Learning to search & track dynamic targets with graph representations

Shijie Cong

Delft University of Technology

TU Delft

Learning to search & track dynamic targets with graph representations

by

Shijie Cong

to obtain the degree of Master of Science
at the Delft University of Technology,

Student number: 5156734
Thesis committee: Dr. Javier Alonso-Mora, TU Delft, supervisor
Dr. Wendelin Böhmer, TU Delft
Álvaro Serra-Gómez, TU Delft, daily supervisor

Cover Image credits: Sam McGhee

TU Delft

Acknowledgement

The last couple of years in TU Delft has been a long journey in my life. When I was here for the first time, I was an exchange bachelor student and would never imagine what I was doing now. Teachers and mentors in the Cognitive Robotics department led me into the robotics world and changed my vision pretty much. I would like to first thank Dr. Javier Alonso-Mora and Álvaro Serra-Gómez for granting me the opportunity to work in the Autonomous Multi-Robots Lab. Their feedback and support not only answer my queries but also guide me on how to do research. I want to thank my parents for their support throughout my life. Finally, I would like to thank my friends, both by my side and those far away. Their accompany helps me break through those difficulties. In my way, I love you all.

Shijie Cong
Delft, May 2023

Summary

Autonomous robots have been widely applied to search and rescue missions for information gathering about target locations. This process needs to be continuously replanned based on new observations in the environment. For dynamic targets, the robot needs to not only discover them but also keep tracking their positions. Previous works focus on either searching for static targets or tracking dynamic targets given the number of targets and their initial positions. However, the prior information including targets not moving and initial target states can be difficult to obtain in reality. There are also some efforts to solve the search and tracking task jointly by switching between the search mode and the track mode or designing hybrid heuristics. But these methods cannot account for the effect of target movement during the search process, and the trade-off between search and tracking is sensitive to the heuristics.

To overcome the limitations above, in this thesis, we propose a graph formulation of the search and tracking of an unknown number of dynamic targets. The search and tracking problem is decoupled into two parts: search for undiscovered targets and track discovered ones. The search objective is modeled by minimizing the uncertainty in the environment evolving according to a diffusion mechanism and the tracking objective is formulated as minimizing the entropy of target belief distributions. Based on that, we design a novel graph neural network architecture, trained via Reinforcement Learning, that outputs the next motion primitive for the robot to collect information in the environment. We first evaluate this framework in the pure search and the pure tracking tasks. The results show that our method outperforms a variety of baselines both when searching in small and medium-scale environments, and tracking multiple dynamic targets in medium-scale environments. Then the experiments of the search and tracking task validate that our method achieves a better trade-off under equally good search or tracking performance, and scales to a large number of targets.

Contents

List of Figures

List of Tables

1 Introduction

1.1. Motivation

Search-and-rescue (SAR) scenarios often involve a race against time. For example, when someone is lost in a forest for a long time, first responders must locate the victim as quickly as possible. However, rescuers face several challenges in such time-sensitive situations, including physical limitations, hazardous environmental conditions, and large operational areas. To aid human rescuers, valuable information about the environment can be provided to reduce the places that they must search. In this regard, autonomous robots are a strong candidate for support as they can be deployed to various settings and reduce the risks that human workers would face otherwise [6]. By automating the environment information gathering process, robots free up scarce human rescue resources for more critical tasks such as victim assistance.

Gathering information about victim locations, including victim search and tracking, is vital in a SAR mission. This task has been explored from different perspectives, such as state estimation [40], sensory management [9], and decision-making [15]. The first two concern the perception process, whilst the last one considers the problem of where to make the next information collection. The effective deployment of the robot for information gathering is the topic of active perception [4]. To reduce the time it takes to localize humans (i.e. targets), active perception adapts where to perceive according to the information it receives compared to the passive one. For example, when searching for a victim lost in the forest, the robot is faced with an unexplored environment with a static or dynamic target. This requires it to make reactive decisions about where to go next according to environment observations such that the information needed to localize the target is maximized.

One main approach to solve this problem is to decrease the uncertainty of the entire environment [39][30]. These methods try to explore more unobserved areas in the environment to discover the target. However, this is only feasible for static targets since a dynamic target can move from an unexplored region into a previously explored one. For a dynamic target, localization not only requires discovering it but also updating its location over time. There have been research efforts in tracking single and multiple dynamic targets [15][13]. Current methods rely on assumptions that the number of targets is known and initial target states are available as prior information as well [36][2]. However, such assumptions overlook some limitations that cannot be ignored in reality. They do not consider situations when such prior information is difficult to obtain, such as when the only available information is an unknown number of victims got lost in a forest.

State-of-the-art planning-based information gathering methods have to face the challenge of computational complexity due to their planning horizon [20][2]. Myopic methods release the computational burden but also fail to achieve optimal target localization efficiency. For instance, a myopic approach can fail to collect information from distant areas in the environment. To improve the performance, non-myopic methods sacrifice computation speed to plan in a long horizon for optimized performance in terms of energy cost or gained information [10][25]. To improve execution efficiency without sacrificing localization performance, some attention is given to approaches based on Deep Reinforcement Learning (DRL). Recent works use DRL to train a policy to maximize the expected information gain of future observation [15][13]. Thus a trade-off between planning horizon and online computation efficiency is

achieved. But still, these methods only solve problems with the assumptions mentioned before. The research gap between DRL and target search and tracking is huge and worthwhile for further exploration.

Another limitation of existing methods is that the representation of the environment only captures low-level information. In previous search and tracking works, the environment is mostly represented as a discrete grid map [26][15][21]. Each cell in the map describes the information of a small area and does not interact with others. Recent advances in target search with semantic information prove that relational/topological information between landmarks in the environment improves the situational understanding of the robot and thus its target localization performance [1][32]. Moreover, to deploy a robot in a large-scale environment with more targets, its decision-making needs to reason about information with increased complexity. Thus, there is still much room for further exploration in the use of autonomous robots for target information gathering in SAR.

1.2. Research objective and contribution

In this thesis, to alleviate the limitations mentioned above, we focus on solving the problem of target search and tracking in a large-scale environment with a DRL policy. The main research question is formulated as follows:

How can an autonomous robot contribute to searching and tracking an unknown number of dynamic targets?

To solve this problem, the following research questions will be addressed:

1. **How to account for the movement of dynamic targets during the search process?**
   Compared with static ones, dynamic targets can move to areas that previously observed by the robot. This question addresses the importance of revisiting observed areas to search for unobserved targets.

2. **How to trade-off between target search and target tracking to maximize the total information gain?**
   This question aims to study the exploration and exploitation problem between searching for undiscovered targets and focusing on tracking discovered targets when the number of targets is not known as prior.

3. **How to encode target search and tracking related information in the environment representation?**
   This question addresses modeling the target search and tracking mathematically and formulating it as part of the environment representation.

The main contributions of this research to the state-of-the-art are:

1. The design of a novel graph formulation of the search and tracking of an unknown number of targets. An entropy diffusion mechanism based on heat diffusion is applied to capture the effect of moving targets.

2. We present a novel graph neural network architecture to learn a navigation policy.

3. We assess the performance in the search and tracking task. The architecture is also verified to generalize to pure search and pure tracking tasks.

1.3. Thesis overview

The work in this thesis is presented as follows. In chapter 2, the relevant literature on Active perception for target search and target tracking, and DRL with graphs are discussed. Chapter 3 formulates the search and tracking problem with a graph representation. Chapter 4 presents our approach to solve the search and tracking problem with the graph formulation. Chapter 5 contains three main parts: the baselines to compare, the test experiment setup and the evaluation metrics, and the experiment results. Finally, we conclude in Chapter 6.

2

Related work

The search and tracking task has two primary objectives: search for undiscovered targets and track discovered ones. Although numerous studies have addressed one of them, only a minority have tackled both simultaneously. Graph neural networks have gained significant popularity in recent years and are now also applied to RL. Our contributions build upon recent work in active perception for search and tracking and DRL with spatial graphs. In this chapter, relevant research about active perception for target search and tracking is reviewed in section 2.1. Then learning-based methods with graph representations in motion planning are investigated in section 2.2.

2.1. Active perception for target search and tracking

Active perception is the automation and optimization of the perception process through the use of a dynamic system with actuation and sensing capabilities [4]. In this section, we first review active perception strategies for target search and target tracking, At the end of this section, we discuss some efforts that have been made to extend from target tracking to search and tracking.

2.1.1. Active target search

Active target search (ATS) is defined as locating static targets in the environment by collecting information about the environment [29]. It requires the robot to make proper viewpoint selections to locate targets efficiently. When the map of the environment is known and there is no additional information about target distribution, which means the target is equally possible to appear anywhere in the environment, ATS is the same as a coverage path planning problem. This problem has been widely addressed by the robotics community [34][22].

Myopic approaches choose one-step look ahead actions at the current time step. Two important parts of these methods are the generation of candidate viewpoints and the criteria to evaluate each of them. The candidates can be derived by sampling near the robot [11] or motion primitives [7]. The difference is that the former requires the support of another local path planner to navigate to the sampled viewpoint. As for the candidate viewpoint evaluation, there have been many efforts to propose utility functions for measuring the information gain of each candidate. González-Baños *et al.* [11] use the product of the observable area and the exponential function of path length to balance the coverage gain and the navigation cost. Amarjeet Singh *et al.*[38] compute the information gain through the mutual information between observed and unobserved regions. Due to the short planning horizon, these methods are computationally efficient (especially when deploying to a team of robots [7]) but sacrifice in terms of search time or energy efficiency.

Non-myopic methods, on the other hand, aim to maximize the information objective over a long time horizon. These methods rely on global [3] or receding horizon [46] optimization approaches. Bähnemann *et al.* [3] formulate the coverage problem in a known environment as an equality generalized traveling salesman problem (E-GTSP) on a boustrophedon cell decomposition graph. Cells in the graph are divided into clusters and the goal is to find the shortest tour path that visits one node in each cluster. Then an exhaustive exact solution is derived based on Dijkstra. The computation time of this method is limited by the size of the graph since the amount of edges grows quadratically with the number of

nodes, thus it cannot be online deployed. Leonardo Zacchini *et al.* [46] propose an online method by constructing a two-layer planner where a higher layer generates viewpoints with rapidly-exploring random tree (RRT) in a receding-horizon manner and a lower layer planning a feasible path considering the kinematic constraints with RRT*. Then the next best viewpoint is selected based on the expected information gain and the path length. However, in their experiments, with the same mission time, this approach cannot outperform the classic lawnmower path in terms of coverage.

DRL-based methods have been addressed more recently due to their execution efficiency and ability to act considering historical observations [48][26]. Policies learned by DRL build a mapping from states to actions instead of searching over action space explicitly at each time step. In this way, a promising trade-off between fast execution and information gain performance can be achieved. Delong Zhu *et al.* [48] attempt to learn to infer the global visiting order of the environment. To do this, two region candidates, one from a DRL model and the other from a greedy strategy are evaluated according to structural integrity. Then the better one is used to generate viewpoint candidates from which the next best viewpoint is obtained. This suggests that the DRL model doesn't always generate reasonable candidates and needs further improvement. Max Lodel *et al.* [26] propose a hierarchical framework that combines a high-level viewpoint recommendation with a low-level trajectory planner. The former is done by a DRL policy and the latter uses a model predictive control (MPC) module. The hierarchical structure frees the DRL agent from the burden of learning local navigation behaviors (e.g. collision avoidance), thus improving the exploration performance.

Our method builds upon DRL to learn a non-myopic navigation policy to maximize the information gain. Previous work focuses on searching for static targets. By modeling undiscovered dynamic targets as sources of uncertainty in the given environment, we expand the applied fields into searching for dynamic targets. Compared with the hierarchical DRL-MPC structure [26], our graph formulation removes unreachable positions (e.g. occupied by obstacles) and unfeasible motion primitives from the graph representation. In our architecture, the policy only evaluates each feasible action independently and learns to recommend a reachable viewpoint even without the low-level controller.

2.1.2. Active target tracking

ATS mentioned in the previous section aims only to discover static targets. When dealing with detected dynamic targets, the objective shifts to estimating their locations, which is called active target tracking (ATT). Problems in ATT address challenges related to target motions including estimation of target states, non-myopic planning, cost function, and stochasticity of the targets.

Non-myopic ATT strategies can be divided into two categories: search-based methods [2][36][35], and optimization-based methods [33][25]. Search-based approaches create a search tree of potential robot trajectories using known system models for the robot, targets, and observation. Nikolay Atanasov *et al.* [2] propose the Reduced Value Iteration (RVI) algorithm to compute an open-loop control policy that maximizes the objective of mutual information. A pruning technique is applied to reduce the size of the search tree and guarantee finite execution time while ensuring sub-optimality. Brent Schlotfeldt *et al.* [36] present the Anytime Reduced Value Iteration (ARVI) makes improvements upon RVI by eliminating the need to adjust pruning parameters in RVI while reducing the computation load by reusing computations from prior iterations.

Both RVI and ARVI only have a sub-optimal guarantee. To get optimal performance and reduce the size of the search tree, Brent Schlotfeldt *et al.* [35] use the A* search algorithm with a novel heuristic based on the bounding of a Kalman filter estimate covariance matrix and prove the heuristic admissible. However, this method can not deal with continuous motion robots since the search tree can grow much larger. Thus it needs a roadmap to deploy A* and another low-level controller to execute the trajectory.

Optimization-based methods predict the distribution of the target over a multi-step horizon. Ryan *et al.* [33] compute the entropy of the estimated target distribution by the sequential Monte Carlo method in the context of particle filtering. However, due to the computation complexity of the Monter Carlo method, this approach cannot be deployed in real-time. To improve the computation efficiency, Chang Liu *et al.*[25] present a two-stage sequential MPC-based approach. Similar to previous works, they supplement an objective about the entropy of the estimation of the target to the MPC cost function. The objective is defined based on the trace of the Kalman filter.

These ATT methods have a strict limitation that the number of targets to track is not only known as a prior but also equal to the number of robots. That means even if the robot moves fast, its mobility advantage can not help to track multiple targets. Furthermore, an accurate dynamic model of targets

is needed to predict target trajectory. Inadequate knowledge of dynamics will navigate the robot to someplace where the target does not exist. However, due to practical limitations, the true target model is hard to get in reality.

Heejin Jeong *et al.* [15] present a DRL method that learns a robust policy to deal with noisy measurements and inaccurate target models. It also has the capability to deal with multiple targets. However, like all the other ATT methods, it requires the initial target belief. Although they increase the stochasticity with larger initial uncertainty and initialize the belief away from the true state, it is like having the robot compensate for a pre-existing mistake.

Our method is inspired by the work of Heejin Jeong *et al.* [15] that computes the reward based on the cumulative mutual information about target distributions. By considering the search part, we do not have to make assumptions that the number of targets and initial target positions are known, which are obtained by the robot itself during target search.

2.1.3. Search and tracking

In the sections above, strategies for ATS and ATT are investigated respectively. As is introduced in chapter 1, in realistic situations these two tasks need to be solved jointly when prior information about the targets is not accessible, which is known as search and tracking (SAT). In this process, the robot is expected to first explore the environment and the detection of a target provides initial information for tracking it. For a single target, the solution of SAT is just to first explore and then keep tracking it [10][8]. However, when dealing with multiple targets, the robot needs to make the trade-off between searching for undiscovered targets and tracking discovered ones since the perception capability of the robot is limited.

Some ATT methods try to extend the capability from pure tracking to discover the target when the prior belief is incorrect [25][15]. But as stated in the previous section, these solutions are correcting an existing error and can not start with empty prior. Heejin Jeong *et al.* [15] utilize a visit frequency to encourage the robot to explore unvisited areas. But the visit frequency concerns each cell in the grid map separately and doesn't consider the moving behavior of targets.

Jonathan P How *et al.* [12] introduce the concept of the mode switch, which divides the SAT task into two sub-tasks. They solve the problem of the unknown number of targets by assuming there's always at least an undiscovered target. Thus the robot keeps working on the search mode and switches to the tracking mode either a new target is discovered or the uncertainty of a previously detected target exceeds a certain threshold.

Alberto Dionigi *et al.* [8] follow the idea of the mode switch and learn separate DRL policies for target search and target tracking respectively. However, this method has several limitations. First, it focuses on SAT with a single target in the environment and thus only switches from the search mode to the track mode once the target is detected. Second, it assumes that the target is static during the search process to compute the probability of target appearance in the environment.

Instead of doing explicit rule-based mode switches, another group of methods puts efforts into balancing the two sub-tasks by designing hybrid heuristics. Ryan R Pitre *et al.* [31] present an objective function to combine the conflicting objectives about target search and target tracking. The objective function is to maximize the sum of the trace of the estimated covariance matrix over all discovered targets. They propose that this method concerns the two sub-tasks at the same time. But an obvious problem is that a poor estimation with high uncertainty can also bring more benefits according to their definition. This is reflected in their experiment results that their method has a natural preference for target search rather than target tracking.

Xianfeng Li *et al.* [23] concern the two objectives separately and propose a combined observation profit metric to account for their impact. It evaluates the target tracking profit by its relative location in the Field of View (FoV) and deals with the exploration part in a way similar to the observation frequency [15]. In order to solve the trade-off, the weights of these two terms need to be calibrated. Furthermore, this method assumes the total number of targets is known as well.

To relax the limitation of the known number of targets, Peng Yan *et al.* [44] train a DRL policy that incorporates the exploration rate of the environment and the observation rate of each target in the reward function. The trade-off between exploration and exploitation is expected to be learned. Nevertheless, one shortcoming of this method is that it only considers the observation rate instead of the uncertainty of the target. This means two consecutive observations at the beginning are considered to be equivalent to two observations at a certain time interval although the former behavior results in

a larger target state uncertainty. Furthermore, this method assumes the environment is obstacle-free, which is another limitation that needs further improvement.

The hybrid objectives mentioned above are all composed of a weighted sum of two independent metrics about target search and target tracking. Julius Ibenthal *et al.* [14] take the detection uncertainty and the occlusion effect of obstacles into account and propose a set-membership estimation method to maintain set estimates about identified, non-identified, and non-detected targets from a geometric perspective. This approach successfully uses a metric of the volume of the set-membership to evaluate target search and target tracking performance uniformly. However, the search part still does not consider the target moving behavior.

To solve the SAT task, previous methods either assume targets keep static during the search process or assign higher priority to positions not observed for a longer time. The former ignores target movements and can fail to discover targets moving into observed areas while the latter considers the importance of each area independent from others. This means the search process is endless since there are always regions not observed at the moment. However, we apply a diffusion mechanism to formulate the effect of undiscovered target movement during target search and avoid omissions.

2.2. Reinforcement Learning with spatial graph representations

Graph Neural Network (GNN) applications in RL have gained popularity in recent years. In this section, we mainly focus on methods that encode spatial graph observations to learn a navigation policy. Zambaldi *et al.* [47] solve a visual navigation task by encoding RGB image observations as fully connected graphs. Then the multi-head dot product attention [42] is applied to learn different sets of relations. Theoretically, each head can learn a unique set of edges with their own weights, thus the topological information can be abstracted. Considering the permutation-invariant property of graphs, this method assumes that the global position of each node is available.

Lu *et al.* [27] further apply RL with GNN to learn a navigation policy for searching a semantic object. Instead of using raw image inputs, this method constructs an abstract map of the environment and models it as a Markov network [17]. The co-appearance of different objects (e.g. a chair and a desk) is modeled as a probability distribution with a random vector describing the positions of objects and a joint probability capturing their relative positions. Then a GNN-based agent is applied to learn it. In this work, the knowledge graph is required to initialize the graph structure.

Zachary *et al.* [32], on the other hand, get rid of the dependence of prior knowledge graph and generates a hierarchical graph representation of the environment with 3D Dynamic Scene Graphs (DSG). An additive action layer containing nearby sampled nodes is connected to the scene graph for message passing. With a feature embedding vector computed from the action layer nodes, the RL agent learns the navigation policy to search for semantic objects. But still, like other works investigated in section 2.1.1, this method can only deal with static objects.

One comparable work to our research is presented by Ekaterina *et al.* [41] that models the environment as a lattice graph shown in figure 2.1 and applies it to a coverage task. To learn a non-myopic policy, this work stacks a number of GNN layers to get information from distant nodes. However, like CNNs, GNNs also suffer from the over-smoothing problem [45]. That means as the receptive field k increases, after k-hop message passing all node features tend to be the same. To solve this, the method concatenates the output feature of every GNN layer together to compute the final embedding.

Our work is inspired by the work of Ekaterina *et al.* [41]. However, we design a hierarchical lattice graph and reduce the receptive field required to reason information from distant nodes. Besides, we leverage the use of attention-based GNN layers [42] to encode the dynamic environment and learn the importance of different nodes.

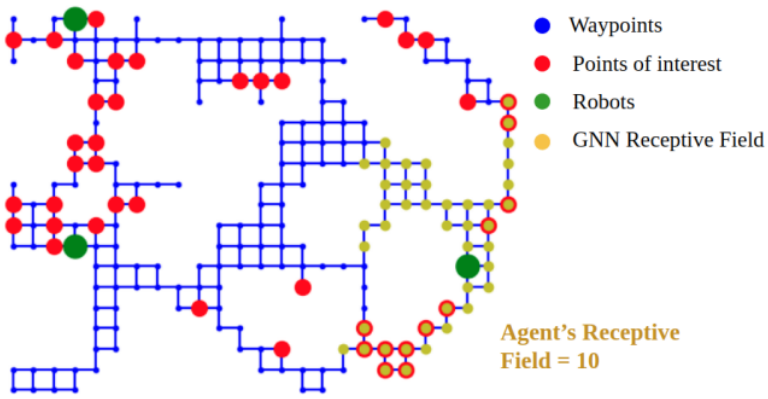


Figure 2.1: The spatial graph built in [41]

3

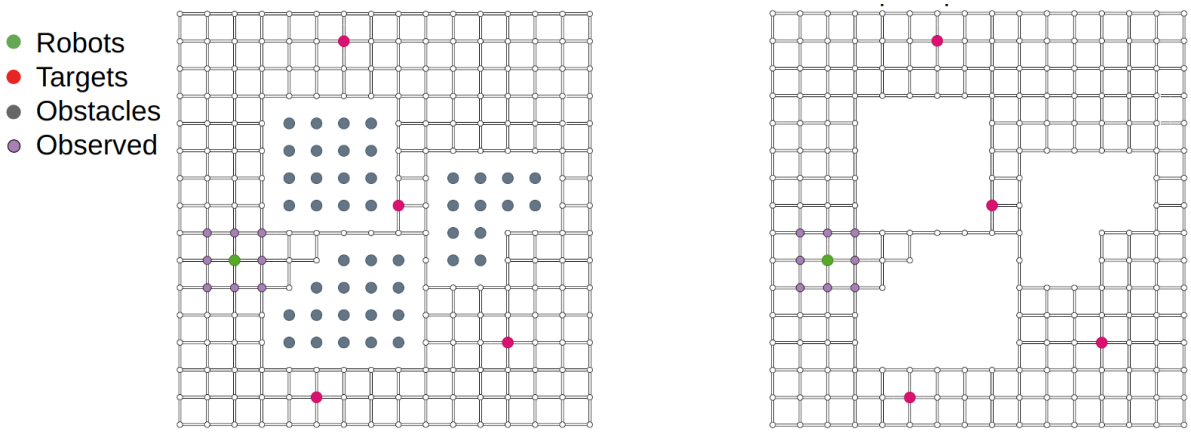
Problem formulation

Considering a robot that has to explore a 2D environment $\mathcal{W} \subset \mathbb{R}^2$ with static obstacles located at $\mathcal{O} \subset \mathcal{W}$. The main goal of the robot is to search for and keep track of the location of an unknown number of *M* dynamic targets $j \in \{1, ..., M\}$ within a given horizon F. The state of the robot is $\mathbf{x}(t) \in \mathcal{X} := \mathbb{R}^2$ whilst the state of target j is denoted by $\mathbf{y}_j(t) \in \mathcal{Y} := \mathbb{R}^2$. For all targets we have $\mathbf{y}(t) = [\mathbf{y}_1(t), ..., \mathbf{y}_M(t)]$. Information about how many targets to search and prior about target states is unknown to the robot.

The robot does not have access to the ground truth of target positions. Instead, the robot maintains a belief on each discovered target. We assume targets do not react to the robot, e.g. $\mathbf{y}(t+1)$ is independent of state $\mathbf{x}(t)$ and action $\mathbf{u}(t)$ of the robot. Hence, the belief distribution of $\mathbf{y}_j(t)$ according to observation history $\mathbf{z}_{1:t}$ is denoted as $B(\mathbf{y}_j(t)) = P(\mathbf{y}_j(t)|\mathbf{z}_{1:t})$. And $\mathcal{B}_t = \{B(\mathbf{y}_j(t))\}_{j=1,..,M}$ denotes the belief of all discovered targets.

To make observation z_t, we have the assumption that the robot is equipped with a sensing system that can detect targets within a certain distance. We assume that the map of the environment and the exact location of the robot are known to the robot.

Motivated by the success of motion planning with lattices [41], we construct a graph representation of the environment based on its coarse map. A grid of points is first initialized, then points within obstacles in the environment are removed and connections between adjacent nodes in free space are added as in Figure 3.1.



(a) Coarse grid map

(b) Lattice representation

Figure 3.1: Graph representation of the environment

The graph is defined as a 2-tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents a set of nodes $\{v_k\}_{k=1,...,N}$, $\mathcal{E}$ is a set of edges $\{e_{s,r}\}$. In our case, the edge $e_{s,r} = (v_s, v_r)$ is directed, pointing from the source node v_s to the receiver node v_r. As is explained in subsection 2.1.3, SAT has two sub-objectives: searching for undiscovered targets and tracking discovered ones. To model the search task, we associate each

node k with a Bernoulli random variable $Q_{k,t} \in \{0, 1\}$, where $Q_{k,t} = 1$ indicates k is occupied by an undiscovered target and the search task is to minimize the uncertainty of undiscovered target appearance $\{p(Q_{k,t} = 1)\}_{k=1,...,N}$. We assume that the targets follow a random walk dynamics that there are five actions with equal probability: $\{$*left*, *right*, *up*, *down*, *stay still*$\}$. Then the tracking task is converted to infer the discovered target states according to their belief distributions, which is modeled as discrete probability maps:

$$B(\mathbf{y}_j(t)) = [p_{j,1}(t), ..., p_{j,N}(t)] \tag{3.1}$$

Where $p_{j,k}(t)$ is the probability of target j appearance at node k, and $B(\mathbf{y}_j(t))$ is maintained by the prediction step and the update step according to the observation z_t:

$$B^p(\mathbf{y}_j(t+1)) = B(\mathbf{y}_j(t)) + B(\mathbf{y}_j(t))\hat{\mathbf{L}} \tag{3.2}$$

$$B^u(\mathbf{y}_j(t+1)) = [p^u_{j,1}(t+1), ..., p^u_{j,N}(t+1)] \tag{3.3}$$

$$\hat{L}_{kb} = \sum_a \hat{A}_{ka}\delta_{kb} - \hat{A}_{kb} \tag{3.4}$$

$$\hat{A}_{kb} = \frac{1}{d_k} \tag{3.5}$$

$$\delta_{kb} = \begin{cases} 1 & \text{if } k = b \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

$$p^u_{j,k}(t+1) = \begin{cases} 1 & \mathbf{O}_{j,k,t+1} = 1 \\ 0 & \mathbf{O}_{k,t+1} = 1 \\ \eta(p_{j,k}(t)) & \text{otherwise} \end{cases} \tag{3.7}$$

Where $B^p(\cdot)$ and $B^u(\cdot)$ are the predicted and updated belief respectively, δ_{kb} is the Kronecker delta[19], d_k is the degree of node k, $\mathbf{O}_{j,k,t}$ and $\mathbf{O}_{k,t}$ are indicator functions from the observation that $\mathbf{O}_{j,k,t} = 1$ means target j is observed at node k at time t and $\mathbf{O}_{k,t} = 1$ means node k is in the FoV of the robot at time t, $\eta(\cdot)$ is a normalization operator to keep $\sum_{k=1}^{N} p_{j,k}(t) = 1$.

We assume that the velocity of the robot is n_{vel} faster than the target, thus the prediction is carried out every n_{vel} steps. At each time step, the update process iterates according to the observation z_t. Recent ATT methods [36][15] represent target state distributions with continuous belief models (e.g. the Gaussian distribution), by which only an observation including target j contributes to update its belief (e.g. through a Kalman filter). However, in our formulation, an observation confirming non-occupancy by targets also collects valid information for updating $B(\mathbf{y}_j(t))$.

We use the entropy to capture the uncertainty of $\{p(Q_{k,t} = 1)\}_{k=1,...,N}$ and $\mathcal{B}_t$. With the assumption that target states are independent from each other:

$$H(\mathbf{Q}_t, \mathcal{B}_t) = H(\mathbf{Q}_t) + H(\mathcal{B}_t) \tag{3.8}$$

Where

$$H(\mathbf{Q}_t) = \sum_{k=1}^{N} H(Q_{k,t}) \tag{3.9}$$

$$H(\mathcal{B}_t) = \sum_{j=1}^{M} H(B(\mathbf{y}_j(t))) \tag{3.10}$$

$$H(Q_{k,t}) = \begin{cases} 0 & \text{if } \mathbf{O}_{k,t} = 1 \\ g(H(Q_{k,t-1})) & \text{otherwise} \end{cases} \tag{3.11}$$

Where $g(H(Q_{k,t-1}))$ is the dynamics of $H(Q_{k,t-1})$.

An explicit advantage of graphs is that information can be propagated between neighbor nodes. As is mentioned in section 1.2, one challenge when searching for dynamic targets is that targets can move to previously observed areas. Thus the uncertainty of unobserved nodes is able to flow to their neighbor observed ones. On the other hand, the uncertainty of nodes with higher entropy should not decrease since no additional information is obtained. Hence, this kind of entropy diffusion mechanism is supposed to be unidirectional and the entropy can only propagate from higher values toward lower

values. More intuitively, the entropy of unobserved nodes will not get reduced if their adjacent nodes have lower entropy while nodes get income entropy if their neighbors have lower ones.

Inspired by the 2D heat diffusion equation [28], $g(H(Q_{k,t}))$ is modeled as following:

$$\begin{aligned}\frac{\mathrm{d}H_{k,t}}{\mathrm{d}t} &= C\sum_b A_{kb}(H_{b,t} - H_{k,t}) \\ &= C(\sum_b A_{kb}H_{b,t} - \sum_b A_{kb}H_{k,t}) \\ &= C(\sum_b A_{kb}H_{b,t} - d_k H_{k,t})\end{aligned} \tag{3.12}$$

Where $H_{k,t}$ is short for $H(Q_{k.,t})$, C is the diffusion coefficient that is related to the estimated target speed and we assume is given as prior, A_{kb} is the edge weight between node k and b in the adjacency matrix **A**. In our model, A_{kb} is computed by the reciprocal of the distance between node k and b.

Given that

$$d_k H_{k,t} = \sum_b \delta_{kb} d_b H_{b,t} \tag{3.13}$$

Then we have:

$$\begin{aligned}\frac{\mathrm{d}H_{k,t}}{\mathrm{d}t} &= C\sum_b (A_{kb} - \delta_{kb}d_b)H_{b,t} \\ &= -C\sum_b L_{kb}H_b\end{aligned} \tag{3.14}$$

Where $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the graph Laplacian matrix, **D** is the degree matrix of the graph. Therefore:

$$H_{k,t+1} = H_{k,t} - C\Delta t\sum_b L_{kb}H_{b,t} \tag{3.15}$$

To model the unidirectional diffusion process from higher entropy to lower one, we have:

$$g(H_{k,t}) = H_{k,t} + dH_{k,t} \tag{3.16}$$

$$dH_{k,t} := \max\left\{0, -C\Delta t\sum_b L_{kb}H_{b,t}\right\} \tag{3.17}$$

Where $H_{k,0}$ is initialized as 1 for all node k.

Based on the formulations above, the objective function is defined here:

$$\max_{u_{0:t-1}} \sum_{t=1}^{F} I(\mathbf{Q}_t, \mathcal{B}_t; z_t|\mathbf{z}_{0:t-1}) \tag{3.18}$$

$$s.t. \quad \mathbf{x}_t = s_r(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{3.19}$$

$$\mathbf{y}_{j,t} = s_t(\mathbf{y}_{j,t-1}) \tag{3.20}$$

$$z_t = h(\mathbf{x}_t, \mathbf{y}_t) \tag{3.21}$$

Where $s_r(\cdot)$ and $s_t(\cdot)$ are the dynamics of the robot and targets respectively, N is the number of nodes in the graph, $I(\cdot)$ represents the mutual information.

The mutual information between $(\mathbf{Q}_t, \mathcal{B}_t)$ and observation z_t is the reduction of the conditional entropy in $(\mathbf{Q}_t, \mathcal{B}_t)$ by z_t:

$$I(\mathbf{Q}_t, \mathcal{B}_t; z_t|\mathbf{z}_{0:t-1}) = H(\mathbf{Q}_t, \mathcal{B}_t|\mathbf{z}_{0:t-1}) - H(\mathbf{Q}_t, \mathcal{B}_t|z_t, \mathbf{z}_{0:t-1}) \tag{3.22}$$

4

Methodology

In the previous chapter, we propose a graph formulation of the search and tracking problem. In this way, the search and tracking objective is modeled nodewise in the lattice graph. From now on, we solve the search and tracking problem based on a graph with encoded information. In this chapter, we first provide the structure of our proposed solution pipeline, including the graph representation for the SAT task and the architecture of the GNN-based policy. Then the training setup of our method is introduced.

4.1. Proposed solution

Figure 4.1 displays the general structure of our solution for a single target scenario. At time t, the robot gets an action a_t from the policy, and after executing it, the states of the robot and the target evolve to $\mathbf{x}_{t+1}$&$\mathbf{y}_{t+1}$. With the new observation z_{t+1}, the graph observation is updated to $\mathcal{G}_{t+1}$ and sent to the GNN-RL agent, then the whole process iterates. In the following sections, the graph observation and the policy structure are discussed in detail.

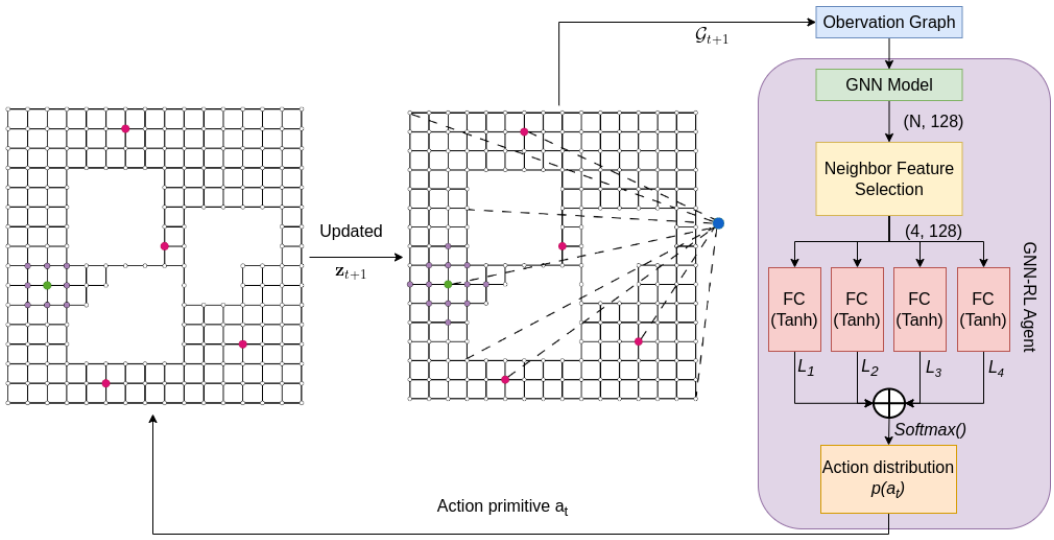


Figure 4.1: Solution structure

4.1.1. Graph representation for search and tracking

To solve this task, We construct a graph representation that the robot, targets, and all waypoints are included as part of a computation graph. The robot is considered a node and its action space is discrete: it chooses one nearby waypoint to move towards.

Besides, inspired by the hierarchical graph structure [32], we use a different approach to make the robot reason about information from distant nodes instead of stacking many (e.g. up to 15) GNN layers as the work from Ekaterina *et al.* [41]. As is shown in Figure 4.2, we add an abstract node outside the spatial graph and construct a two-layer hierarchy graph representation.

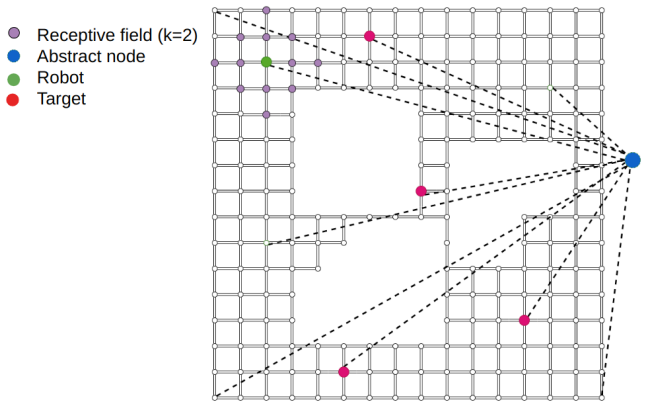


Figure 4.2: Hierarchy graph representation

The abstract node is connected to every node in the free space. Compared with the limited receptive field in the single-layer graph (shown in Figure 4.2, the robot now can get information from any place node, even the one in the diagonal corner. Although we do not utilize semantic information in this research, it can be regarded as working in a single room described by [32]. Each node $v_i \in \mathcal{V}$ is assigned a six-dimensional feature vector $\mathbf{v}_i^t$:

$$\mathbf{v}_i^t = \{\mathbb{1}_r, H(Q_{i,t}), \xi_k(\mathcal{B}_t), \mathbb{1}_a, d_{i,x}, d_{i,y}\} \tag{4.1}$$

Where $\mathbb{1}_r$&$\mathbb{1}_a$ are binary variables with 1 indicating node i is occupied with the robot and is the abstract node respectively, $d_{i,x}$&$d_{i,y}$ captures the position of node i relative to the robot node, $\xi_k(\mathcal{B}_t)$ is a feature to score the information about target belief:

$$\xi_k(\mathcal{B}_t) = \sum_{j=1}^{M} H(B(\mathbf{y}_{j,t}))\mathbf{1}(p_{j,k}(t)) \tag{4.2}$$

$$\mathbf{1}(p_{j,k}(t)) = \begin{cases} 1 & p_{j,k}(t) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

In our work, a square lattice is used, which means the distance between adjacent nodes is equal. Hence we do not consider the effect of edge features and edge weights.

4.1.2. Policy architecture

In this part, we present the architecture of the GNN-RL agent in Figure 4.1. The graph observation is first passed through the GNN model. Since we compute on a large graph with hundreds of nodes, it is challenging to capture minor node-wise feature changes like distinguishing a few different words in two almost identical articles. Therefore, instead of generating embeddings of the whole graph, we extract learned features from neighboring nodes of the robot (in our square lattice case there are maximum 4 neighboring nodes) and send them to a fully connected layer (FC) to get logits of different actions $[L_1, L_2, L_3, L_4]$. Then with a softmax function, the distribution of the discrete action space is derived.

Here we use the same policy network to evaluate each neighboring node individually. This is because the action space in our work is exactly the edges in the graph, hence the node with more valuable features would be preferred and get a higher score. Compared with the other way that evaluates all neighboring node features at the same time, our structure frees the burden to learn the mapping between node features and geometric movements.

Our GNN model structure is visualized in Figure 4.3. We name nodes in the first layer of the hierarchy graph as place nodes $\mathcal{V}_p = \{v_p\}$, the 1-hop neighboring nodes of the robot node as neighbor

nodes $\mathcal{V}_n = \{v_n | v_n \in \mathcal{N}_r\}$, and the abstract node as v_a. There are three kinds of edges in our graph model: $\mathcal{E}_{p_1p_2} = \{e_{p_1,p_2} | v_{p_1}, v_{p_2} \in \mathcal{V}_p\}$, $\mathcal{E}_{na} = \{e_{n,a} | v_n \in \mathcal{V}_n\}$, and $\mathcal{E}_{pa} = \{e_{p,a} | v_p \in \mathcal{V}_p\}$. It is intuitive that the first one is bi-directional while the latter two ones are uni-directional. In GNN, the message-passing mechanism is based on edges between nodes. We applied different propagation operators for different edges. For the first one, we apply the graph convolution network (GCN) operator [16] :

$$\mathbf{V}_p^l = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{V}_p^{l-1}\mathbf{W}_p^{l-1} \tag{4.4}$$

Where $\mathbf{V}_p^l$ is the feature vector of all place nodes at layer l, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denotes the adjacency matrix with inserted self-loops and $\hat{\mathbf{D}}$ is its diagonal degree matrix, $\mathbf{W}$ is the learned transformation. For the latter two kinds of message passing, we apply the graph attention network operator (GATv2) [43]:

$$\mathbf{v}_a^l = \sum_{q \in \mathcal{N}_a} \alpha_{a,q}\mathbf{v}_q^{l-1}\mathbf{W}_a^{l-1} \tag{4.5}$$

$$\alpha_{a,q} = softmax(\mathbf{a}^T LeakyReLU(\mathbf{W}_a^{l-1}[\mathbf{v}_a || \mathbf{v}_q])) \tag{4.6}$$

Where we have $\mathcal{N}_a = \mathcal{V}_n$ for layer 1 and layer 2, and $\mathcal{N}_a = \mathcal{V}_p$ for layer 3. Then each place node feature vector is concatenated with the abstract node feature vector and through a fully connected layer we get the final features for neighbor feature selection block in Figure 4.1. The general idea of this structure is that we first learn the embedding of local information nearby the robot. Then a global embedding of the entire environment is learned to extract valuable information from distant nodes conditioned on the local embedding.

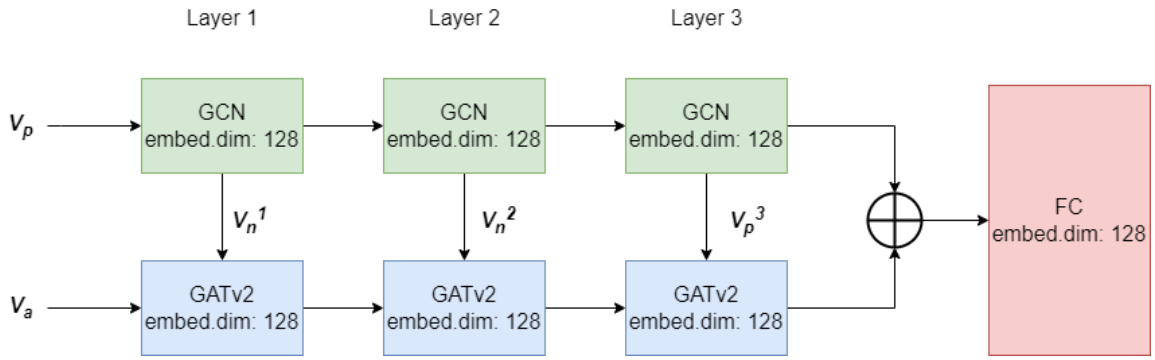


Figure 4.3: GNN model structure

The reward function is defined by the entropy reduction of the environment as Equation 3.18:

$$r_t = w_{search}(H(\mathbf{Q}_t) - H(\mathbf{Q}_{t-1})) + w_{track}(H(\mathcal{B}_t) - H(\mathcal{B}_{t-1})) \tag{4.7}$$

Where w_{search} and w_{track} are the weights that scale each term. In this way, we can change the tendency of the robot to prioritize one of the two parts.

4.2. Training setup

4.2.1. Applied tasks

We applied our framework to three tasks:

- The pure search task
- The pure tracking task
- The search and track task

In the pure search task, it is given that there are an unknown number of dynamic targets in the environment and the diffusion coefficient C is given as a prior according to the estimation of target speed. In this scenario, the robot needs to reduce the entropy of undiscovered dynamic targets in the environment (i.e. $\sum_{k=1}^{N} H(Q_{k,t})$).

In the pure tracking task, the initial position of each target is given as prior. Since the speed of the robot is faster than the target, it is achievable for the robot to track multiple targets. The objective of the robot is to keep track of discovered targets and reduce the entropy of their distribution in the environment.

Finally, for the SAT task, there is no prior about the number of targets in the environment or information about their positions. Prior about C is still assumed to be given. In this scenario, the robot is required to both explore the environment to find targets and observe discovered targets to track them.

4.2.2. Training conditions

Our policy is trained in a closed environment of $80 \times 80\ m^2$ with the resolution of the graph representation as $5.5m$ (i.e. a 15×15 square lattice). We consider the sensor on the robot to be omnidirectional with a sensing range of $8m$. The target follows a random walk model in chapter 3. The training is performed in randomly generated environments with cluttered obstacles as depicted in Figure 4.4 with the robot initialized at a random node in the graph. For the search task, considering the diffusion of entropy brings punishments (i.e. negative rewards) to the robot at the beginning stage, we randomly initialized the environment as either a sparse entropy source scenario ($\rho = 0.05$) or a dense entropy source scenario ($\rho = 1$). For the tracking task, the environment contains a random number between one to six targets, and the initial target positions are known to the robot. For the SAT task, we combine the conditions of the previous two tasks.

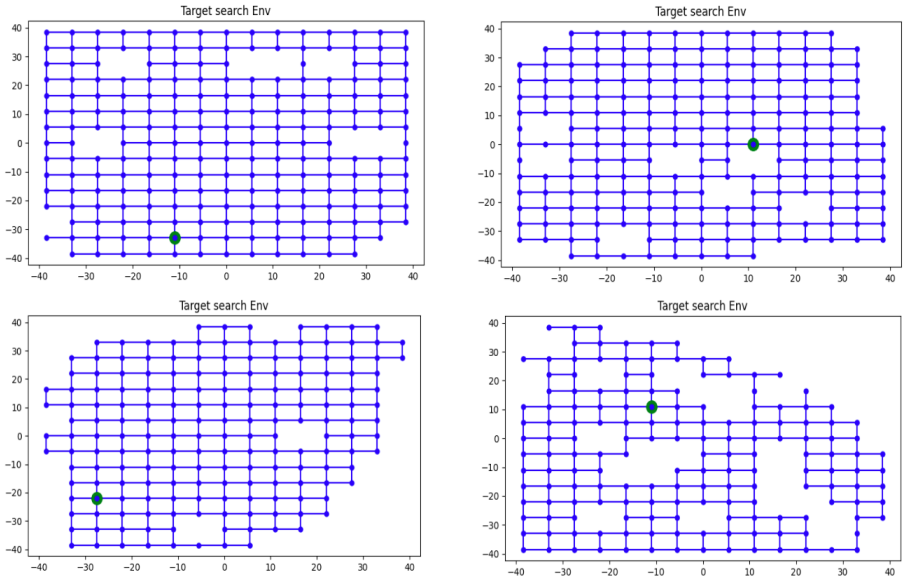


Figure 4.4: Examples of the random environments used in training

For the search task, the robot has a maximum of 800 steps to explore the environment. The episode is finished after reaching the timeout or the maximum entropy of a node is lower than a threshold ($H_{term} = 0.1$) in our setting). For the tracking task, the robot keeps tracking discovered targets for 250 steps in each episode. For the SAT task, the robot performs the task for 800 steps in each episode. Values for the reward weights are $w_{search} = w_{track} = 1$.

4.2.3. Training algorhtim

The policy is trained with DRL using Proximal Policy Optimization (PPO) algorithm [37] implemented by the RLlib framework [24]. PPO is an Actor-Critic method [18], where an actor (i.e. the policy) $\pi_\theta(a_t|\mathcal{G}_t)$ parameterized by θ predicts the action distribution a_t given the observation $\mathcal{G}_t$ and the critic $V^\pi_\phi(\mathcal{G}_t)$ parameterized by ϕ gives an estimate of the expected return based on $\mathcal{G}_t$. The hyperparameters for training the PPO algorithm are shown in Table 4.1.

The policy and value functions are trained for $3e6$ steps using an Intel i7-10875H CPU@2.30GHz computer. Based on this hardware setup, it takes less than $7ms$ to compute the policy action per time step.

Table 4.1: Hyperparameters for PPO training

Parameter	Value
Lambda λ	0.95
Gamma γ	0.99
Learning rate	2e-4
Value function loss coeff.	0.5
PPO Clip param.	0.2
KL coeff.	0.2
KL target	0.01
Entropy loss coeff.	0.01
Time steps per update	8000
SGD minibatch size	128
Numer of epoch per update	30

5

Experiments and results

In this chapter, we first introduce the baselines to compare with in section 5.1, the test conditions, and the criteria to evaluate the performance in section 4.2.1. Then the results from different experiments are analyzed including the pure dynamic target search task in section 5.3, the pure tracking task in section 5.4, and the search and tracking task in section 5.5. Finally, the effect of different graph structures and the trade-off of search and tracking are discussed in section 5.6

5.1. Baselines

In this section, we introduce several baselines we use for comparison: two next-best-viewpoint baselines, a GNN baseline inspired by the hierarchical graph structure from [32], and an ablation of our method.

5.1.1. Next-best-view baseline

For the next-best-view (NBV) baseline, we build an egocentric subgraph of the robot $\mathcal{G}_{s,t}$ without considering the abstract node. This is constructed by the m-hop receptive field of the robot node. In our implementation, we have $m = 3$ shown in Figure 5.1. Then we apply the same entropy diffusion dynamics of the environment to $\mathcal{G}_{s,t}$ and do m_p-step prediction. In our experiments, we have two baseline settings with $m_p = 1$ and $m_p = 3$ (the same receptive field as our method) respectively. After that, the candidate with the highest total information gain of search and tracking (i.e. r_t) is selected as the action. If there are multiple candidates having the same gain, the action is randomly sampled from them. These methods are qualified since they are exposed to the true entropy diffusion dynamics without noise.

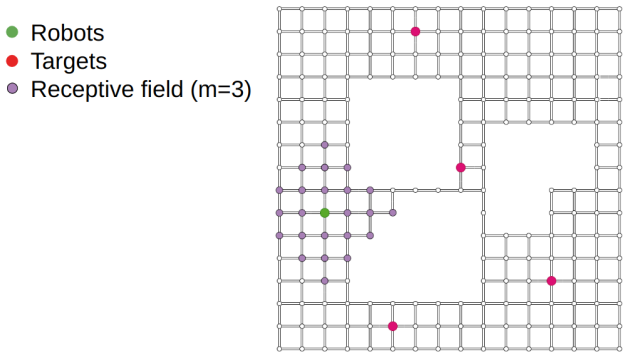


Figure 5.1: Egocentric subgraph of the robot

5.1.2. GNN baseline

The first GNN baseline we have is adapted from [32]. In that work, a hierarchical graph representation of the environment is built through 3D scene graphs. Recalling our hierarchy graph representation in Figure 4.2, there are multiple types of edges: connecting place nodes and connecting one place node with the abstract node. Compared with our method, all edges in [32] are modeled homogeneously bi-directional, and a three-layer GCN is directly applied as the GNN model in Figure 4.3.

Another GNN baseline is an ablation of our method. The local GATv2 operator of layer 1 and layer 2 shown in Figure 4.3 are removed and the global GAT of layer 3 is reserved. This means the query vector $\mathbf{v}_a$ used to compute the attention coefficient in equation (4.6) is always the same and unconditioned on the local information, which results in a static attention [5]. The GCN and the static attention baselines are trained with the same hyperparameter setting and scenarios as our method.

5.2. Test conditions and evaluation metrics

At test time, the pipeline proposed in chapter 4 is used. The targets follow random walk dynamics as described in chapter 3. The diffusion coefficient is set as $C = 0.7$ and the velocity gap between the robot and the target is set as $n_{vel} = 8$. Each learning method is trained with 3 different seeds. The results of all seeds are averaged and their standard deviations are computed. For every test, we evaluate and average the results over 50 episodes with random initial conditions. The same initial conditions of each episode are maintained over all methods.

Multiple performance indicators are used to evaluate each method and are listed below. Not all metrics are used in each scenario, relevant ones will be discussed for each result.

- Entropy decrease rate over time
- Percentage of targets discovered over time
- Number of nodes to achieve 95% confidence
- Normalized sum of target belief entropy over time [15]

The entropy decrease rate is computed by the ratio between the entropy reduction and the initial entropy of the entire environment. For environments with different numbers of nodes, this measures the ability of target search oriented information gathering uniformly. To evaluate the target tracking performance at the end of each episode, we first compute the minimum number of candidate nodes to achieve 95% confidence of each target belief. Then we count the percentage of targets with candidates within thresholds. Here we refer to three values $\{4, 8, 15\}$, which correspond to *tracked*, *not lost*, and *almost lost* tracking performance respectively. If the minimum number of candidates is larger than 15, we then evaluate it as a lost target. The percentage of discovered targets is used to evaluate the efficiency of target search during the search and tracking task. The normalized sum of target belief entropy is inspired from the normalized sum of the log of determinant of covariance [15] and defined as:

$$\bar{J} = \frac{J_{max} - \sum_{t=1}^{F} \sum_{j=1}^{M} H(B(\mathbf{y}_j(t)))}{J_{max} - J_{min}} \quad \in [0, 1] \tag{5.1}$$

Where the upper bound J_{max} is obtained when there is no observation of any targets in an episode, and the lower bound J_{min} is met when every target is observed at each step (i.e. $J_{min} = 0$). J_{max} is computed by:

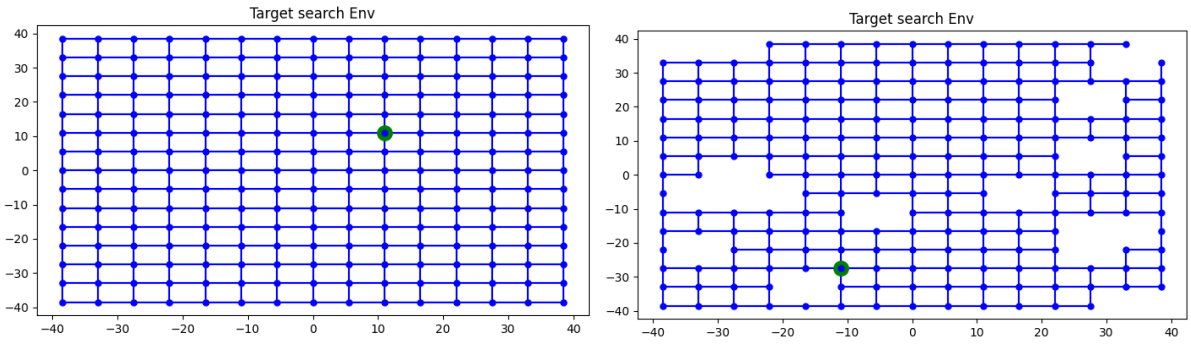
$$J_{max} = \sum_{j=1}^{M} \sum_{n_p=1}^{N_p} n_{vel} H(B(\mathbf{y}_j(0))(I + \hat{\mathbf{L}})^{n_p}) \tag{5.2}$$

Where N_p is the number of prediction steps in an episode.

5.3. Pure search task

5.3.1. Capability to generalize to different graph structures

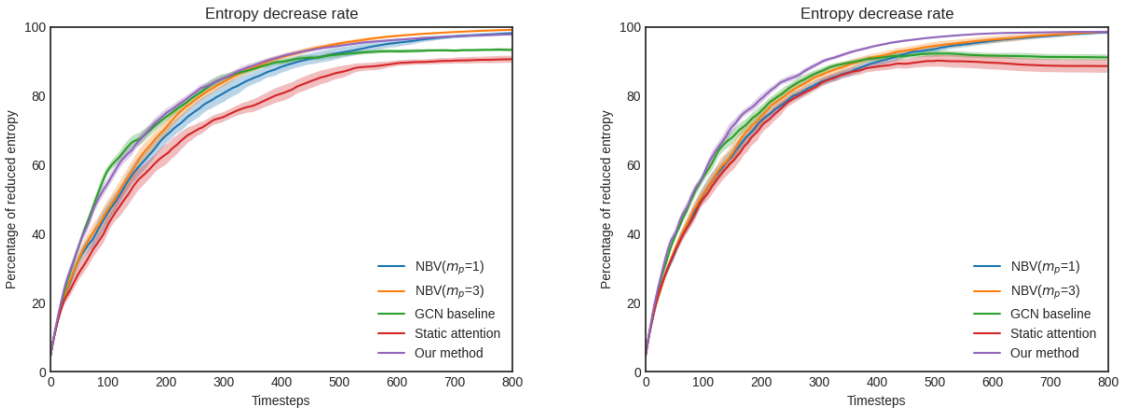
In realistic scenarios, the robot can be deployed in environments with different maps, and it is impossible to train the policy with all of them. Therefore, we first have an analysis of the capability of the learned policy to generalize to different graph structures. To do this, we do tests in environments with/without random cluttered obstacles as shown in figure 5.2. Same as the training stage, the robot has 800 steps to search in the environment with potential dynamic targets.



(a) Empty environment **(b)** Cluttered environment

Figure 5.2: Examples of test environments

Figure 5.3 compares the entropy decrease rate of the search process in the two kinds of environments. For each method, the mean and standard deviation of the results are plotted. In the empty environment, the GCN baseline and our method decrease higher entropy than other methods till around 350 steps, then the GCN baseline curve converges and gets passed by the two NBV baselines. The NBV baseline with $m_p = 3$ catches up with our method at around 500 steps. Since the entropy decrease rate at 500 steps is already higher than 95%, it shows that in most time of the search task in an empty environment, our method outperforms the others. In the cluttered environment, in general, our method achieves a higher entropy decrease rate than the others most of the time. Hence, it shows that the policy is able to generalize to environments with different graph structures. In both tests, the NBV baseline with a larger prediction horizon results in a better performance as expected.



(a) Empty environment **(b)** Cluttered environment

Figure 5.3: Entropy decrease rate of search process over time in the same size of environments as training stage

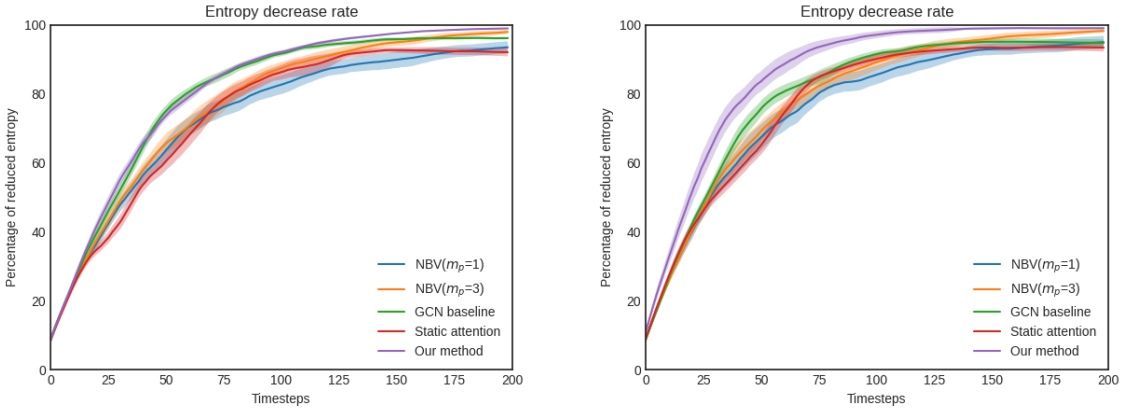
5.3.2. Scalability analysis

Apart from different obstacles, the robot also needs to handle different scales of maps. To verify whether the learned behavior applies to different sizes of environments, we do tests in two kinds of environ-

ments:

- Small scale environments constructed by an 11×11 lattice with half amount of nodes than the training environment
- Large scale environments constructed by a 21×21 lattice with double amount of nodes than the training environment

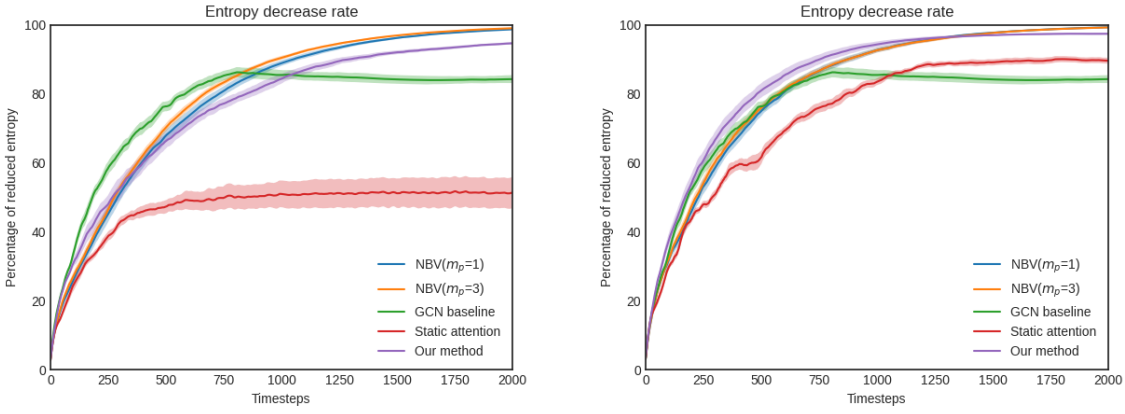
For each of them, we evaluate the performance in both empty and cluttered environments like in the previous section. In the small-scale environments, the robot has 200 steps for each episode whilst for the large-scale ones it has 2000 steps each.



(a) Empty environment **(b)** Cluttered environment

Figure 5.4: Entropy decrease rate of search process over time in small-scale environments

Figure 5.4 compares the entropy decrease rate over time in small-scale environments. In the empty environment, the GCN baseline and our method outperform the other methods. In the cluttered environment, our method decreases the entropy in the environment quicker than any other baseline. The performance gap between the two NBV baselines is more explicit when the entropy decrease rate is higher than 80%. This means when there are few nodes with high entropy in the environment, a longer horizon helps the robot to find a better solution as expected.



(a) Empty environment **(b)** Cluttered environment

Figure 5.5: Entropy decrease rate of search process over time in large-scale environments

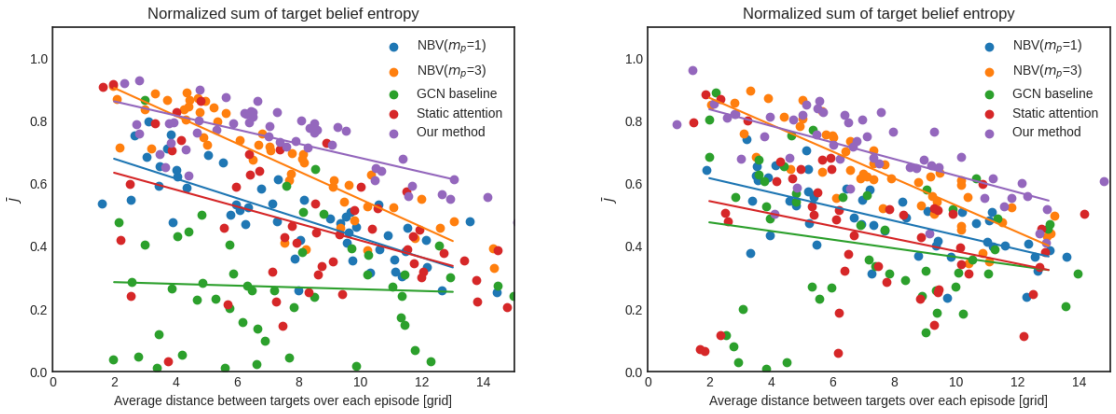
Figure 5.5 compares the entropy decrease rate over time in large-scale environments. In the empty environment, the GCN baseline starts with a better performance till 800 steps and then fails to decrease more entropy in the environment afterward. Our method only performs better than the two NBV

baselines at the first 350 steps, then it gets surpassed and never catches up with them throughout the process. In the cluttered environment, the GCN baseline faces a similar bottleneck and our method performs better than the two NBV baselines till around 1300 steps when the entropy decrease rate is already over 95%. This suggests that the policy is able to generalize to different scales of cluttered environments but is not scalable to larger empty environments.

5.4. Pure tracking task

5.4.1. Two target domains

In chapter 3, the robot is formulated to have an obvious velocity advantage over the targets. Therefore it is possible for the robot to continuously gather information about multiple targets by traveling among them. Nevertheless, if targets are too distant from each other, keeping tracking nearby targets can bring more information gain than trying to observe all targets. Hence, we first evaluate the performance in two-target environments. In both empty and cluttered environments, we evaluate each method over 50 episodes and compute the average relative distance between the two targets throughout each episode.



(a) Empty environment

(b) Cluttered environment

Figure 5.6: Relationship between $\bar{J}$ and target relative distance

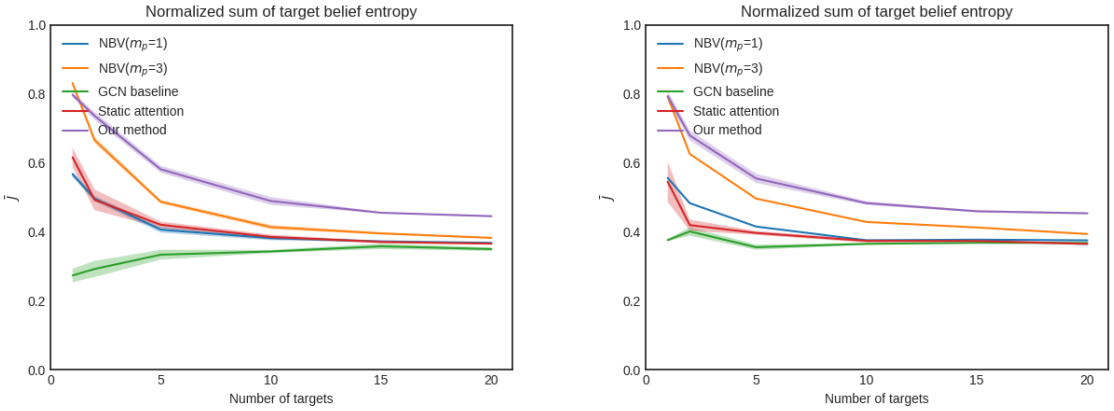
Figure 5.6 shows the relationship between the relative target distance and the normalized sum of target entropy $\bar{J}$. As is expected, the tracking performance decreases as the average distance between the two targets increases. When the two targets are relatively close to each other (i.e. the average distance between them is shorter than 5 grids), the performance of the NBV baseline with $m_p = 3$ is slightly better than our method. However, when the targets scatter more sparsely in the environment, the performance of the NBV baseline drops quicker than our method. This suggests that our method can reason about distant information to improve the tracking performance.

5.4.2. Scalability analysis

As is shown in the previous section, the relative distance between targets has an explicit effect on the tracking performance. On the other hand, the number of targets is also a key factor. We evaluate each method in both kinds of environments with up to 20 targets to test their scalability to a larger number of targets than the training process.

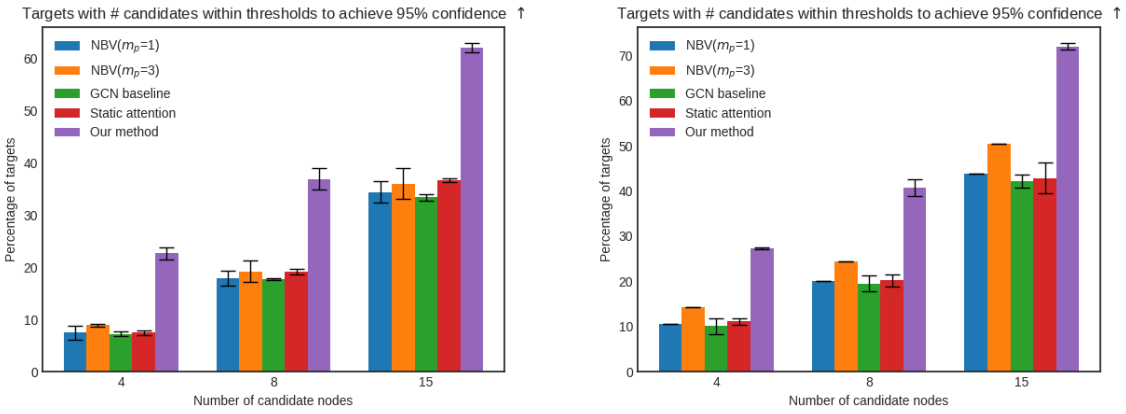
In Figure 5.7, we report each method's $\bar{J}$ performance in both kinds of environments with different numbers of targets. As is expected, there is a drop in the tracking performance when the quantity of targets increases. It is worth noting that the drop is getting slower with a larger target quantity. One reason is that as there are more targets in the environment, the robot can get more information due to a denser target distribution and does not have to waste time traversing among distant targets. When there are multiple targets in the environment, our method consistently outperforms all other baselines.

Figure 5.8 shows the percentage of 20 targets with candidates within thresholds in the environment. In both kinds of environments, our method outperforms the other methods by maintaining more tracked targets within each threshold. In the empty environment, the baseline methods perform similarly to each other while in the cluttered environment, the NBV baseline with $m_p = 3$ is the second best method.



(a) Empty environment **(b)** Cluttered environment

Figure 5.7: Experiments with different numbers of targets in the environment



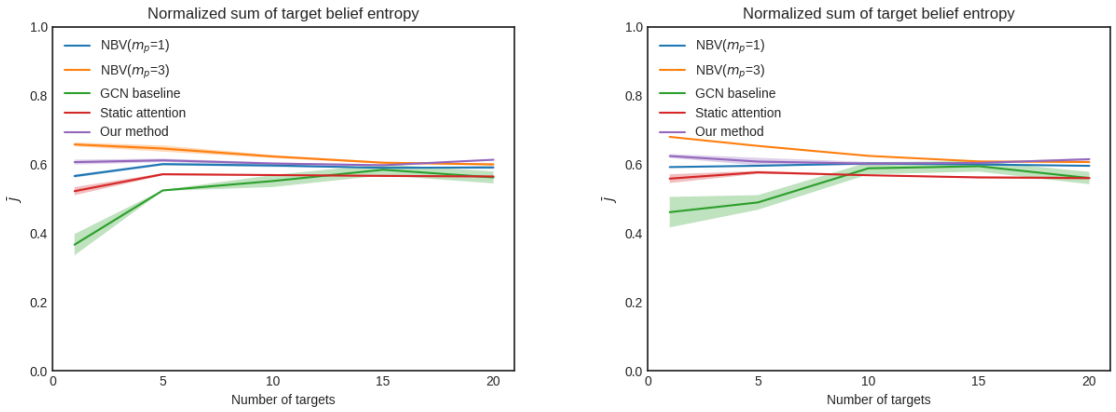
(a) Empty environment **(b)** Cluttered environment

Figure 5.8: Percentage of 20 targets with candidates within thresholds

5.5. Search and tracking task

5.5.1. Small-scale environments

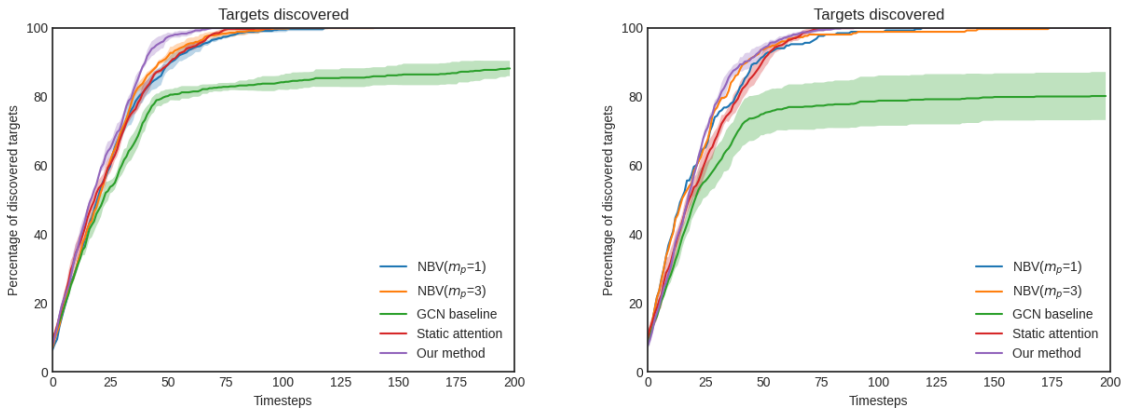
We first evaluate the SAT performance of each method in the small-scale environments used in section 5.3.2 with up to 20 targets.



(a) Empty environment **(b)** Cluttered environment

Figure 5.9: Experiments with different numbers of targets in the small-scale environment

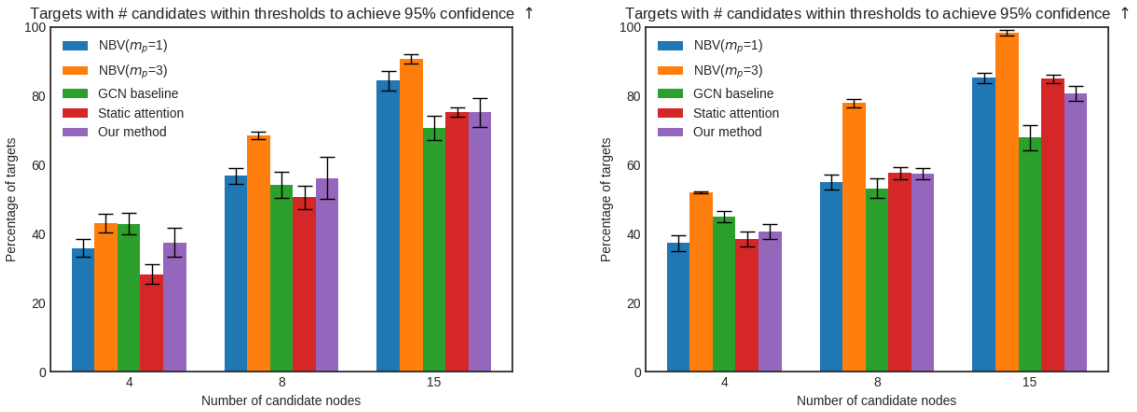
Figure 5.9 shows the $\bar{J}$ performance of each method in both kinds of environments with different numbers of targets. The NBV baseline with $m_p = 3$ achieves a better tracking performance than our method when the number of targets does not exceed 15. It is worth noting that the NBV baseline with $m_p = 1$, the static attention baseline, and the GCN baseline all have a performance that first grows and then slowly declines. The difference in this metric can be attributed to two possible reasons. One is that the earlier detection of more targets increases the difficulty of target tracking. As is reported in section 5.4.2, the increasing number of targets explicitly results in a drop of $\bar{J}$ when the number of targets does not exceed 10. The other reason is that the method prioritizes the search part over the tracking part. To further verify the speculations above, we compare the behavior of all methods in environments with 5 and 20 targets respectively as follows.



(a) Empty environment **(b)** Cluttered environment

Figure 5.10: Percentage of discovered targets over time with 5 targets in small-scale environments

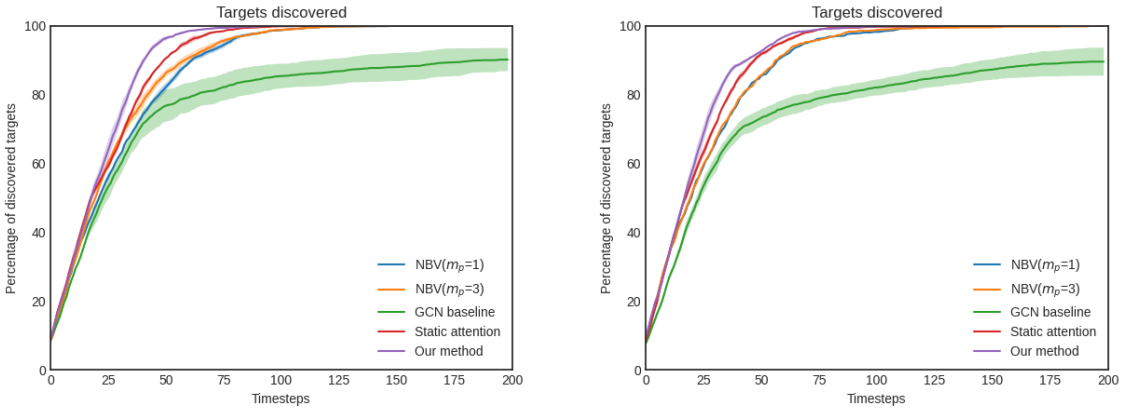
Figure 5.10 compares the percentage of discovered targets over time with 5 targets in the environment, which quantify the search performance. In the empty environment, our method outperforms all baselines in general. In the cluttered environment, the NBV baseline with $m_p = 3$ discovers the targets as fast as our method at first but it takes a longer time to discover all targets. Hence, it is obtained that our method performs better than the others for the search part.



(a) Empty environment **(b)** Cluttered environment

Figure 5.11: Percentage of 5 targets with candidates within thresholds in small-scale environments

Figure 5.11 compares the target tracking performance at the end of each episode. It shows that our method does not perform well in both kinds of environments. This confirms our previous speculations that our method does discover targets quicker than other baselines but the main reason for the second best $\bar{J}$ in Figure 5.9 is that our method weights more on searching for undiscovered targets than tracking discovered ones when there are 5 targets in the environment.

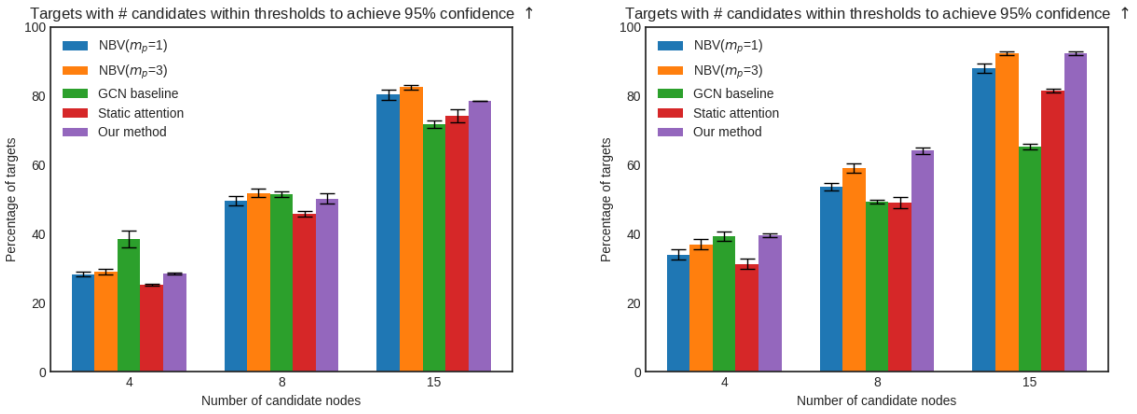


(a) Empty environment **(b)** Cluttered environment

Figure 5.12: Percentage of discovered targets over time with 20 targets in small-scale environments

Figure 5.12 compares the search performance through the percentage of discovered targets out of 20. In general, similar to the scenario with 5 targets, our method outperforms the others in both environments in terms of target search.

Figure 5.13 reports the target tracking performance of each method at the end of the episode. In the empty environment, our method performs similarly to NBV methods. The GCN baseline achieves the most *tracked* targets but according to Figure 5.12, it sacrifices to discover all targets and focuses on several discovered ones. In the cluttered environment, the NBV baseline with $m_p = 3$ and our method are the top two methods for target tracking without explicit gaps. Based on the results above, it is obtained that when there are a large number of targets in small-scale environments, our approach achieves a better search performance while maintaining the tracking performance, especially in the cluttered environment. This indicates that our method is not trapped by discovered targets to give up searching for more undiscovered ones.

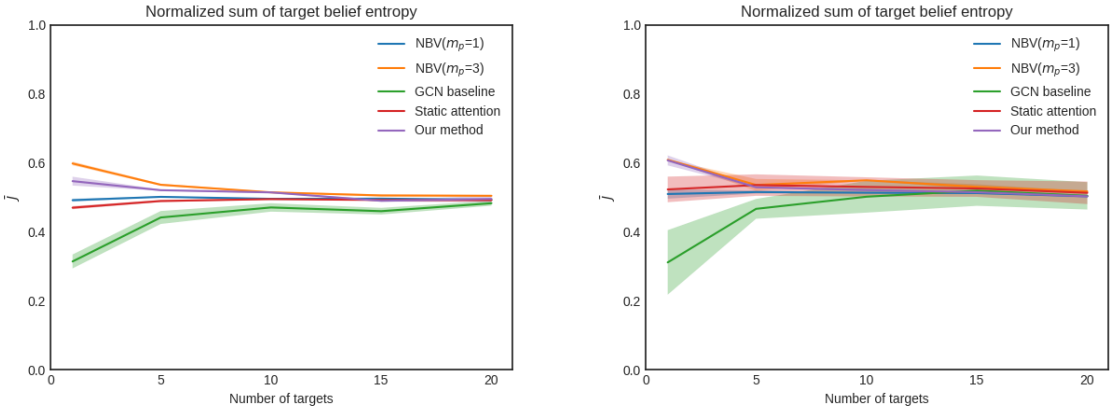


(a) Empty environment **(b)** Cluttered environment

Figure 5.13: Percentage of 20 targets with candidates within thresholds in small-scale environments

5.5.2. Medium-scale environments

We then present the results in larger environments used in section 5.3.1 with up to 20 targets. Figure 5.14 compares the $\bar{J}$ performance of each method in the medium-scale environment with different numbers of targets. When there are more than 5 targets in the environment, the performance of all the other methods is close except for the GCN baseline.



(a) Empty environment **(b)** Cluttered environment

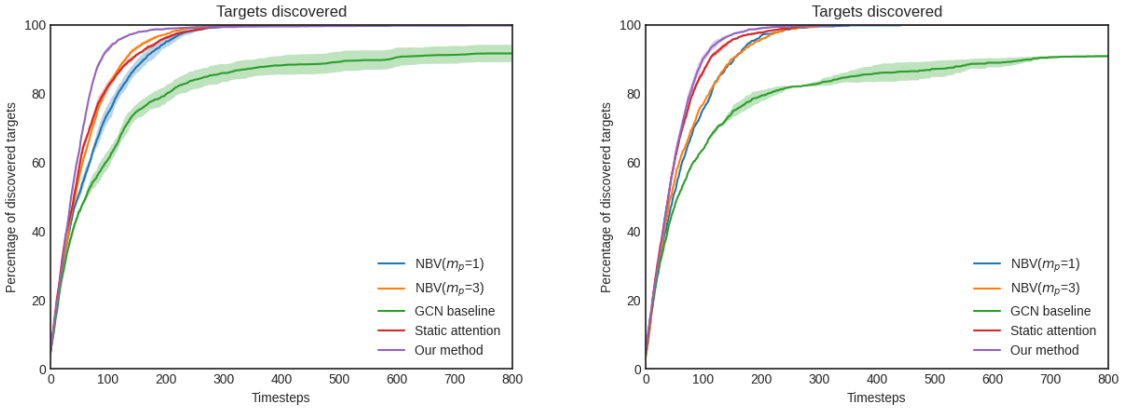
Figure 5.14: Experiments with different numbers of targets in the medium-scale environments

To analyze the search and tracking performance further, we compare the search performance in the environments with 20 targets in Figure 5.15 whilst the tracking performance at the end of each episode in Figure 5.16. According to the results, our method outperforms other baselines in searching for undiscovered targets while sacrificing the target tracking performance. This indicates that in medium-scale environments, our approach prioritizes target search over target tracking.

5.5.3. Large-scale environments

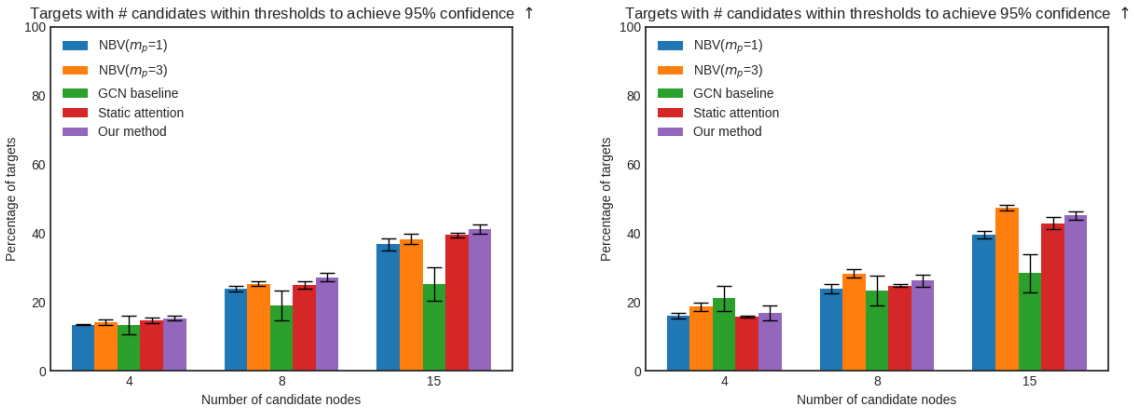
We further increase the number of nodes in the graph and do tests in large-scale environments used in section 5.3.2 with 20 targets.

As is shown in Figure 5.17, our method keeps the advantage of searching for undiscovered targets quicker than other baselines. Compared with its performance in small and medium-scale environments, the static attention baseline no longer outperforms the NBV methods. According to the tracking performance at the end of each episode in Figure 5.18, all the other methods perform similarly except the GCN baseline. This is because the GCN approach fails to discover all the targets as in Figure 5.17,



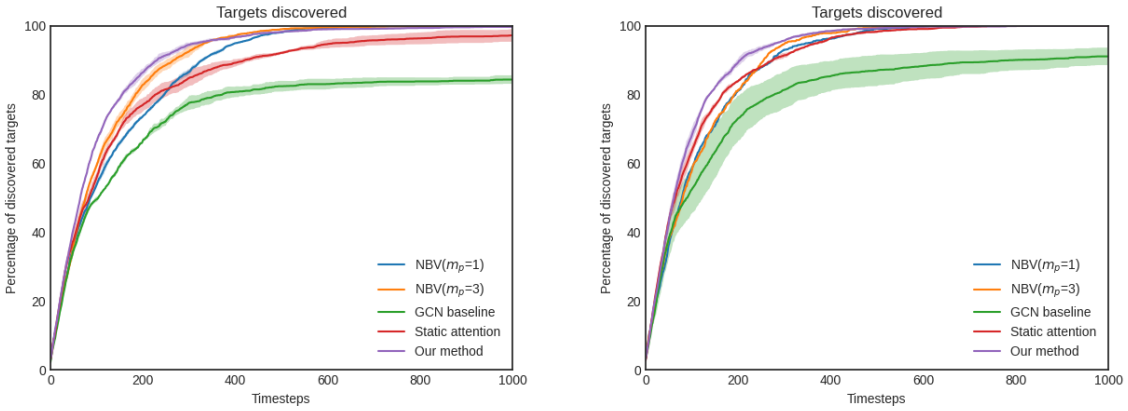
(a) Empty environment **(b)** Cluttered environment

Figure 5.15: Percentage of discovered targets over time with 20 targets in medium-scale environments



(a) Empty environment **(b)** Cluttered environment

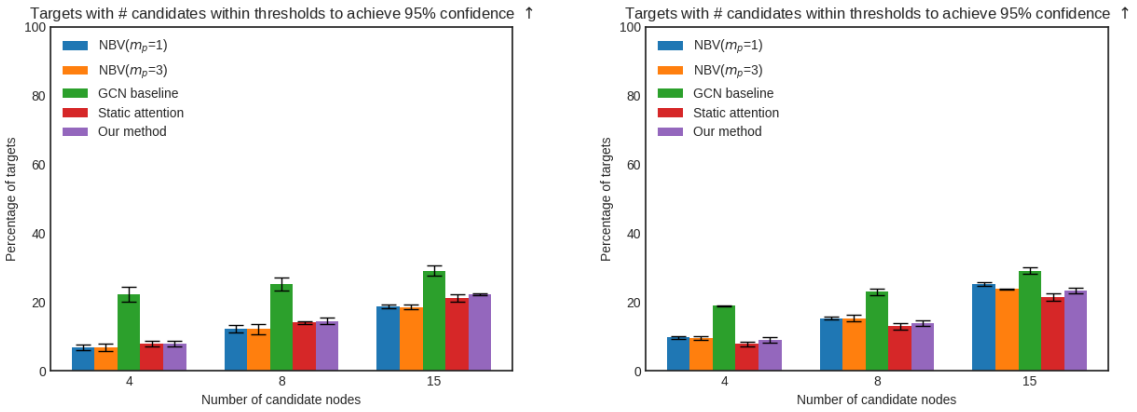
Figure 5.16: Percentage of 20 targets with candidates within thresholds in medium-scale environments



(a) Empty environment **(b)** Cluttered environment

Figure 5.17: Percentage of discovered targets over time with 20 targets in large-scale environments

thus the difficulty of target tracking decreases. According to Table 5.1, there is no explicit performance gap among different methods especially in cluttered environments. Consequentially, under equal performance in target tracking, our method achieves a better trade-off in terms of target search.



(a) Empty environment

(b) Cluttered environment

Figure 5.18: Percentage of 20 targets with candidates within thresholds in large-scale environments

Table 5.1: Target tracking performance throughout the episode in large-scale environments with 20 targets

	Normalized sum of target belief entropy $\bar{J}$ (mean/std)↑	
Env. Type	Empty	Cluttered
NBV ($m_p = 1$)	0.435/0.004	0.445/0.001
NBV ($m_p = 3$)	0.453/0.002	0.449/0.001
GCN baseline	0.447/0.007	0.431/0.032
Static attention	0.440/0.029	0.430/0.003
Our method	0.416/0.007	0.443/0.003

5.6. Discussion

5.6.1. The effect of the heterogeneous graph structure

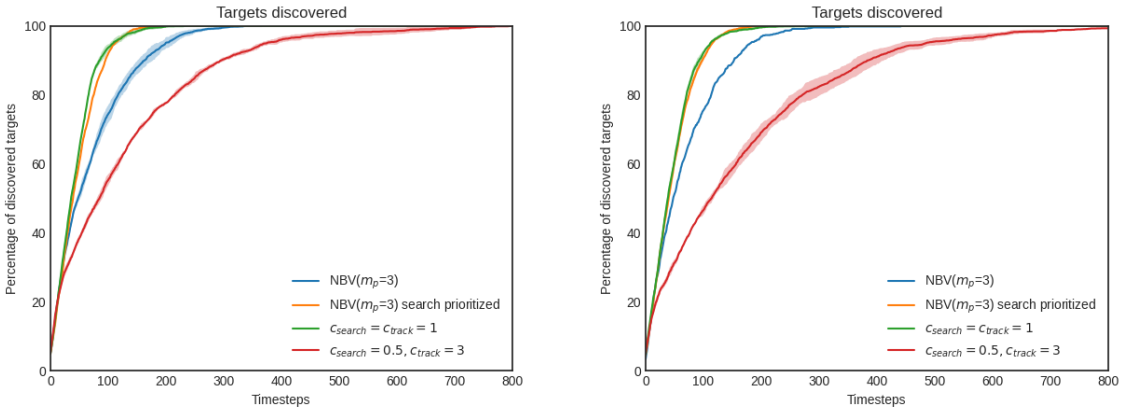
According to the results in the preceding sections, the GCN baseline gets unsatisfactory performances across most tasks. The primary divergence between the GCN baseline and our method is the manner in which the hierarchy graph representation is modeled. Specifically, the GCN baseline adopts a homogeneous style in modeling the edges that connect place nodes and the abstract node, treating them as the same type. Due to the permutation invariance of the graph, any two place nodes in the homogeneous hierarchy graph are 2-hop neighbors through the abstract node. Consequentially, this approach disrupts the spatial topological structure of the lattice graph representation shown in Figure 3.1. This means that information from nearby place nodes and distant places nodes are aggregated in the same way. When evaluating the behavior of the GCN baseline policy, we observe that the robot often gets trapped at the border of the environment, which reflects that it misses the structural information of the graph and can not do reasoning properly.

On the other hand, our method uses a heterogeneous graph structure to represent the two types of edges. In this way, the spatial topology of the original graph is reserved and the edges connected to the abstract node are used only to aggregate global information according to the local information.

5.6.2. The adjustable trade-off of search and tracking

According to experiment results in section 5.5, as the size of the environments increases, our method prioritizes target search over target tracking. Recalling the reward function in equation (4.7), c_{search} and c_{track} scale the search part and the tracking part of the total reward. This means their priority can be tuned by different weights. To verify this, we compare the policy trained by different weights with the

best baseline, the NBV baseline with $m_p = 3$, and its tuned variant to prioritize target search. We test all methods in medium-scale environments with 20 targets.

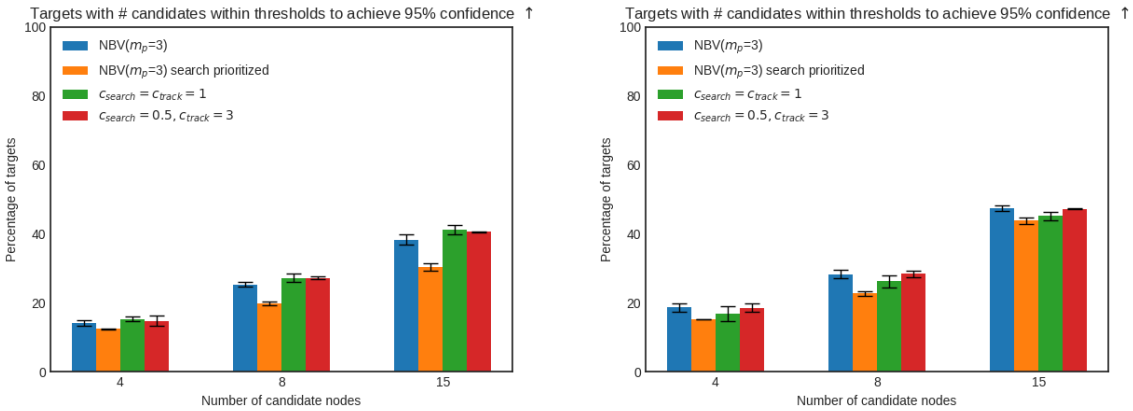


(a) Empty environment

(b) Cluttered environment

Figure 5.19: Percentage of discovered targets over time with 20 targets in medium-scale environments

Figure 5.19 compares the search performance of each method. The NBV variant performs as well as our method while the policy emphasizes tracking is attracted by discovered targets and sacrifices the target search efficiency.



(a) Empty environment

(b) Cluttered environment

Figure 5.20: Percentage of 20 targets with candidates within thresholds in medium-scale environments

Figure 5.20 presents the tracking performance at the end of each episode. It shows that there are small differences between our method and its variant. This is understandable that after most of the environment has been explored, the robot is doing pure tracking. In this scenario, both policies are supposed to perform the same. It is also noteworthy that the amount of lost targets by the NBV variant is larger than our method, especially in the empty environment. When comparing the target tracking performance throughout the episode in Table 5.2, our method outperforms the NBV variant in terms of continual tracking, and the variant of our method has an obvious advantage over others. This proves that our method achieves a better trade-off in search and tracking under equally good search performance, and the trade-off between search and tracking can be adapted through different weights on them.

Table 5.2: Target tracking performance throughout the episode in medium-scale environments with 20 targets

	Normalized sum of target belief entropy J (mean/std)↑	
Env. Type	Empty	Cluttered
NBV ($m_p = 3$)	0.504/0.002	0.517/0.002
NBV ($m_p = 3$) search prioritized	0.425/0.001	0.445/0.001
$c_{search} = c_{track} = 1$	0.495/0.009	0.503/0.006
$c_{search} = 0.5, c_{track} = 3$	0.546/0.005	0.585/0.007

6

Conclusions and future work

6.1. Conclusions

In this work, we introduce a framework for the search and tracking of an unknown number of dynamic targets with a graph representation. A novel graph formulation of the search and tracking problem is proposed and our framework learns a policy that outputs discrete viewpoint actions through Deep Reinforcement Learning using a GNN architecture. The effect of undiscovered dynamic targets is modeled by an entropy diffusion mechanism, and the tracking of discovered targets is converted to decrease the entropy of target belief distribution. Then the information is encoded into the observation graph and processed by the policy.

Under this framework, we demonstrate that the robot learns the pure search, pure tracking, as well as the search and tracking task. Our work could release the assumptions used by previous methods that the targets are static during the search process and the initial target positions are given as prior. The results of the experiments have shown that our policy outperforms multiple baselines, is able to generalize to different types of environments (with or without cluttered obstacles), and scales to environments with more than double the amount of targets during the training. Due to the design of the reward function, the priority of the search part and the tracking part can be adjusted through different weights for scenarios with different emphases.

6.2. Limitations and future work

There are also some limitations of this work. According to the experiment results of the pure search task, our policy does not scale well to all kinds of large-size environments. Besides, in this work, we generate the graph representation of the environment by a square lattice, which can be extended to other probabilistic roadmap methods (PRM).

As is introduced in Chapter 3, we assume the targets follow the discrete random walk dynamics. The velocity gap between the target and the robot is modeled by different state update frequencies. However, this does not reflect the ground truth in realistic conditions. Besides, the random walk model is difficult to track since the target can move in any direction. In the future, other types of dynamics can be applied.

In our experiments, we proved that the trade-off of search and tracking can be adjusted by different weights of the two parts. So far what we do is to set the priority of the robot beforehand according to the reward function and make the robot decide the switch between target search and target tracking. In the future, the priority of the robot can be conditioned on historical observations and self-adaptive.

Last but not least, according to the experiment results, it takes the robot much more time to search over a larger environment and there is a limit to the number of targets one robot can track. In the future, multiple robots could be deployed for better search and tracking performance.

References

[1] Iro Armeni et al. “3d scene graph: A structure for unified semantics, 3d space, and camera”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5664–5673.

[2] Nikolay Atanasov et al. “Information acquisition with sensing robots: Algorithms and error bounds”. In: *2014 IEEE International conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 6447–6454.

[3] Rik Bähnemann et al. “Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem”. In: *Field and Service Robotics: Results of the 12th International Conference*. Springer. 2021, pp. 277–290.

[4] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. “Revisiting active perception”. In: *Autonomous Robots* 42.2 (2018), pp. 177–196.

[5] Shaked Brody, Uri Alon, and Eran Yahav. “How attentive are graph attention networks?” In: *International Conference on Learning Representations*. 2022.

[6] Jennifer Casper and Robin R. Murphy. “Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 33.3 (2003), pp. 367–385.

[7] Mihir Dharmadhikari et al. “Motion primitives-based path planning for fast and agile exploration using aerial robots”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 179–185.

[8] Alberto Dionigi et al. “E-VAT: An Asymmetric End-to-End Approach to Visual Active Exploration and Tracking”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4259–4266.

[9] Arnaud Doucet et al. “Particle filtering for multi-target tracking and sensor management”. In: *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002.(IEEE Cat. No. 02EX5997)*. Vol. 1. IEEE. 2002, pp. 474–481.

[10] Alex Goldhoorn et al. “Searching and tracking people with cooperative mobile robots”. In: *Autonomous Robots* 42.4 (2018), pp. 739–759.

[11] Héctor H González-Banos and Jean-Claude Latombe. “Navigation strategies for exploring indoor environments”. In: *The International Journal of Robotics Research* 21.10-11 (2002), pp. 829–848.

[12] Jonathan P How et al. “Increasing autonomy of UAVs”. In: *IEEE Robotics & Automation Magazine* 16.2 (2009), pp. 43–51.

[13] Christopher D Hsu et al. “Scalable Reinforcement Learning Policies for Multi-Agent Control”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 4785–4791.

[14] Julius Ibenthal et al. “Target search and tracking using a fleet of UAVs in presence of decoys and obstacles”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 188–194.

[15] Heejin Jeong et al. “Deep Reinforcement Learning for Active Target Tracking”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 1825–1831.

[16] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[17] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[18] Vijay Konda and John Tsitsiklis. “Actor-critic algorithms”. In: *Advances in neural information processing systems* 12 (1999).

[19] Dexter Kozen and Marc Timme. “Indefinite summation and the Kronecker delta”. In: (2007).

[20] Christopher M Kreucher. *An information-based approach to sensor resource allocation*. University of Michigan, 2005.

[21] Miroslav Kulich, Juan José Miranda-Bront, and Libor Přeučil. “A meta-heuristic based goal-selection strategy for mobile robot search in an unknown environment”. In: *Computers & Operations Research* 84 (2017), pp. 178–187.

[22] Miroslav Kulich, Libor Přeućil, and Juan José Miranda Bront. “Single robot search for a stationary object in an unknown environment”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 5830–5835.

[23] Xianfeng Li et al. “Profit-driven adaptive moving targets search with UAV swarms”. In: *Sensors* 19.7 (2019), p. 1545.

[24] Eric Liang et al. “RLlib: Abstractions for distributed reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3053–3062.

[25] Chang Liu and J Karl Hedrick. “Model predictive control-based target search and tracking using autonomous mobile robot with limited sensing domain”. In: *2017 American Control Conference (ACC)*. IEEE. 2017, pp. 2937–2942.

[26] Max Lodel et al. “Where to look next: Learning viewpoint recommendations for informative trajectory planning”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 4466–4472.

[27] Yi Lu et al. “MGRL: Graph neural network based inference in a Markov network with reinforcement learning for visual navigation”. In: *Neurocomputing* 421 (2021), pp. 140–150.

[28] A v Luikov. *Analytical heat diffusion theory*. Elsevier, 2012.

[29] Yifei Ma, Roman Garnett, and Jeff Schneider. “Active search for sparse signals with region sensing”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[30] Farzad Niroui et al. “Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 610–617.

[31] Ryan R Pitre, X Rong Li, and R Delbalzo. “UAV route planning for joint search and track missions—An information-value approach”. In: *IEEE Transactions on Aerospace and Electronic Systems* 48.3 (2012), pp. 2551–2565.

[32] Zachary Ravichandran et al. “Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 9272–9279.

[33] Allison Ryan and J Karl Hedrick. “Particle filter based information-theoretic active sensing”. In: *Robotics and Autonomous Systems* 58.5 (2010), pp. 574–584.

[34] Alejandro Sarmiento, Rafael Murrieta-Cid, and Seth Hutchinson. “A multi-robot strategy for rapidly searching a polygonal environment”. In: *Ibero-American Conference on Artificial Intelligence*. Springer. 2004, pp. 484–493.

[35] Brent Schlotfeldt, Nikolay Atanasov, and George J Pappas. “Maximum information bounds for planning active sensing trajectories”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4913–4920.

[36] Brent Schlotfeldt et al. “Anytime planning for decentralized multirobot active information gathering”. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1025–1032.

[37] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[38] Amarjeet Singh et al. “Efficient informative sensing using multiple robots”. In: *Journal of Artificial Intelligence Research* 34 (2009), pp. 707–755.

[39] Andrew J Smith and Geoffrey A Hollinger. “Distributed inference-based multi-robot exploration”. In: *Autonomous Robots* 42.8 (2018), pp. 1651–1668.

[40] Yoonchang Sung and Pratap Tokekar. “Algorithm for searching and tracking an unknown and varying number of mobile targets using a limited fov sensor”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 6246–6252.

[41] Ekaterina Tolstaya et al. “Multi-robot coverage and exploration using spatial graph neural networks”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 8944–8950.

[42] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[43] Petar Veličković et al. “Graph attention networks”. In: *International Conference on Learning Representations*.

[44] Peng Yan, Tao Jia, and Chengchao Bai. “Searching and tracking an unknown number of targets: a learning-based method enhanced with maps merging”. In: *Sensors* 21.4 (2021), p. 1076.

[45] Chaoqi Yang et al. “Revisiting over-smoothing in deep GCNs”. In: *arXiv preprint arXiv:2003.13663* (2020).

[46] Leonardo Zacchini, Alessandro Ridolfi, and Benedetto Allotta. “Receding-horizon sampling-based sensor-driven coverage planning strategy for AUV seabed inspections”. In: *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*. IEEE. 2020, pp. 1–6.

[47] Vinicius Zambaldi et al. “Relational deep reinforcement learning”. In: *arXiv preprint arXiv:1806.01830* (2018).

[48] Delong Zhu et al. “Deep reinforcement learning supervised autonomous exploration in office environments”. In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 7548–7555.