Multi-Robot Active Mapping via Neural Bipartite Graph Matching

Kai Ye[1*] Siyan Dong[2,1*] Qingnan Fan[3†] He Wang[1] Li Yi[4] Fei Xia[5]
Jue Wang[3] Baoquan Chen[1†]
[1]Peking University [2]Shandong University [3]Tencent AI Lab
[4]Tsinghua University [5]Stanford University
{siyandong.3, fqnchina, ericyi0124, xf1280, arphid, baoquan.chen}@gmail.com
{ye_kai, hewang}@pku.edu.cn

Abstract

We study the problem of multi-robot active mapping, which aims for complete scene map construction in minimum time steps. The key to this problem lies in the goal position estimation to enable more efficient robot movements. Previous approaches either choose the frontier as the goal position via a myopic solution that hinders the time efficiency, or maximize the long-term value via reinforcement learning to directly regress the goal position, but does not guarantee the complete map construction. In this paper, we propose a novel algorithm, namely NeuralCoMapping, which takes advantage of both approaches. We reduce the problem to bipartite graph matching, which establishes the node correspondences between two graphs, denoting robots and frontiers. We introduce a multiplex graph neural network (mGNN) that learns the neural distance to fill the affinity matrix for more effective graph matching. We optimize the mGNN with a differentiable linear assignment layer by maximizing the long-term values that favor time efficiency and map completeness via reinforcement learning. We compare our algorithm with several state-of-the-art multi-robot active mapping approaches and adapted reinforcement-learning baselines. Experimental results demonstrate the superior performance and exceptional generalization ability of our algorithm on various indoor scenes and unseen number of robots, when only trained with 9 indoor scenes.

1. Introduction

Constructing the map of indoor environments is of great importance to a wide range of applications in the computer vision and robotics communities. With the fast development of range sensors (Kinect, RealSense), many scene mapping approaches [23, 29, 11, 15] are developed to empower scene traversal by human operators with handheld sensors, yet incomplete or unaligned scene meshes are common flaws for inexperienced users due to the noisy and unstable scanned trajectory. To alleviate the inconvenience of human-operated traversal, there emerges autonomous map construction [46, 17, 24, 32] via active sensor movement, also known as *active mapping*. Previous works in this field mainly focus on using a single robot, which is time-consuming for large-scale environments. In this paper, we study the problem of *multi-robot active mapping*: coordinating multiple robots for the autonomous reconstruction of unknown scenes.

The goal of active mapping is mainly twofold: *time efficiency*, and *map completeness*. The pioneering work for active mapping [46] introduces the concept of frontier: regions on the boundary between open space and unexplored space. By continuously moving the robot to new frontiers, the scene map can be completely constructed when no frontier can be found. Many follow-up approaches in the following decades [6, 16, 31] aim to improve the time efficiency of the process. However, the problem of active mapping is highly ambiguous, which makes a theoretically-optimal solution almost impossible to be found in an unknown environment.

The key module of active mapping that influences time efficiency is the global planner that estimates the goal position for path planning. The vast majority of literature for both single robot [3, 37, 39, 1] and multiple robots [4, 30, 14, 16] are *frontier-based*, which decides the goal position from a set of frontiers. However, these approaches are mostly myopic [6] and hence hinder the time efficiency, since they either handcraft heuristics [46, 17, 21] to choose the frontier in the shortest geodesic distance to the robot, or find the one that maximizes the information gain over the next few actions via information-theoretic optimization [37, 1]. The more recent approaches adopt the *reinforcement learning* strategies [13, 8, 31] as a replacement of the traditional approaches to decide the goal position for single robot. These policy learning approaches have dominated the active mapping field lately, thanks to their potential to achieve more efficient solu-

[*]Joint first authors
[†]Corresponding authors

tions by maximizing the long-term value [22, 25]. However, as their goal positions are mostly regressed and may not lay on the frontiers, it has no guarantee to construct the complete map [13, 8]. When the setting of active mapping is extended to the multi-robot scenario, the action space is linearly increased with the robot number, which makes the problem even more ambiguous. The past multi-robot approaches [4, 30, 14, 16, 18] are mostly frontier-based myopic solutions and are still limited in time efficiency.

In this paper, we propose a novel multi-robot active mapping approach that takes advantage of both the traditional frontier-based and recent reinforcement learning solutions for more efficient and complete map construction. To be specific, we coordinate multiple robots to decide the goal positions from a set of frontiers according to the neural distance optimized by maximizing the long-term value via reinforcement learning. To achieve this goal, we reduce the multi-robot active mapping problem to *bipartite graph matching*, which establishes node correspondences between two graphs, denoting robots and frontiers separately. The key issue for bipartite graph matching lies in the computation of the affinity matrix between two sets of nodes. The traditional frontier-based approaches can be considered as handcrafting the affinity matrix with the geodesic distance between robots and frontiers, which limits the time efficiency of active mapping. In our algorithm, we propose to learn the neural distance with a *multiplex graph neural network* (mGNN) to estimate the affinity matrix for graph matching. The problem of graph matching is NP-hard in nature [5] and often formulated as quadratic assignment programming, which is expensive and complex to solve. Many recent works relax graph matching as a linear assignment problem [42], which can be efficiently tackled with a differentiable and approximate solution [36]. Therefore, we optimize the graph neural network with the differentiable linear assignment by maximizing the long-term value that favors high time efficiency and map completeness via reinforcement learning.

Our algorithm is trained with only 9 indoor scenes, and exhibits exceptional generalization ability to various indoor scene datasets and unseen number of robots. The experimental results demonstrate the superiority of our algorithm over state-of-the-art multi-robot active mapping approaches and a couple of adapted reinforcement-learning baselines.

All in all, our contributions can be summarized as follows:

- We reduce the multi-robot active mapping problem to bipartite graph matching, which is solved by a novel multi-robot active mapping algorithm that takes advantage of both the traditional frontier-based and recent reinforcement learning approaches.
- Our algorithm employs a multiplex graph neural network to estimate the affinity matrix, followed by a linear assignment layer for graph matching. The entire process is optimized by maximizing the long-term value via reinforcement learning.
- While achieving the complete map construction, our algorithm outperforms the existing multi-robot active mapping approaches over time efficiency by a large margin, and demonstrates exceptional generalization ability to unseen robot numbers.

2. Related Works

Single-robot active mapping. The vast majority of past works for active mapping lie in the single-robot scenario [46, 17, 21, 26]. The pioneering work of Yamauchi [46] for active mapping presents the concept of frontier and moves the robot towards the nearest frontier in the occupancy map. Dornhege and Kleiner [17] extend this idea with the new concept of void, which is the unexplored volumes in the 3D occupancy map. The other popular thread for active mapping relies on the information theory [3, 37, 39, 1] to choose the frontier based on instant information gain.

Unlike the above traditional approaches that decide the goal position from a set of frontiers mainly with myopic strategies, recent approaches [13, 8] directly employ a convolution neural network to regress the goal position by maximizing the long-term value via reinforcement learning. Ramakrishnan et al. [31] follow the above works and proposes to broaden the map coverage beyond the visible area through occupancy anticipation. Some other works explore the environment by constructing a topological map [10] or semantic map [9], which is beyond the scope of this work.

Multi-robot active mapping. The large variety of related works about multi-robot active mapping [16, 41, 2, 19, 4, 30, 14] share a lot of methodologies with the single-robot approaches and differ mainly in how they coordinate multiple robots for the goal assignment. Faigl et al. [19] have a good summary of the common multi-robot active mapping approaches. One solution [43] is to sort the robot-goal pairs with the geodesic distance and traverse the ordered sequence from the first element to assign the next not-assigned goal to the robot. More recent approaches [16, 19] mainly rely on solving an optimal mass transport problem between robots and goals, such as multiple traveling salesman problem, for goal assignments. All the aforementioned approaches are myopic as analyzed [16, 6] since the goal positions are chosen from the nearest target in the geodesic distance with multi-robot coordination constraints. In this work, we propose to learn better neural distance via reinforcement learning to achieve more efficient map construction.

Graph neural networks. Graph neural networks [44] enable learning on top of graph representations. Sykora et al. [38] use graph neural network to solve the multi-agent graph coverage problem. Zhang et al. [48] introduce multiplex network structures for the multi-behavior recommendation. In this work, we perform learning on multiple graphs, namely multiplex graph neural network (mGNN), to estimate the

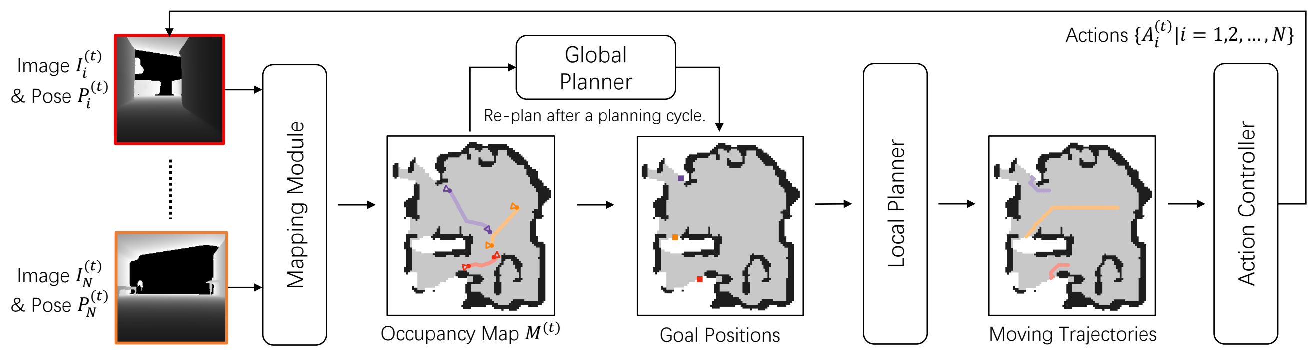

Figure 1. The entire pipeline of our framework. In each planning cycle, mapping (Section 4.1), local planning and action control (Section 4.3) are conducted at each step, while global planning (Section 4.2) is conducted only once at the beginning. Such a planning cycle is iterated until meeting the termination criterion of active mapping.

affinity matrix for bipartite graph matching.

3. Problem Statement

In an unknown environment, the goal of multi-robot active mapping is to construct a complete map in minimum time steps. At each time step t, the robot i receives a first-person depth image $I_i^{(t)}$ and its corresponding camera pose $P_i^{(t)}$ as input, and estimates its action $A_i^{(t)}$ for movement, following the problem setting of the previous work closest to ours [16]. We adopt the common TurtleBot model as our robot, and it runs in the physically-realistic simulator iGibson [35, 27], which contains the physical robot body and simulates the realistic action noise and collisions.

For better comparison with previous works [16, 19], we further consider the problem settings below. 1) *Map completeness first*: we value map completeness the most, hence the map is continuously explored until no accessible frontier is found. 2) *Co-located robots*: the pose information is shared among all the robots, hence the global map can be constructed by synchronizing the local maps from all the robots. 3) *Spatially-close initialization*: all the robots are randomly initialized in the traversable region of the map with the constraint that the geodesic distance between every two robots is smaller than a threshold λ_r. Note uniformly sampling the robots in the entire map will save more scanning effort for exploration, and we are working in a more challenging setting with spatially-close robot initialization.

4. NeuralCoMapping

We introduce the entire framework (Figure 1) of our algorithm below. The *mapping module* constructs the occupancy map based on the current depth observations and camera poses from all the robots. The *global planner* estimates the goal position for each robot in the occupancy map by solving a neural bipartite graph matching problem. To navigate to the goal position, the *local planner* calculates an obstacle-free moving trajectory for each robot, followed by the *action controller* that performs the specific robot moves along the trajectory. We define a *planning cycle* as a short period of a fixed horizon. The global planner only estimates the goal positions at the beginning of the planning cycle, while mapping, local planning, and action control are alternatively conducted until the end of the planning cycle. Such a planning cycle is iterated until meeting the termination criterion.

4.1. Mapping Module

Provided the depth observations and the corresponding camera poses from all the robots, the goal of the mapping module is to construct the global map. We represent the map at time step t as the occupancy grid, which is the top-down view of the 3D scene, denoted as $M^{(t)} \in [0,1]^{X\times Y\times 2}$, where X, Y are the map size. The two map channels indicate the explored and occupied regions separately. Hence each cell in the grid can be classified as one of the three classes, open (explored but not occupied), occupied, and unknown (unexplored). Frontier is defined as the open cells whose adjacency contains unknown cells. The occupancy map $M^{(0)}$ is unexplored and initialized as all zero. For robot i, the pose of its mounted camera $P_i^{(t)} \in \mathbb{SE}(2)$ is represented as (x, y, θ), where $x \in [0, X-1], y \in [0, Y-1]$ denote the robot position and $\theta \in [0, 2\pi)$ denotes the robot orientation around the vertical axis.

To compute the occupancy map, the depth image $I_i^{(t)}$ is firstly back-projected as the point cloud in the world space with known camera intrinsics and provided poses $P_i^{(t)}$, then the 3D points from all the robots are fused and projected onto the 2D plane to overlay the overlapping region of the existing occupancy map $M^{(t-1)}$ to obtain the new one $M^{(t)}$. We consider a cell as occupied if there is any 3D point (staying between two horizontal planes at the height of the ground and robot head) that falls into the cell, and consider a cell as open if it lies on the ray from the camera center to the end of the definitively visible space [20]. The multi-robot scenario creates a highly dynamic environment, where one moving robot may be visually observed by the others. It poses new challenges for obstacle-free path planning. To tackle this issue, we label all the cells covered by any robot

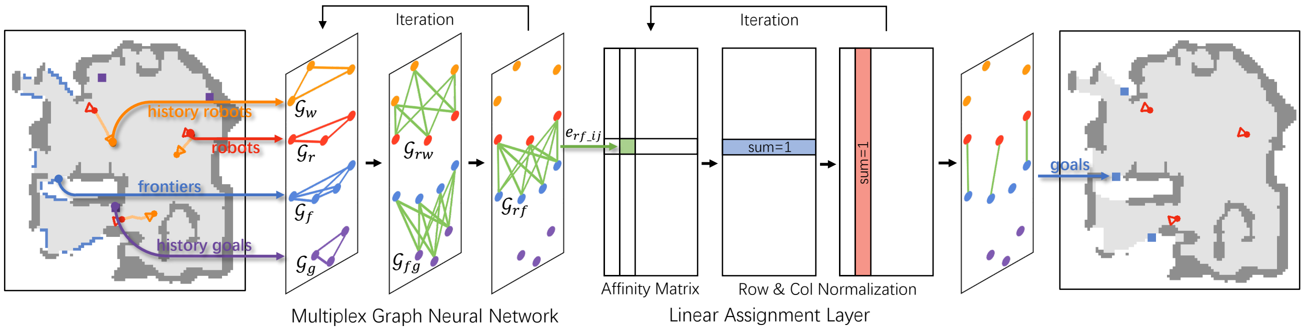

Figure 2. Illustration of the global planner. It consists of a multiplex graph neural network (mGNN) that learns the affinity information between robots and frontiers, and a linear assignment layer to match them. The robot and frontier nodes are extracted from the mapping module, while the history robot and goal nodes are extracted from the past planning cycles. With the affinity matrix formed by the learned directed edge features, the linear assignment layer conducts bipartite graph matching to assign each robot a unique frontier as the goal.

being observed as occupied, and label those cells as open again once the robot leaves them.

4.2. Global Planner

With the constructed occupancy map and provided robot positions, the global planner aims to estimate the optimal goal positions for all the robots for efficient and complete map construction. In this work, we formulate the problem of goal position estimation as *bipartite graph matching*, which establishes the correspondences between the robot and frontier[1] nodes extracted from the constructed occupancy map. The affinity matrix for graph matching is not composed of the geodesic distance as adopted in the traditional frontier-based approaches [41, 2, 19, 16], yet is filled with the neural distance estimated by a graph neural network, which is optimized with a differentiable linear assignment layer by maximizing the long-term value via reinforcement learning. We introduce the solution to bipartite graph matching via the following two components, *multiplex graph neural network* for affinity matrix estimation, and *linear assignment layer* to pair the robot and frontier nodes. Figure 2 gives the illustration of the aforementioned global planner.

4.2.1 Multiplex Graph Neural Network

Provided the constructed occupancy map and known robot positions, we first construct two self-graphs $\mathcal{G}_r = (\mathcal{V}_r, \mathcal{E}_r)$ and $\mathcal{G}_f = (\mathcal{V}_f, \mathcal{E}_f)$ that denote the robot and frontier sets separately. We also build a cross-graph $\mathcal{G}_{rf} = (\mathcal{V}_r, \mathcal{V}_f, \mathcal{E}_{rf})$ that connects the robots and frontiers, and $\mathcal{E}_{rf}$ denotes the affinity information we want to learn for graph matching. The multiplex graph neural network (mGNN) learns such information between robots and frontiers with iterative intra-graph and inter-graph operations.

For sake of simplicity, we introduce the node feature computation only for robots below, and it applies to the frontier nodes as well. For robot i, we represent the raw robot information as $s_{r\_i} \in \mathbb{R}^3$, which includes the x, y coordinates in the occupancy map and its semantic label (*robot*, or *frontier*). We extract the initial high-dimensional robot node feature $v_{r\_i}^{(0)} \in \mathbb{R}^{32}$ from s_{r_i} via a multi-layer perception (MLP) f_{init}:

$$v_{r\_i}^{(0)} = f_{init}(s_{r\_i}) \tag{1}$$

Intra-graph operation. The intra-graph operation updates the node and edge features for both the robot and frontier self-graphs. We consider a fully connected self-graph, hence each node will be updated by the messages received from all the other nodes. Inspired by the attention mechanism [40], such a node aggregation operation can be treated as the retrieval process which maps your query against a set of keys associated with candidate nodes in the graph and finally presents the best matched nodes (values). Hence, we first compute the query $q_{r\_i}^{(l)} \in \mathbb{R}^{32}$, key $k_{r\_i}^{(l)} \in \mathbb{R}^{32}$, and value $u_{r\_i}^{(l)} \in \mathbb{R}^{32}$ from the node feature $v_{r\_i}^{(l)}$ at layer l,

$$q_{r\_i}^{(l)} = f_{query}(v_{r\_i}^{(l)}), k_{r\_i}^{(l)} = f_{key}(v_{r\_i}^{(l)}), u_{r\_i}^{(l)} = f_{value}(v_{r\_i}^{(l)}) \tag{2}$$

where $f_{query}, f_{key}, f_{value}$ are parameterized as linear projections. Then we represent the directed edge feature $e_{r\_ij}^{(l)} \in \mathbb{R}^1$ from node j to node i as the attention weight scalar,

$$e_{r\_ij}^{(l)} = \frac{\exp\left(q_{r\_i}^{(l)} \cdot k_{r\_j}^{(l)}\right)}{\sum_{h:(i,h)\in\mathcal{E}_r} \exp\left(q_{r\_i}^{(l)} \cdot k_{r\_h}^{(l)}\right)} \tag{3}$$

which is the softmax over all the query-key dot product results directed to node i. Then the node feature $v_{r\_i}^{(l+1)}$ is computed as

$$v_{r\_i}^{(l+1)} = v_{r\_i}^{(l)} + f_{node}(v_{r\_i}^{(l)}, \sum_{h:(i,h)\in\mathcal{E}_r} e_{r\_ih}^{(l)} u_{r\_h}^{(l)}) \tag{4}$$

where f_{node} concatenates the input information and is parameterized as a multi-layer perception.

[1] The frontier nodes are sampled from the frontier cells, and each frontier node represents a frontier cell.

Inter-graph operation. The inter-graph operation updates the node and edge features for the cross-graph. We consider a complete bipartite graph, where each node in one set is connected with all the nodes in the other set, yet not connected with any nodes in the same set. The geodesic distance is the shortest distance for traversal between two points and implicitly encodes the underlying scene layout information. We compute geodesic distances with the Fast Marching Method (FMM) [34]. The geodesic distance between two nodes i, j is denoted as d_{ij}, which is incorporated into the directed edge feature $e^{(l)}_{rf\_ij} \in \mathbb{R}^1$ computation as below,

$$e^{(l)}_{rf\_ij} = \frac{\exp\left(f_{edge}(q^{(l)}_{r\_i}, k^{(l)}_{f\_j}, d_{ij})\right)}{\sum_{h:(i,h)\in\mathcal{E}_{rf}} \exp\left(f_{edge}(q^{(l)}_{r\_i}, k^{(l)}_{f\_h}, d_{ih})\right)} \quad (5)$$

where f_{edge} concatenates the input information and is parameterized as a multi-layer perception. Then the node feature $v^{(l+2)}_{r\_i}$ is computed similarly to Equation 4 by replacing the directed edges in $\mathcal{E}_r$ with $\mathcal{E}_{rf}$,

$$v^{(l+1)}_{r\_i} = v^{(l)}_{r\_i} + f_{node}(v^{(l)}_{r\_i}, \sum_{h:(i,h)\in\mathcal{E}_{rf}} e^{(l)}_{rf\_ih} u^{(l)}_{f\_h}) \quad (6)$$

History node module. For the problem of multi-robot active mapping, not only do the current robots and frontiers matter for the global planning, the robots and estimated goals in the past should also serve as the guidance to encourage consistent robot movements and discourage the redundant traversal over explored regions. Therefore, we further enhance mGNN with two sets of new nodes with the semantic label of *history robot* or *history goal*, which are derived from the robots and goals in the past individually. To be specific, we construct two more self-graphs $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w)$, $\mathcal{G}_g = (\mathcal{V}_g, \mathcal{E}_g)$ that denote the history robots and history goals, and two more cross-graphs $\mathcal{G}_{rw} = (\mathcal{V}_r, \mathcal{V}_w, \mathcal{E}_{rw})$, $\mathcal{G}_{fg} = (\mathcal{V}_f, \mathcal{V}_g, \mathcal{E}_{fg})$ that associate robots with history robots, frontiers with history goals separately. In this manner, we achieve the history node module by applying the intra-graph operation to $\mathcal{G}_g, \mathcal{G}_w$, and the cross-graph operation to $\mathcal{G}_{fg}, \mathcal{G}_{rw}$ as well. In the implementation, instead of considering the history robot positions among all the past steps, we only count in the history robot positions at the beginning of each past planning cycle to balance the number of history robot and history frontier nodes.

Therefore, the entire mGNN is composed of four self-graphs $\mathcal{G}_r, \mathcal{G}_f, \mathcal{G}_w, \mathcal{G}_g$ and three cross-graphs $\mathcal{E}_{rf}, \mathcal{E}_{rw}, \mathcal{E}_{fg}$. For each intra-graph operation, all the four self-graphs can be simultaneously updated, while for each inter-graph operation, the three cross-graphs need to be sequentially updated as the robot and frontier nodes will be updated twice in two cross-graphs. Note the cross-graph order for inter-graph operation does not affect the final node feature, which is simply added with the residual as computed in Equation 6. We consider one block of graph operations as the composition of one intra-graph and one inter-graph operation. The entire mGNN is composed of N_l blocks of graph operations.

4.2.2 Linear Assignment Layer

For our bipartite graph matching problem, the affinity matrix denotes the distance between the robot and frontier nodes. The learned edge feature e_{rf_ij} emitted from robot j to frontier i indicates how much the robot prefers the frontier and is treated as the neural distance to form the affinity matrix for graph matching[2]. With the cross-graph $\mathcal{G}_{rf} = (\mathcal{V}_r, \mathcal{V}_f, \mathcal{E}_{rf})$, the goal of multi-robot active mapping can be considered as achieving a maximum matching in a bipartite graph. It is required to conduct any many connections between robot and frontier nodes as possible by assigning at most one robot to one frontier, and also at most one frontier to one robot, in such a way the summed affinity among all the connections are maximized. It corresponds to the linear assignment problem and can be solved efficiently with the popular Sinkhorn algorithm [36]. It works by normalizing the rows and columns of the affinity matrix alternatively until convergence and is often treated as the approximate and differentiable version of the Hungarian algorithm [28]. We implement the linear assignment layer as the Sinkhorn algorithm, whose output decides the goal position as the matched frontier for each robot.

4.3. Local Planner and Action Controller

With the robot position, estimated goal position, and constructed occupancy map, the local planner decides a moving trajectory from the robot to the goal position. We adopt Fast Marching Method (FMM) [34] to achieve this purpose. However, due to the fact that the occupancy map is only a rough representation of the scene world and the unavoidable action noise when executing the moving trajectory, the robot will collide with obstacles from time to time, for example, in the narrow corridor, in which situation the fast marching approach may not help free the robot from the collision in the physically-realistic simulator iGibson [35, 27] and hence time efficiency is significantly influenced. To alleviate the above difficulty, we propose to improve FMM with an *obstacle-resistant strategy* that plans the moving trajectory away from the obstacles. To be specific, given the occupancy map and robot positions at time step t, we first generate two geodesic distance maps $D^{(t)}_r, D^{(t)}_o$ whose zero contour lies in the robot positions and obstacles separately with FMM, then update the robot distance value $D^{(t)}_{r\_i}$ at position i as,

$$D^{(t)}_{r\_i} = D^{(t)}_{r\_i} / (\epsilon + min(\tau, D^{(t)}_{o\_i}) \times \lambda_o) \quad (7)$$

[2]Experimentally we observe the directed edge feature from the frontier to robot makes the similar effect.

where $\tau = 0.001, \tau = 4, \lambda_o = 0.25$. It means amplifying the robot distance values whose positions are close to obstacles. Then $D_r^{(t)}$ is used to calculate the moving trajectory from the robot to its goal position.

The robot is able to perform three actions, $A = \{move\_forward, turn\_left, turn\_right\}$. Given the next waypoint in the moving trajectory, the robot controls its actions via a simple heuristic [12]: if the robot faces the waypoint, it moves forward; otherwise, it rotates towards the waypoint. To be specific, we compute the relative angle θ_a by subtracting the robot orientation from the orientation of the directed robot-to-waypoint edge, the action $A_i^{(t)}$ at time step t for robot i is decided as,

$$A_i^{(t)} = \begin{cases} move\_forward & \text{if } \theta_a > -\lambda_a \text{ or } \theta_a < \lambda_a \\ turn\_left & \text{if } \theta_a \leq -\lambda_a \\ turn\_right & \text{if } \theta_a \geq \lambda_a \end{cases} \tag{8}$$

where λ_a is the threshold that controls the forward and rotation movement.

4.4. Reinforcement Learning Design

We treat the mGNN as the policy network, which is optimized with the differentiable linear assignment layer together by maximizing the accumulated reward in the entire episode via reinforcement learning. Despite the multi-robot scenario, as our global planner adopts the centralized decision setting, we directly use the off-policy learning approach Proximal Policy Optimization (PPO) [33] as the policy optimizer.

Reward function. The goal of active mapping is to pursue high time efficiency and map completeness. To achieve this goal, we design a time reward $\hat{\mathcal{R}}_{time}$ and a coverage reward $\hat{\mathcal{R}}_{coverage}$. The time reward punishes unnecessary time steps to encourage high exploration efficiency, hence is defined as,

$$\hat{\mathcal{R}}_{time} = -0.015 \tag{9}$$

Coverage $C(M^{(t)})$ at time step t is the area of the open space in the occupancy map $M^{(t)}$. The coverage reward $\hat{\mathcal{R}}_{coverage}$ is defined as the coverage increment measured in m^2,

$$\hat{\mathcal{R}}_{coverage} = C(M^{(t)}) - C(M^{(t-1)}) \tag{10}$$

Then the entire reward $\hat{\mathcal{R}}$ is computed as the weighted summation of the two above two rewards,

$$\hat{\mathcal{R}} = \hat{\mathcal{R}}_{time} + \lambda_c \hat{\mathcal{R}}_{coverage} \tag{11}$$

where λ_c is the hyper-parameter that balances the two rewards.

5. Experiments

5.1. Experimental Setup

Data processing. Our algorithm is trained on the Gibson dataset [45] and evaluated on both the Gibson and Matterport3D datasets [7]. Our algorithm runs in the physically-realistic iGibson [35, 27] simulator. We adopt the TurtleBot model as our robot, which has a physical body and can be visually observed in the simulator to create a more realistic multi-robot scenario. To demonstrate the robustness of our algorithm on training scenes and the extraordinary generalization ability of our algorithm to novel scenes, we collect only 9 scenes randomly sampled in the Gibson dataset for training[3] and use the other scenes for evaluation.

Termination criterion. We aim for complete map construction, however, in the iGibson environment, all the robots have physical bodies and get stuck occasionally [47]. Hence following [16], our algorithm terminates when there is no accessible frontier in the environment.

Parameter setting. $X = 480$; $Y = 480$; $N_l = 3$; $\lambda_a =$ 12.5 degrees; $\lambda_r = 3$ meters; $\lambda_c = 0.005$. Each cell in the grid represents a $0.01m^2$ region. The bandwidth between robots is 10-20 KB. The $move\_forward$ action advances 6.5 cm, and the $turn\_left/turn\_right$ action rotates 12.5 degrees. The horizon of the planning cycle is 25 steps.

Evaluation metrics. We evaluate the map completeness via the coverage ratio (Cov. (%)) that calculates the percentage of explored open space over the ground truth open space in the environment. We measure the time efficiency (Time (#steps)) as the number of steps taken for map construction.

5.2. Compared Approaches

We compare our algorithm with several state-of-the-art multi-robot active mapping approaches (Greedy, VorSEG, mTSP, CoScan) [16, 41, 2, 19], which are mostly frontier-based and rely on the geodesic distance to choose the goal position. Inspired by the past reinforcement learning solutions for single-robot active mapping [13, 8, 31], we also design two multi-robot baselines (ANS-DeCen, ANS-Cen) that directly regress the coordinates of goal positions. To justify the effectiveness of our global planner, we compare with these approaches by only replacing our global planner with the alternatives in these approaches while leaving the mapping, local planner, action controller and termination criterion the same for everyone. We elaborate on all these methods below.

- **Greedy** [41]. It assigns the frontiers to all the robots in a greedy manner. Each robot chooses the closest frontier among its assigned ones as the goal position.
- **VorSEG** [2]. It performs a Voronoi segmentation over all the frontiers. Each robot chooses the closest segmented region and scans its frontiers in a greedy manner.

[3] The training is once for all and takes about 1 day.

	Small Scenes ($< 35m^2$)		Middle Scenes ($35 - 70m^2$)		Large Scenes ($> 70m^2$)	
Method	Cov. (%)	Time (#steps)	Cov. (%)	Time (#steps)	Cov. (%)	Time (#steps)
Greedy [41]	98.7	420.6	98.8	669.4	99.4	1057.0
VorSEG [2]	98.5	350.8	98.8	570.3	99.6	1080.5
mTSP [19]	98.8	351.4	98.6	564.5	98.9	1029.0
CoScan [16]	98.6	304.2	99.0	496.1	99.5	985.0
NeuralCoMapping (Ours)	98.6	302.5 (-0.6%)	98.8	471.7 (-4.9%)	98.9	882.0 (-10.5%)

Table 1. Numerical results on the Gibson dataset [45]. Parentheses: %steps reduced against the best competitor (blue) for our algorithm (red).

	Small Scenes ($< 100m^2$)		Middle Scenes ($100 - 300m^2$)		Large Scenes ($> 300m^2$)	
Method	Cov. (%)	Time (#steps)	Cov. (%)	Time (#steps)	Cov. (%)	Time (#steps)
Greedy [41]	95.9	801.2	95.1	2043.1	91.3	3345.4
VorSEG [2]	95.5	652.7	94.7	1693.2	91.0	2852.0
mTSP [19]	96.6	712.5	95.6	1742.6	91.4	2963.6
CoScan [16]	97.1	581.7	96.1	1505.5	92.1	2781.0
ANS-DeCen	85.1	1860.4	59.9	3229.8	48.5	5639.6
ANS-Cen	89.7	1536.8	64.3	2781.3	52.2	4961.3
NeuralCoMapping (Ours)	96.7	506.1 (-13.0%)	96.0	1217.3 (-19.1%)	92.4	1874.8 (-32.6%)

Table 2. Generalization to the unseen Matterport3D dataset [7], which is consistently larger than the Gibson dataset. Note our algorithm is trained only on 9 scenes in the Gibson dataset, while the ANS variants are trained on the entire Gibson dataset.

- **mTSP** [19]. It constructs a frontier graph where the edge weights are defined as the geodesic distance between two frontiers, then it solves a multiple traveling salesman problem based on the frontier graph for the goal assignment.
- **CoScan** [16]. It performs a k-means clustering over all the frontiers and assigns each robot a frontier cluster by solving an optimal mass transport problem based on the geodesic distance. Then for each robot, it computes an optimal traverse path over its assigned frontiers.
- **ANS-DeCen**. Following the recent reinforcement learning solution, Active Neural SLAM (ANS) [8], we replace our global planner with their global policy, which learns a convolution neural network that consumes an egocentric local map and a geocentric global map as input, and regresses the goal estimation for path planning. We design a fully decentralized setting where multiple robots construct the map independently without communication.
- **ANS-Cen**. We design a simple centralized setting of the ANS [8] approach. To adapt to the global policy, we modify the input by stacking the local maps together with the information of all robots to fuse the global map and reform the output by regressing the goal positions of all the robots together.

5.3. Results

We run all the approaches in the 3-robot scenario. The learnable approaches (ANS variants and our approach) are both trained and evaluated with 3 robots. The two ANS variants directly extract features from the raw map with a convolution neural network, which is more sensitive to the variance of the map distribution and needs to be trained in a larger set of training scenes (72 scenes in ANS [8]), unlike 9 scenes in our setting. Hence it would be unfair for the ANS variants to directly compare to our approach with our Gibson train/test split. During the implementation, we choose to train the ANS variants on the entire Gibson dataset and compare with them directly on the Matterport3D dataset.

We demonstrate the results of our algorithm and the four multi-robot active mapping approaches on the Gibson dataset in Table 1. All the approaches are able to achieve the roughly complete map construction, as also observed in [16], while our algorithm achieves superior performance over all the other approaches regarding time efficiency. The scenes in the Gibson dataset are relatively small, where the time efficiency tends to saturate in the multi-robot scenario, hence the performance difference between our approach and the best competitor is less significant. In Table 2, when we evaluate the approaches on the Matterport3D dataset, which contains consistently larger scenes than the ones in the Gibson dataset, our algorithm exhibits more significant superiority compared to the best competitor, especially in the large area interval ($> 300m^2$) we save more than 30% steps. We count time step rather than running time for the efficiency evaluation, while we take only 0.04s for each global planning, which is much faster than the best competitor CoScan 0.4s. The results also demonstrate the outstanding generalization ability of our algorithm to novel indoor scenes. Note the ANS variants do not guarantee a complete map construction as their goal positions are regressed and may not lay on frontiers. Therefore, to obtain a reasonable number for comparison, we

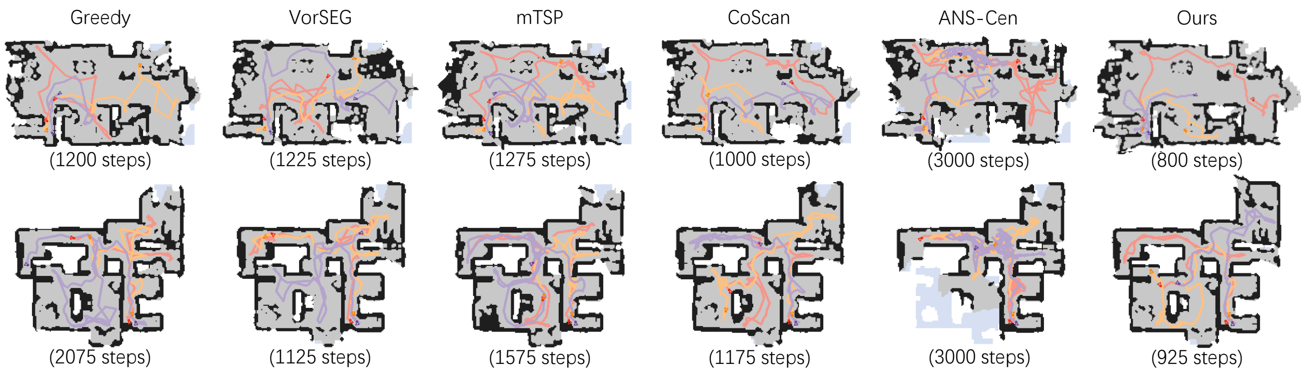

Figure 3. Visual results of our algorithm compared with state-of-the-art approaches on two scene samples from the Matterport3D dataset.

	Train with 2 robots		Train with 3 robots		Train with 4 robots	
Test number of robots	Cov. (%)	Time (#steps)	Cov. (%)	Time (#steps)	Cov. (%)	Time (#steps)
2 robots	96.7	1002.6	96.4	1005.0	97.1	1019.7
3 robots	96.7	680.0	96.3	661.7	96.1	683.7
4 robots	97.1	617.2	96.7	626.3	96.7	614.6

Table 3. Generalization to unseen numbers of robots evaluated on the Matterport3D dataset.

Method	Cov. (%)	Time (#steps)
Affinity: geodesic distance	96.9	785.5
Affinity: node correlation	94.3	1045.6
w/o history module	96.1	729.5
w/o obstacle-resistance	96.8	792.7
NeuralCoMapping (Ours)	96.3	661.7

Table 4. Ablation study on the Matterport3D dataset.

set a maximum horizon (5000 steps as default, 10000 steps for scenes $> 300m^2$), which is sufficiently large for most approaches to complete the map construction. We visualize the moving trajectories in Figure 3.

5.4. Generalization to Unseen Number of Robots

We evaluate how our algorithm generalizes to the unseen number of robots in Table 3. Our algorithm is trained with 2, 3, and 4 robots separately, and evaluated with different robot numbers on the Matterport3D dataset. From the results, we observe that our algorithm achieves very close time efficiency to its upper bound (trained and evaluated on the same number of robots). We consider such a good generalization ability mainly stems from three aspects. 1) Our entire framework contains only one learnable module, the global planner, while the other modules are non-learnable and can naturally generalize to the unseen number of robots. 2) The multiplex graph neural network in the global planner decomposes the goal estimation into many small and independent tasks, where the robot nodes are only partially involved in the entire network. 3) For the inter- and intra- graph operations in mGNN, the node feature is updated as the weighted sum of its neighborhood features, and hence relatively invariant to the number of its neighborhood (robots).

5.5. Ablation Study

We conduct an ablation study to evaluate the importance of each component of our algorithm to the multi-robot active mapping problem, as shown in Table 4. We firstly validate the design of the affinity matrix, by replacing the neural distance (edge feature) with 1) the geodesic distance between robots and frontiers and 2) the node correlation computed as the dot product between robot and frontier node features. We further justify the effectiveness of the history node module in mGNN and the obstacle-resistance strategy in the local planner by removing them separately from the entire framework for ablation study. Experimentally, we observe that our full framework achieves the best time efficiency. One of the major arguments in this paper is that the pure geodesic distance is not the optimal measurement to choose a reasonable goal position, which motivates our algorithm to learn the neural distance via reinforcement learning. Such an argument is validated in the experiment, where our neural distance is superior to the pure geodesic distance by a large margin.

6. Conclusion

In this work, we propose a novel multi-robot active mapping algorithm to achieve efficient and complete map construction. We formulate the problem as neural bipartite graph matching, which is solved via the proposed multiplex graph neural network and a differentiable linear assignment layer. The entire framework is optimized by maximizing the long-term value via reinforcement learning.

Acknowledgements. We thank the anonymous reviewers for their valuable comments. This work was supported by NSFC (62161146002).

References

[1] Shi Bai, Jinkun Wang, Fanfei Chen, and Brendan Englot. Information-theoretic exploration with bayesian optimization. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1816–1822. IEEE, 2016.

[2] Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Multi-robot coverage and exploration on riemannian manifolds with boundaries. *The International Journal of Robotics Research*, 33(1):113–137, 2014.

[3] Frederic Bourgault, Alexei A Makarenko, Stefan B Williams, Ben Grocholsky, and Hugh F Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 1, pages 540–545. IEEE, 2002.

[4] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005.

[5] Tibério S Caetano, Julian J McAuley, Li Cheng, Quoc V Le, and Alex J Smola. Learning graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(6):1048–1058, 2009.

[6] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. Tare: A hierarchical framework for efficiently exploring complex 3d environments. In *Robotics: Science and Systems Conference (RSS), Virtual*, 2021.

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[8] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations (ICLR)*, 2020.

[9] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33, 2020.

[10] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12875–12884, 2020.

[11] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(4):1–16, 2013.

[12] Kevin Chen, Junshen K Chen, Jo Chuang, Marynel Vázquez, and Silvio Savarese. Topological planning with transformers for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11276–11286, 2021.

[13] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *International Conference on Learning Representations*, 2019.

[14] Micah Corah and Nathan Michael. Efficient online multi-robot exploration via distributed sequential greedy assignment. In *Robotics: Science and Systems*, volume 13, 2017.

[15] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics 2017 (TOG)*, 2017.

[16] Siyan Dong, Kai Xu, Qiang Zhou, Andrea Tagliasacchi, Shiqing Xin, Matthias Nießner, and Baoquan Chen. Multi-robot collaborative dense scene reconstruction. *ACM Transactions on Graphics (TOG)*, 38(4):1–16, 2019.

[17] Christian Dornhege and Alexander Kleiner. A frontier-void-based approach for autonomous exploration in 3d. *Advanced Robotics*, 27(6):459–468, 2013.

[18] Jan Faigl and Miroslav Kulich. On determination of goal candidates in frontier-based multi-robot exploration. In *2013 European Conference on Mobile Robots*, pages 210–215. IEEE, 2013.

[19] Jan Faigl, Miroslav Kulich, and Libor Přeučil. Goal assignment using distance cost in multi-robot exploration. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3741–3746. IEEE, 2012.

[20] Christian Häne, Torsten Sattler, and Marc Pollefeys. Obstacle detection for self-driving cars using only monocular cameras and wheel odometry. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5101–5108. IEEE, 2015.

[21] Dirk Holz, Nicola Basilico, Francesco Amigoni, and Sven Behnke. Evaluating the efficiency of frontier-based exploration strategies. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8. VDE, 2010.

[22] Jiayi Huang, Mostofa Patwary, and Gregory Diamos. Coloring big graphs with alphagozero. *arXiv preprint arXiv:1902.10162*, 2019.

[23] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[24] Ayoung Kim and Ryan M Eustice. Active visual slam for robotic area coverage: Theory and experiment. *The International Journal of Robotics Research*, 34(4-5):457–475, 2015.

[25] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[26] Miroslav Kulich, Jan Faigl, and Libor Přeučil. On distance utility in the exploration task. In *2011 IEEE International Conference on Robotics and Automation*, pages 4455–4460. IEEE, 2011.

[27] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *Conference on Robot Learning (CoRL)*, 2021.

[28] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[29] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011.

[30] Carlos Nieto-Granda, John G Rogers III, and Henrik I Christensen. Coordination strategies for multi-robot exploration and mapping. *The International Journal of Robotics Research*, 33(4):519–533, 2014.

[31] Santhosh K Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. In *European Conference on Computer Vision*, pages 400–418. Springer, 2020.

[32] María L Rodríguez-Arévalo, José Neira, and José A Castellanos. On the importance of uncertainty representation in active slam. *IEEE Transactions on Robotics*, 34(3):829–834, 2018.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[34] James A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[35] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne P. Tchapmi, Micael E. Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[36] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[37] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and systems*, volume 2, pages 65–72, 2005.

[38] Quinlan Sykora, Mengye Ren, and Raquel Urtasun. Multi-agent routing value iteration network. In *International Conference on Machine Learning*, pages 9300–9310. PMLR, 2020.

[39] Wennie Tabib, Micah Corah, Nathan Michael, and Red Whittaker. Computationally efficient information-theoretic exploration of pits and caves. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3722–3727. IEEE, 2016.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[41] Arnoud Visser, Julian De Hoog, Adrian Jiménez-González, and J-R Martinez de Dios. Discussion of multi-robot exploration in communication-limited environments. In *Workshop" Towards Fully Decentralized Multi-Robot Systems: Hardware, Software and Integration" at the ICRA Conference*. Citeseer, 2013.

[42] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3056–3065, 2019.

[43] Barry Brian Werger and Maja J Matarić. Broadcast of local eligibility for multi-target observation. In *Distributed Autonomous Robotic Systems 4*, pages 347–356. Springer, 2000.

[44] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[45] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.

[46] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997.

[47] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16117–16126, 2021.

[48] Weifeng Zhang, Jingwen Mao, Yi Cao, and Congfu Xu. Multiplex graph neural networks for multi-behavior recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2313–2316, 2020.