figure 11a, in the test of 50 episodes, Graph-STNN finish exploring the environment fastest, with the highest accuracy and efficiency of mapping compared with GCN-based and information entropy-based methods. Meanwhile, the advantage is more significant than in the 20 m $\times$ 20 m environment, showing higher robustness and generalization, as our claim (iii). Eventually, the results verify that the global spatiotemporal information plays a more critical role in the exploration problem than single structural information, as our claim (ii).

To verify the role of the spatial encoder, temporal encoder, state encoder, and their combination in Graph-STNN, we train the networks with different components separately. The reward curves change during the training process are as shown in Figure 12, in which the rewards are averaged every ten thousand steps. Furthermore, as shown in the picture, the approaches considering separately state features and action features converge faster and obtain more rewards, as our claim (iv).

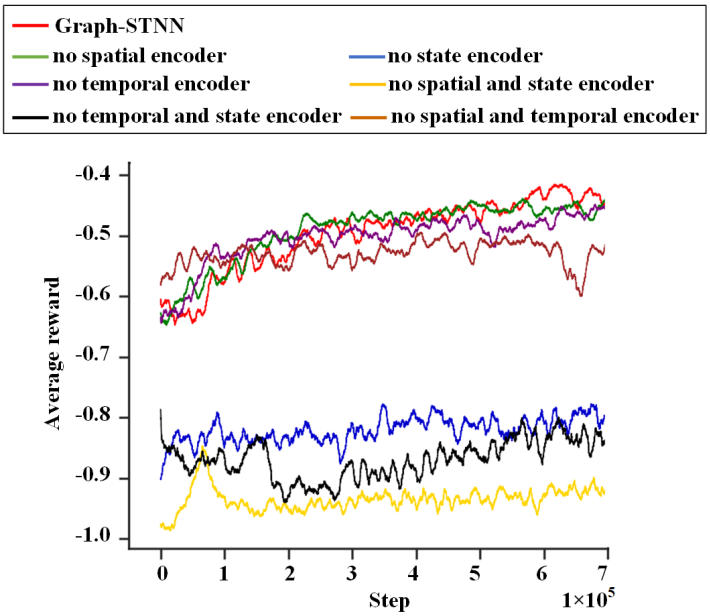


Figure 12. The training performance of Graph-STNN and the removal of some of its components on randomly generated simulation environments, with the size of 20 m $\times$ 20 m.

Next, we test the trained network with the map size of 20 m $\times$ 20 m, 30 m $\times$ 30 m and 40 m $\times$ 40 m, with different obstacle and landmark distributions. We test 50 episodes, and the results are shown in Figures 13–15 and Table 3.

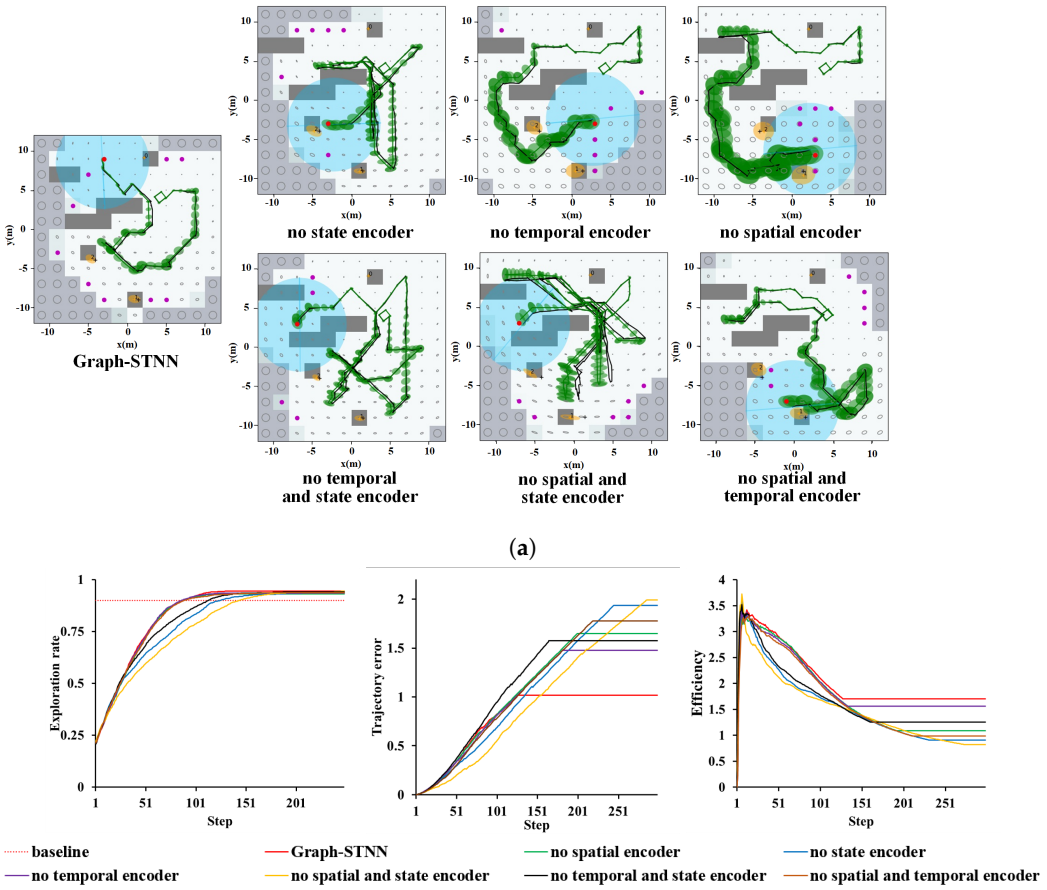


Figure 13. The results of 50 exploration trials of Graph-STNN and the removal of its components, with the model trained in 20 m × 20 m maps and the same initialized landmarks and robot's start locations, on 20 m × 20 m maps. (**a**) One of the test result of 50 exploration trials. (**b**) Average exploration rate (left), trajectory error (middle), and exploration efficiency (right).

The results show that the convergence speed of the models without spatial or temporal information (no temporal encoder, no spatial encoder, no spatial and temporal encoder) is close to Graph-STNN. However, they obtain fewer rewards and have worse efficiency. Besides, we can see that without state features (no state encoder, no spatial and state encoder, no temporal and state encoder) the performance of convergence speed, mapping accuracy, and exploration efficiency are all not very good, and they are difficult to converge. The reason is that spatial and temporal information tend to guide robots to reduce the environment's uncertainty, which leads to fast exploration. In the meanwhile, the state information tends to balance the exploration speed and the map accuracy, since it always contains the global information during the exploration progress.

To further check the model's robustness and generalization, we directly extend the trained model to 30 m × 30 m and 40 m × 40 m environment.

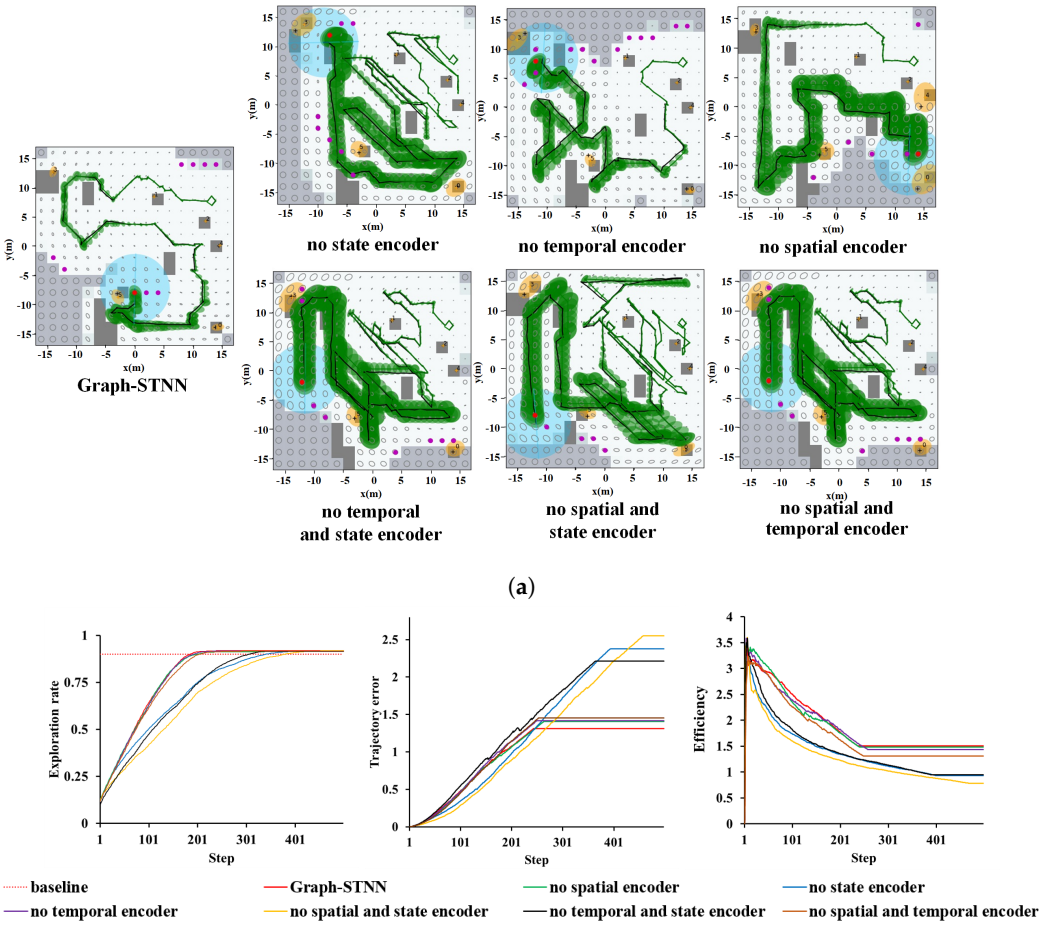


Figure 14. The results of 50 exploration trials of Graph-STNN and the removal of its components, with the model trained in 20 m × 20 m maps and the same initialized landmarks and robot's start locations, on 30 m × 30 m maps. (**a**) One of the test result of 50 exploration trials. (**b**) Average exploration rate (left), trajectory error (middle), and exploration efficiency (right).

Table 3. The trajectory error and exploration efficiency of Graph-STNN and its removal of some components with the size of 20 m × 20 m, 30 m × 30 m and 40 m × 40 m.

Method	e			η		
	20 m × 20 m	30 m × 30 m	40 m × 40 m	20 m × 20 m	30 m × 30 m	40 m × 40 m
Graph-STNN	1.0160	1.3088	1.7588	1.7051	1.5085	1.4032
no state	1.9370	2.3748	2.7195	0.9039	0.9304	0.9153
no temporal	1.4761	1.8144	2.2950	1.5585	1.4389	1.3101
no spatial	1.6475	1.8303	2.5787	1.0854	1.4828	1.1593
no temporal and state	1.5732	2.2140	2.5679	1.2563	0.9476	0.9950
no spatial and state	1.9935	2.5510	2.7051	0.8231	0.7813	0.8142
no spatial and temporal	1.7770	2.5523	2.3844	0.9823	1.3127	1.0719

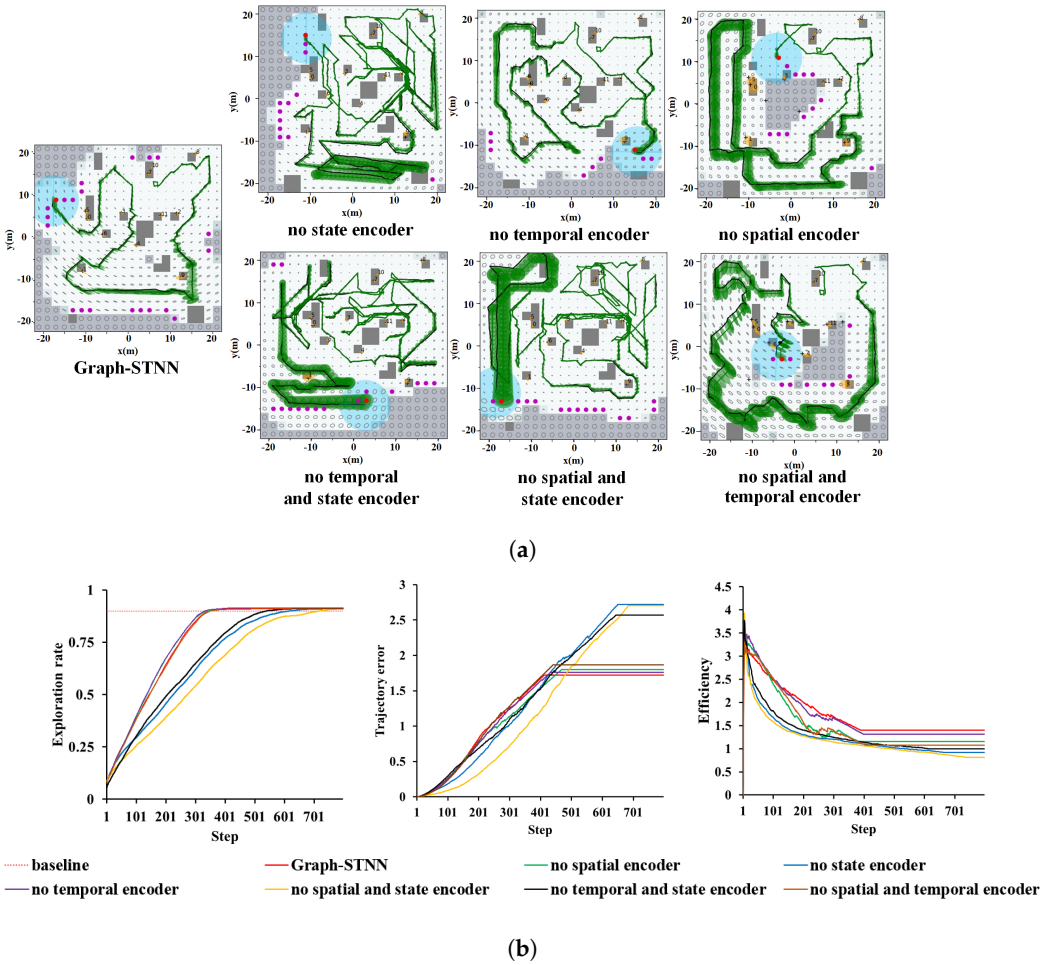
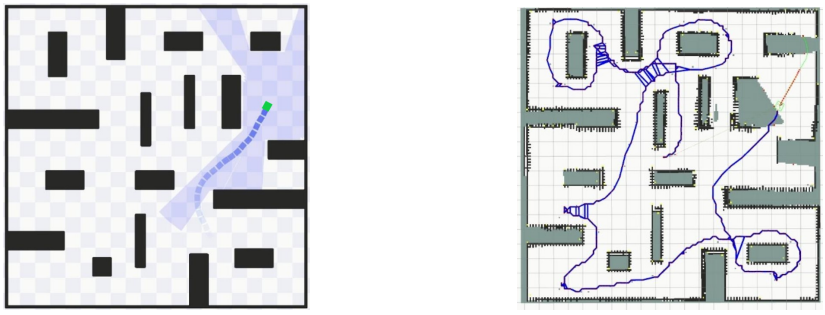


Figure 15. The result of 50 exploration trials of Graph-STNN and the removal of its components, with the model trained in 20 m × 20 m maps and the same initialized landmarks and robot's start locations, on 40 m × 40 m maps. (**a**) One of the test result of 50 exploration trials. (**b**) Average exploration rate (left), trajectory error (middle), and exploration efficiency (right).

As shown in Figures 14 and 15, the complete Graph-STNN achieves the best results compared to the models using temporal, spatial, and state features alone or in part. These results once again confirm our claim (ii): Graph-STNN can better adapt to the exploration problem using temporal and spatial information and lead to more accurate exploration. As we can see, the state encoder can promote exploration speed and efficiency, which plays a major role in improving the model's generalization. Meanwhile, the temporal and spatial encoders mainly play a role in improving the mapping accuracy.

Finally, we use a physical robot model and a lifelike environment to further validate our approach on Stage. Stage is a simulation tool for researching mobile robots and intelligent sensor systems, which can simulate objects such as mobile robots, sensors, and obstacles. As shown in Figure 16a, the size of our scene is 20 m × 20 m, with dense obstacles. The robot achieves omni-directional motion by differential rotation of two wheels, and it carries a 2D lidar with the field of view (FOV) of 360 degrees and the range of 10 m. Then, we test our autonomous exploration in this platform. First, the robot builds the map through the Simultaneous Localization and Mapping (SLAM) algorithm Karto [37], and the frontier waypoints are obtained from the map. The test result is shown in Figure 16b, in which the established map, the robot's historical trajectories, and the robot's current path obtained by Mobile Robot Local Planning Benchmark (MRPB) [38] are visualized by Rviz. As we can see, our approach can guide the robot equipped with 2D lidar to effectively explore the unknown environment, which means the proposed method has great potential in actual application.



(**a**) (**b**)

Figure 16. The test result of our autonomous exploration algorithm in Stage. (**a**) The Stage scene. In this scene, the black squares are obstacles, the green square is the robot's position, the blue squares are the robot's historical trajectories, and the light blue area is the radar coverage area. (**b**) The explored map. In this map, the black grids are obstacles, the white grids are the free areas, and the dark green grids are unknown areas. The blue curve is the robot's historical trajectories and the blue line indicates that a closed loop has been detected. Red and green curves are the robot's current path.

5. Conclusions

This research develops a novel DRL-based approach called Graph-STNN for mobile robot autonomous exploration in unknown environments. We modeled the exploration problem as an exploration graph that considers the historical trajectories, frontier waypoints, landmarks, and obstacles. We employ GCN as a spatial encoder to extract topological structure information, TCN as a temporal encoder to extract sequence information of trajectories, GRU to capture dynamic state change of spatiotemporal information to obtain the current state feature. The simulation experiment shows that our approach could choose a better target point by considering obstacle avoidance. Besides, the results also show that the method can promote the exploration efficiency by both using temporal and spatial information. In addition, it performs better robustness and generalization by using hybrid global information. Next, we are going to validate the network on physical robots, and DRL-based path planning and heterogeneous graphs will be considered.

Author Contributions: Conceptualization, Z.Z. (Zhiwen Zhang), C.S., P.Z., Z.Z. (Zhiwen Zeng) and H.Z.; methodology, Z.Z. (Zhiwen Zhang), C.S. and P.Z.; software, Z.Z. (Zhiwen Zhang); validation, Z.Z. (Zhiwen Zhang); formal analysis, Z.Z. (Zhiwen Zhang) and P.Z.; investigation, Z.Z. (Zhiwen Zhang); resources, Z.Z. (Zhiwen Zhang); data curation, Z.Z. (Zhiwen Zhang); writing—original draft preparation, Z.Z. (Zhiwen Zhang); writing—review and editing, Z.Z. (Zhiwen Zhang), C.S., P.Z., Z.Z. (Zhiwen Zeng) and H.Z.; supervision, H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China, grant number U1913202, and also supported by National Key R&D Program of China, grant number YFC20170806503.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available at https://github.com/zhangzw-nudt/Graph-STNN_Exploration (accessed on 22 July 2021).

Acknowledgments: This study is a part of the M.S. research of the corresponding author. We are grateful to anonymous reviewers and editors for their suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lluvia, I.; Lazkano, E.; Ansuategi, A. Active Mapping and Robot Exploration: A Survey. *Sensors* **2021**, *21*, 2445. [CrossRef] [PubMed]
2. Juliá, M.; Gil, A.; Reinoso, Ó. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Auton. Robot.* **2012**, *33*, 427–444. [CrossRef]
3. Stachniss, C. *Robotic Mapping and Exploration*; Springer: Berlin/Heidelberg, Germany, 2009; Volume 55.
4. Chen, F.; Wang, J.; Shan, T.; Englot, B. Autonomous Exploration Under Uncertainty via Graph Convolutional Networks. In Proceedings of the International Symposium on Robotics Research, Hanoi, Vietnam, 6–10 October 2019.
5. Tai, L.; Liu, M. Mobile robots exploration through cnn-based reinforcement learning. *Robot. Biomim.* **2016**, *3*, 1–8. [CrossRef] [PubMed]
6. Bai, S.; Chen, F.; Englot, B. Toward autonomous mapping and exploration for mobile robots through deep supervised learning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 2379–2384.
7. Tai, L.; Paolo, G.; Liu, M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 31–36.
8. Chen, F. Autonomous Exploration under Uncertainty via Deep Reinforcement Learning on Graphs. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 6140–6147.
9. Li, H.; Zhang, Q.; Zhao, D. Deep reinforcement learning-based automatic exploration for navigation in unknown environment. *IEEE Trans. Neural Netw. Learn. Syst. (TNNLS)* **2019**, *31*, 2064–2076. [CrossRef] [PubMed]
10. Niroui, F. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robot. Autom. Lett. (RA-L)* **2019**, *4*, 610–617. [CrossRef]
11. Yamauchi, B. A frontier-based approach for autonomous exploration. In Proceedings of the IEEE International Symposium on Computer Intelligence in Robotics and Automation (CIRA), Monterey, CA, USA, 10–11 July 1997; pp. 146–151.
12. Makarenko, A.A. An experiment in integrated exploration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Lausanne, Switzerland, 30 September–4 October 2002; Volume 1, pp. 534–539.
13. Kaufman, E. Bayesian occupancy grid mapping via an exact inverse sensor model. In Proceedings of the IEEE American Control Conference (ACC), Boston, MA, USA, 6–8 July 2016; pp. 5709–5715.
14. Kaufman, E.; Lee, T.; Ai, Z. Autonomous exploration by expected information gain from probabilistic occupancy grid mapping. In Proceedings of the IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), San Francisco, CA, USA, 13–16 December 2016; pp. 246–251.
15. Carrillo, H. Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 487–494.
16. Bai, S. Information-theoretic exploration with Bayesian optimization. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 1816–1822.
17. González-Banos, H.H.; Latombe, J.C. Navigation strategies for exploring indoor environments. *Int. J. Robot. Res. (IJRR)* **2002**, *21*, 829–848. [CrossRef]
18. Wu, Z. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst. (TNNLS)* **2020**, *32*, 4–24. [CrossRef] [PubMed]
19. Mnih, V. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
20. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arxiv:1609.02907.
21. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv* **2015**, arXiv:1506.00019.
22. Rossi, L.; Paolanti, M.; Pierdicca, R.; Frontoni, E. Human trajectory prediction and generation using LSTM models and GANs. *Pattern Recognit.* **2021**, *120*, 108136. [CrossRef]
23. Rossi, L.; Ajmar, A.; Paolanti, M.; Pierdicca, R. Vehicle trajectory prediction and generation using LSTM models and GANs. *PLoS ONE* **2021**, *16*, e0253868.
24. Ye, F.; Yang, J. A Deep Neural Network Model for Speaker Identification. *Appl. Sci.* **2021**, *11*, 3603. [CrossRef]
25. Wang, J.; Englot, B. Autonomous Exploration with Expectation-Maximization. In *Robotics Research*; Springer: Cham, Switzerland, 2020; pp. 759–774.
26. Kaess, M.; Dellaert, F. Covariance recovery from a square root information matrix for data association. *J. Robot. Auton. Syst. (RAS)* **2009**, *57*, 1198–1210. [CrossRef]
27. Rumelhart, D.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
28. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef] [PubMed]

29. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Zhang, C. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In Proceedings of the International Conference on Artificial Intelligence (IJCAI), Macao, China, 10–16 August 2019.
30. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271.
31. Wang, Z. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning (ICML), New York, NY, USA, 19–24 June 2016; pp. 1995–2003.
32. Cho, K.; van Merrienboe, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
33. Fortunato, M.; Azar, M.G.; Piot, B.; Menick, J.; Hessel, M.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; et al. Noisy Networks For Exploration. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
34. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
35. Dellaert, F. *Factor Graphs and GTSAM: A Hands-on Introduction*; Technical Report GT-RIM-CP&R-2012-002; Georgia Institute of Technology: Atlanta, GA, USA, 2012.
36. Brownlee, J. *Long Short-Term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*; Machine Learning Mastery: San Juan, PR, USA, 2017.
37. Konolige, K.; Grisetti, G.; Kümmerle, R.; Burgard, W.; Limketkai, B.; Vincent, R. Efficient Sparse Pose Adjustment for 2D mapping. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 18–22 October 2010; pp. 22–29.
38. Wen, J.; Zhang, X.; Bi, Q.; Pan, Z.; Feng, Y.; Yuan, J.; Fang, Y. MRPB 1.0: A Unified Benchmark for the Evaluation of Mobile Robot Local Planning Approaches. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 8238–8244.