Hierarchical and Partially Observable Goal-driven Policy Learning with Goals Relational Graph

Xin Ye and Yezhou Yang
Active Perception Group, School of Computing, Informatics, and Decision Systems Engineering,
Arizona State University, Tempe, USA
{xinye1, yz.yang}@asu.edu

Abstract

We present a novel two-layer hierarchical reinforcement learning approach equipped with a Goals Relational Graph (GRG) for tackling the partially observable goal-driven task, such as goal-driven visual navigation. Our GRG captures the underlying relations of all goals in the goal space through a Dirichlet-categorical process that facilitates: 1) the high-level network raising a sub-goal towards achieving a designated final goal; 2) the low-level network towards an optimal policy; and 3) the overall system generalizing unseen environments and goals. We evaluate our approach with two settings of partially observable goal-driven tasks — a grid-world domain and a robotic object search task. Our experimental results show that our approach exhibits superior generalization performance on both unseen environments and new goals [1].

1. Introduction

Goal-driven visual navigation defines a task where an intelligent agent (with an on-board camera) is expected to take reasonable steps to navigate to a user-specified goal in an unknown environment (see Figure 1 left). It is a fundamental yet essential capability for an agent and could serve as an enabling step for other tasks, such as Embodied Question Answering [8] and Vision-and-Language Navigation [3]. Goal-driven visual navigation could be formulated as a partially observable goal-driven task. In this paper, we present a novel Hierarchical Reinforcement Learning approach with a Goals Relational Graph formulation (HRL-GRG) tackling it. Formally, a partially observable goal-driven task yields a 10-tuple $< S, A, T, G, R, \Omega, O, G_d, \Phi, \gamma >$, in which S is a set of states, A is a set of actions, $T : S \times A \times S \rightarrow [0, 1]$ is a state transition probability function, $G \subseteq S$ is a set of goal states, $R : S \times A \times S \times G \rightarrow \mathbb{R}$ is a reward function, Ω is a set of observations that are determined by conditional observation probability $O : S \times \Omega \rightarrow [0, 1]$. Similarly, G_d is a set of goal descriptions that describe observations with goal recognition probability $\Phi : \Omega \times G_d \rightarrow [0, 1]$. In particular, g is the goal state of the corresponding goal description g_d iff $\Phi(\operatorname{argmax}_\omega O(g, \omega), g_d)$ is larger than a pre-defined threshold. $\gamma \in (0, 1]$ is a discount factor. The objective of a partially observable goal-driven task is to maximize the expected discounted cumulative rewards $\mathbb{E}[\sum_t^\infty \gamma^t r_{t+1}(s_t, a_t, s_{t+1}, g)|s_t, g]$ by learning an optimal action policy to select an action a_t at the state s_t given the observation o_t and the goal description g_d.

Classic RL methodology optimizes an agent's decision-making action policy in a given environment [30]. To make RL towards real-world applicable, equipped with deep neural networks, Deep Reinforcement Learning (DRL) [22] algorithms are able to directly take the high dimensional sensory inputs as states S and learn the optimal action policy that generalizes across various states. However, the applicability of most advanced RL algorithms is still limited to domains with fully observed state space S and/or fixed goal states G, which is not the case in reality [22, 21, 20, 29, 11].

For real-world applications like visual navigation, an agent's sensory inputs capture the local information of its surrounding environments (a partially observable state space). Additionally, the real-world applications could be subject to goal changes, requiring a system to be goal-adaptive. Therefore, a real-world application can be formulated as a partially observable goal-driven task, that is different from a fully observable goal-driven task [22, 21, 11, 25] or a partially observable task [13, 18, 12]. It requires the agent to be capable of inferring its state in the augmented state space $S \times G$. Namely, the agent should take actions based on its current relative states with respect to the goal states, which can only be estimated from its sensory observations Ω and the goal descriptions G_d. This is challenging due to 1) the large augmented state space, 2) the different modalities that the observations Ω and the goal descriptions G_d could have. For example, while RGB images are usually taken as the

[1]Codes and models are available at https://github.com/Xin-Ye-1/HRL-GRG.

observations, semantic labels are more efficient in describing task goals [5].

To address the challenges, our HRL-GRG incorporates a novel Goals Relational Graph (GRG), which is designed to learn goal relations from the training data through a Dirichlet-categorical process [31] dynamically. In such a way, our model estimates the agent's states in terms of the learned relations between sub-goals that are visible in the agent's current observations and the designated final goal. Furthermore, our HRL-GRG decomposes the partially observable goal-driven task into two sub-tasks: 1) a high-level sub-goal selection task, and 2) a low-level fully observable goal-driven task. Specifically, the high-level layer selects a sub-goal $sg \in G_d$ that is observable in the current sensory input o, i.e. $\Phi(o, sg) > 0$, and could also contribute to achieving the designated final goal $g \in G_d$. The objective of the low-level layer is to achieve the observable sub-goal, yielding a well-studied fully observable task [22, 21, 11].

Many prior DRL methods tackling partially observable tasks [13, 18, 12] are not designed for goal-driven tasks. Therefore, their learned policies are not goal-adaptive. Adapting to new goals is critical for real-world tasks, such as goal-driven visual navigation [40], robotic object search [38, 36, 5] and room navigation [38]. Current goal-driven visual navigation methods generally neglect the essential role of estimating the agent's state under the partially observable goal-driven settings effectively, thus their performance still leaves much to be desired especially in terms of generalization ability (in-depth discussion in Section 2). Here, we argue and show our novel GRG modeling fills the gap.

Formally, we define GRG as a complete weighted directed graph $< V, E, W >$ in which $V = G_d$ is a set of nodes representing the goals G_d, and E is the directed edges connecting two nodes with the weights W. We incorporate GRG into HRL via two aspects: 1) weighing each candidate sub-goal in the high-level layer by $\mathbb{C}(\tau^*)$, the cost of the optimal plan τ^* from the sub-goal to the goal over the GRG; 2) terminating the low-level layer referring to the optimal plan τ^* from the proposed sub-goal to the goal over the GRG. To empirically validate the presented system, we start with demonstrating the effectiveness of our method in a grid-world domain where the environments are partially observable and a set of goals following a pre-defined relation are specified as the task goals. The design follows the intuition in real-world applications that certain relations hold in the goal space. For example, in the robotic object search task, users arrange the household objects in accordance with their functionalities. Another example is the indoor navigation task where room layouts are not random. Furthermore, in addition to the grid-world experiment, we also apply our method to tackle the robotic object search task in both the AI2-THOR [15] and the House3D [34] benchmark environments. We show HRL-GRG model exhibits superior performance in both experiments over other baseline approaches, with extensive ablation analysis.

2. Related Work

Research works on partially observable goal-driven tasks are explored typically under the visual navigation scenarios: an agent learns to navigate to user-specified goals with its first-person view visual inputs. Previous works' contributions lie in representation learning of the agent's underlying state and knowledge embedding for goal state inference.

In [40], the authors present a target-driven DRL model to learn a desired action policy conditioned on both visual inputs and target goals. With a target goal being specified as an image taken at the goal position, their model captures the spatial configuration between the agent's current position and the goal position as the agent's underlying state. However, when the goal position is far away, the inputs of the model lack the information to infer the spatial configurations, and so the model instead memorizes such spatial configurations. As a result, their model relies on a scene-specific layer for every single environment. Similar issues also exist in [33]. [28] represents the agent's current state with respect to the goal state through a semi-parametric topological memory while it requires a pre-exploration stage to build a landmark graph. The authors of [10] locate the goal position in their predicted top-down egocentric free space map. However, the method struggles when the goal is not visible. With more detailed information about the goals, the Vision-and-Language Navigation task has drawn research attention in which a fine-grained language-based visuomotor instruction serves as the goal description for the agent to follow and achieve [2, 9, 32]. Yet, specifying a goal with an image or a visuomotor instruction is inefficient and impractical for real-world applications. Instead, taking a concise semantic concept as a goal description is more desirable [23, 36, 26, 27, 5, 6, 39]. Semantic goals as model inputs, typically come in the form of one-hot encoded vectors or word embeddings. Therefore, goal inputs provide limited information for estimating the agent's states relative to the goal states.

Since it is non-trivial to incorporate complete information with a goal description as input, others embed task-specific prior knowledge to infer the goal states. In [36, 26, 27], the authors extract object relations from the Visual Genome [16] corpus and incorporate this prior into their models through Graph Convolutional Networks [14]. The extracted object relations encode the co-occurrence of objects based on human annotations from the Visual Genome dataset, which may not be consistent with the target application environments and the agent's understanding of the world. More recently, the authors of [35] come up with the Bayesian Relational Memory (BRM) architecture to capture the room layouts of the training environments from the agent's own experience for room navigation. The BRM further serves as a planner to

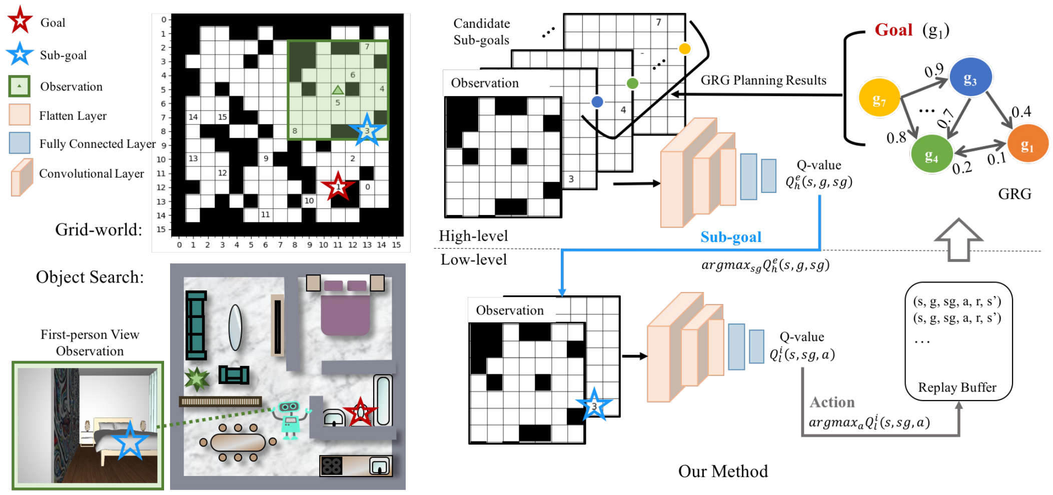


Figure 1: Illustrations of the grid-world domain and the robotic object search task (left), and an overview of our method (right).

propose a sub-goal to the locomotion policy network. Since the proposed sub-goal is still not observable, the authors of BRM train individual locomotion polices for each sub-goal to respectively tackle one partially observable task. In such manner, the BRM model's low-level network is not goal-adaptive and still brings about inefficiency and scaling concerns.

Apart from prior research efforts, we present a novel hierarchical reinforcement learning approach equipped with a GRG formulation for the general partially observable goal-driven task. Our GRG captures the underlying relations among all goals in the goal space and enables our hierarchical model to achieve superior generalization performance by decomposing the task into a high-level sub-goal selection task and low-level fully observable goal-driven task.

3. Hierarchical RL with GRG

3.1. Overview

Our focus is the partially observable goal-driven task where the agent needs to make a decision of which action to take to achieve a user-specified goal relying on its partial observations. Without loss of generality, we represent the observations Ω as images, such as the local egocentric top-down maps in the grid-world domain and the first-person view RGB images in the robotic object search task (see Figure 1 left). We specify the goal descriptions G_d as categorical labels (goal indices in the grid-world domain and object categories in the robotic object search task). To better illustrate our method, we take the grid-world domain as an example. The agent is asked to move to a goal position indicated by the goal index between obstacles. The agent can only observe a local map of obstacles and goals. The objective is to learn an optimal policy for several goals and instances of the grid-world domain which can generalize to new/unseen ones.

Figure 1 (right) depicts an overview of our method that is composed of a Goals Relational Graph (GRG), a high-level network and a low-level network. At time step t, the agent receives an observation $o_t \in \Omega$ that is a local map of its surrounding obstacles and a set of visible goals $VG = \{vg \mid vg \in G_d \ and \ \Phi(o_t, vg) > 0\}$. We take these visible goals as the candidate sub-goals at the time step t, and our high-level network learns a policy to select one from them to achieve the designated final goal $g \in G_d$. In order to be goal-adaptive, the system weighs each candidate sub-goal in VG by its relation to the designated final goal g, estimated from GRG. As a result, our high-level network proposes a sub-goal $sg_t \in VG$ conditioning on both the observation o_t and the designated final goal g. After the sub-goal sg_t is proposed, our low-level network decides an action a_t conditioning on both the observation o_t and the sub-goal sg_t for the agent to perform. Afterwards, the agent receives a new observation o_{t+1}, and our low-level network repeats N_t times to achieve the sub-goal sg_t until 1) the sub-goal sg_t is achieved; 2) the low-level network terminates itself if a better sub-goal appears in its current observation; 3) the low-level network runs out of a pre-defined maximum number of steps N^l_{max} . Either way, the low-level network collects an N_t-step long trajectory and terminates at the observation o_{t+N_t}. The trajectory updates the GRG. Then, the high-level network takes the control back to propose the next sub-goal. Overall, the process repeats until it either achieves the designated final goal g or reaches a predefined maximum number of actions N_{max}.

3.2. Goals Relational Graph (GRG)

GRG representation. We formulate GRG as a complete weighted directed graph $< V, E, W >$ on all goals in the goal space G_d (i.e. $V = G_d$). For any goal g_i and goal g_j, we define the weight w_{ij} on the directed edge (g_i, g_j) as a measure of how likely and quickly the goal g_j would appear according to Φ if our low-level network tries to achieve the goal g_i. We set the weight $w_{ii} = 1$ and adopt a Dirichlet-categorical model to learn w_{ij} for any $i \neq j$.

To be specific, we first assign a random variable X_{ij} to denote what would happen to the goal g_j if our low-level network achieves the goal g_i. Every time when the goal g_i is proposed by our high-level network, our low-level network generates a trajectory that has at most N^l_{max} steps to achieve the goal g_i. It introduces the following $N^l_{max} + 1$ events that X_{ij} may take:

- Event k ($1 \leq k \leq N^l_{max}$): the goal g_j appears when k steps are taken by our low-level network. We quantify the event k as $x_{ij,k} = \gamma^{k-1}$ where $\gamma \in (0, 1]$ is the discount factor to denote how close the goal g_j to the goal g_i.
- Event $N^l_{max} + 1$: the goal g_j doesn't appear. We quantify this event as $x_{ij,N^l_{max}+1} = 0$.

It is fair to assume that $X_{ij} \sim Cat(\boldsymbol{\theta_{ij}})$ in which the parameter $\boldsymbol{\theta_{ij}} = (\theta_{ij,1}, \theta_{ij,2}, ..., \theta_{ij,N^l_{max}+1}) \sim Dir(\boldsymbol{\alpha_{ij}})$ is a learnable Dirichlet prior. $\boldsymbol{\alpha_{ij}} = (\alpha_{ij,1}, \alpha_{ij,2}, ..., \alpha_{ij,N^l_{max}+1})$ is a concentration hyperparameter representing the pseudo-counts of all event occurrences. Thus, it can be empirically chosen. Lastly, the weight w_{ij} is set as $\mathbb{E}[X_{ij}]$.

GRG update. Each time when the low-level network is invoked to achieve the goal g_i, we get a trajectory as a sample $\mathcal{D}$ to update the GRG. For any goal g_j in our goal space, we count the number of the occurrences of all events and denote it as $\boldsymbol{c_{ij}} = (c_{ij,1}, c_{ij,2}, ..., c_{ij,N^l_{max}+1})$. Since the Dirichlet distribution is the conjugate prior distribution of the categorical distribution, the posterior distribution of the parameter $\boldsymbol{\theta_{ij}}$, namely $\boldsymbol{\theta_{ij}}|\mathcal{D} \sim Dir(\boldsymbol{\alpha_{ij}} + \boldsymbol{c_{ij}}) = Dir(\alpha_{ij,1} + c_{ij,1}, \alpha_{ij,2} + c_{ij,2}, ..., \alpha_{ij,N^l_{max}+1} + c_{ij,N^l_{max}+1})$. As a result, the posterior prediction distribution of a new observation $P(X_{ij} = x_{ij,k}|\mathcal{D})$ can be estimated by Equation 1, and the weight $w_{ij} = E(X_{ij}|\mathcal{D}) = \sum_k x_{ij,k} P(X_{ij} = x_{ij,k}|\mathcal{D})$.

$$P(X_{ij} = x_{ij,k}|\mathcal{D}) = \mathbb{E}[\theta_{ij,k}|\mathcal{D}] = \frac{\alpha_{ij,k} + c_{ij,k}}{\sum_k (\alpha_{ij,k} + c_{ij,k})}. \tag{1}$$

GRG planning. With the GRG being learned and updated, we quantify the relation of a goal g_i to a goal g_j by the cost $\mathbb{C}(\tau^*_{i,j})$ of the optimal plan $\tau^*_{i,j}$ searched from g_i to g_j over the GRG. In particular, suppose $\tau_{i,j} = \{\tau_1, \tau_2, ..., \tau_M\}$ is a plan searched from g_i to g_j over the GRG in which $g_{\tau_m\ (1 \leq m \leq M)}$ is a goal from our goal space G, $\tau_1 = i$ and $\tau_M = j$, we define the optimal plan $\tau^*_{i,j} = \mathrm{argmax}_{\tau_{i,j}} \prod_{m=1}^{M-1} w_{\tau_m \tau_{m+1}}$. We adopt the cost of the optimal plan $\tau^*_{i,j}$, $\mathbb{C}(\tau^*_{i,j}) = \max_{\tau_{i,j}} \prod_{m=1}^{M-1} w_{\tau_m \tau_{m+1}}$ as the measure of the relation from the goal g_i to the goal g_j.

3.3. Goal-driven High-level Network

Model formulation. The high-level network selects a sub-goal sg aiming to achieve the designated final goal g. We first introduce an extrinsic reward r^e. Here, we adopt a binary reward as the extrinsic reward to encourage the agent to achieve the final goal. Specifically, the agent receives a reward of 1 if it achieves the final goal g, i.e. $r^e_t(s_{t-1}, a_{t-1}, s_t, g) = 1$ iff the state s_t is the goal g's state, and 0 otherwise. Thus, the high-level task is formulated as maximizing the Q-value $Q^e_h(s, g, sg) = \mathbb{E}[\sum^\infty_t \gamma^t r^e_{t+1} | s_t = s, g = g, sg_t = sg]$, which is the discounted cumulative extrinsic rewards expected over all trajectories starting at the state s_t and the sub-goal sg_t. To approximate $Q^e_h(s, g, sg)$, we adopt the Q-learning technique [22] to update the parameters of the high-level network θ_h by Equation 2, where $R^e_1 = r^e(s, a, s', g) + \gamma \max_{sg'} Q^e_h(s', g, sg')$ is the 1-step extrinsic return. The sub-goal sg is given by $\mathrm{argmax}_{sg}\, Q^e_h(s, g, sg)$ towards achieving the final goal g.

$$\theta_h \leftarrow \theta_h - \nabla_{\theta_h} (R^e_1 - Q_{\theta_h}(s, g, sg))^2. \tag{2}$$

Network architecture. To approximate $Q^e_h(s, g, sg)$, we condition the high-level network on the state s, goal g and the sub-goal sg. A widely adopted way is by taking the state s and the goal g as the inputs, and output Q-values that each of them corresponds to a candidate sub-goal sg. Here, since the state s is unknown, we instead take the observation o as the input to our high-level network attempting to estimate the state s simultaneously. To ensure the sub-goal that can be achieved by the low-level network, the sub-goal space at the time t is set as the observable goals within the observation o_t. [2] As a consequence, the sub-goal space varies at each time stamp and is typically much smaller than the goal space. Thus, it is not efficient for the high-level network to calculate as many Q-values as the size of the goal space. Instead, as the sub-goal space is self-contained in the observation o_t, we hereby extract the information of each candidate sub-goal sg from the observation o_t and feed it into the high-level network to output one single Q-value $Q^e_h(s, g, sg)$ specifically for the sub-goal sg.

Last but not the least, although most prior methods directly take the goal description g_d as an additional input to their networks, we notice that the goal description g_d, typically in the form of a one-hot vector or word embedding, does not directly provide any information for either inferring the goal state or determining a quality sub-goal. Therefore, we opt to correlate the goal g with each candidate sub-goal sg by their relations. Here, our system plans over the GRG and gets the cost $\mathbb{C}(\tau^*_{sg,g})$ of the optimal plan $\tau^*_{sg,g}$ from the sub-goal sg to the goal g as described in

[2] In practice, we supplement a back-up "*random*" sub-goal driving the low-level network to randomly pick an action to perform in case no observable goals available.

Section 3.2. We multiply the cost $\mathbb{C}(\tau^*_{sg,g})$ to the sub-goal input sg elementwise before feeding it into the high-level network to predict its Q-value $Q^e_h(s, g, sg)$. In such a way, $Q^e_h(s, g, sg) = Q^e_h(s, sg \odot \mathbb{C}(\tau^*_{sg,g}))$ where the goal g is embedded with beneficial information for the Q-value prediction and the sub-goal selection. As the inputs and the outputs are specified, the remaining architecture of our high-level network is flexible per application.

3.4. Termination-aware Low-level Network

Model formulation. The objective of the low-level network is to learn an optimal action policy to achieve the proposed sub-goal sg. Similar to the high-level network, we adopt a binary intrinsic reward r^i accordingly that $r^i_t(s_{t-1}, a_{t-1}, s_t, sg) = 1$ iff our low-level network achieves the sub-goal sg, and is otherwise 0. The optimal action policy can then be learned by maximizing the expected discounted cumulative intrinsic rewards $\mathbb{E}[\sum_t^\infty \gamma^t r^i_{t+1} | s_t, sg_t, a_t]$. Since the proposed sub-goal sg_t is guaranteed to be observable in the observation o_t, we have a fully observable goal-driven task that can be efficiently solved by the state-of-the-art reinforcement learning algorithms [22, 21, 11].

Adopting a hierarchical model to decompose a complex task into a set of sub-goal-driven simple tasks has been proven to be efficient and effective [19, 24]. Still, it is under the assumption that the goal/sub-goal space is identical to the state space so that any optimal trajectory can be expressed by a sequence of optimal sub-goal-oriented trajectories. In our work, we consider a practical setting in which the goal/sub-goal space is much smaller than the state space, as lots of intermediate states are not of interest in terms of solving the task. Consequently, an optimal trajectory may not be expressed by the limited presented sub-goals on the trajectory as Figure 2 (a) shows. Instead, following the set of available sub-goals proposed could yield a less optimal trajectory as Figure 2 (b) depicts.

To overcome this issue, we further allow the low-level network to terminate at a valuable state before it achieves the proposed sub-goal. The intuition is that along its way to the sub-goal, the agent may reach a state that is better poised for achieving the final goal. Namely, a state where a better sub-goal appears (see Figure 2 (c)). Some prior methods explore modeling a termination function in their formulations [4] or adding a special "stop" action in the action space for an optimal stop policy [36]. However, they unavoidably increase the exploration difficulty and hurt the sample efficiency. Instead, we terminate our low-level network under the supervision of GRG. Whenever a sub-goal sg is received, an optimal plan $\tau^*_{sg,g}$ starting from the sub-goal sg to the goal g over the GRG is generated following Section 3.2. In fact, any goal on the $\tau^*_{sg,g}$ other than sg is a better sub-goal for achieving the final goal g, and once it appears, our low-level network terminates and returns the control back to the high-level network.

Network architecture. We implement the termination mechanism in the low-level policy using the GRG which is decoupled from low-level policy learning. Therefore, the low-level network still addresses the standard fully observable goal-driven task, i.e. predicting the optimal action policy from the current observation o_t that includes the information of the sub-goal sg_t. This can be solved by methods like DQN [22] and A3C [21], without special requirements on the network architecture.

4. Experiments

Our experiments aim to seek the answers to the following research questions, 1) Is GRG able to capture the underlying relations of all goals? 2) Is GRG able to help solve the new, unseen partially observable goal-driven tasks, and if yes, how? 3) How well does the proposed method work for the goal-driven visual navigation task? To answer the first two questions, we conduct evaluation in an unbiased synthetic grid-world domain. To answer the third question, we apply our system on both AI2-THOR [15] and House3D [34] environments for the robotic object search task . [3]

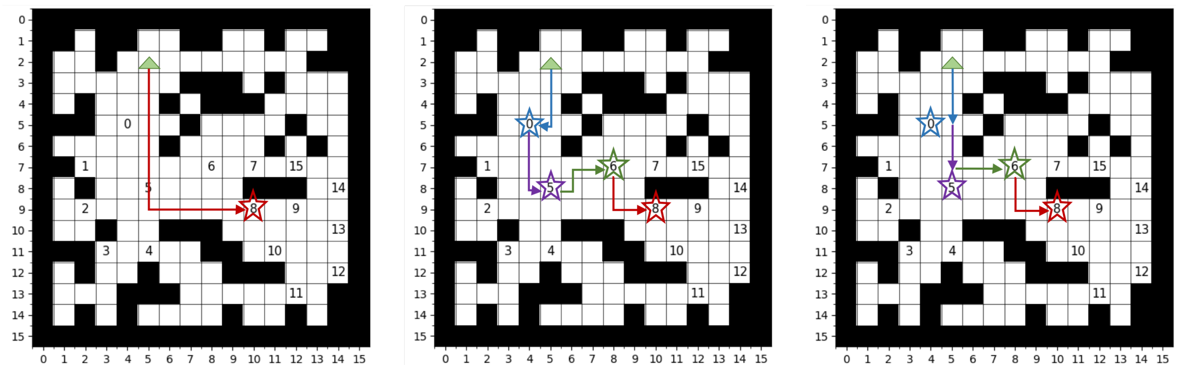
4.1. Grid-world Domain

Grid-world generation. We generate a total of 120 grid-world maps of size 16×16 with randomly placed obstacles taking up around 35% of the space. We arrange 16 goals in the free spaces of each map following a pre-defined pattern to test if our proposed GRG can capture it. Specifically, we randomly place goal g_0 and goal g_8 first. Then, for $0 < i < 16$ and $i \neq 8$, we place goal g_i at a random place in the window of size 7×7 centered at goal g_{i-1}. Figure 1 shows an instance of a grid-world map. We take 100 grid-world maps and 12 goals for training, with the remaining 20 grid-world maps and the corresponding 16 goals are kept for testing.

Baseline methods. We assume the agent can only observe the window of size 7×7 centered at its position, which is represented by an image including the map of obstacles and any goal positions. The agent can take one action as moving up/down/left/right, and would stay at the current position if the action leads to collision. Success is defined as the agent reaches the position of the designated goal. For this task, we adopt the DQN [22] algorithm for our low-level network to learn the optimal action policy to achieve the sub-goal proposed by our high-level network, and we compare our method with the following baseline methods.

- ORACLE and RANDOM. The agent always takes the optimal action or a random action respectively. The two

[3]For technical implementation details, please refer to the supplementary material.



(a) optimal trajectory (b) trajectory w.o. termination (c) trajectory with termination

Figure 2: An illustration of how termination helps. The green triangle denotes the starting position. The stars and the arrows with different colors represent different sub-goals and the corresponding sub-goal-oriented trajectories. Termination helps to express an optimal trajectory with the limited sub-goal space.

Table 1: The performance of all methods on the unseen gird-world maps.

	Seen Goals			Unseen Goals			Overall		
Method	SR↑	AS / MS↓	SPL↑	SR↑	AS / MS↓	SPL↑	SR↑	AS / MS↓	SPL↑
ORACLE	1.00	11.81 / 11.81	1.00	1.00	11.28 / 11.28	1.00	1.00	10.38 / 10.38	1.00
RANDOM	0.16	42.15 / 5.47	0.03	0.15	42.38 / 4.81	0.04	0.18	36.62 / 4.69	0.05
DQN	0.20	20.28 / 5.47	0.13	0.20	11.90 / 4.10	0.15	0.32	16.23 / 5.71	0.23
H-DQN	0.43	**20.25 / 7.95**	0.28	0.19	26.09 / 6.38	0.08	0.45	20.84 / 7.16	0.26
Ours	**0.57**	28.71 / 9.03	**0.33**	**0.70**	**24.19 / 8.73**	**0.45**	**0.74**	**24.02 / 8.65**	**0.46**

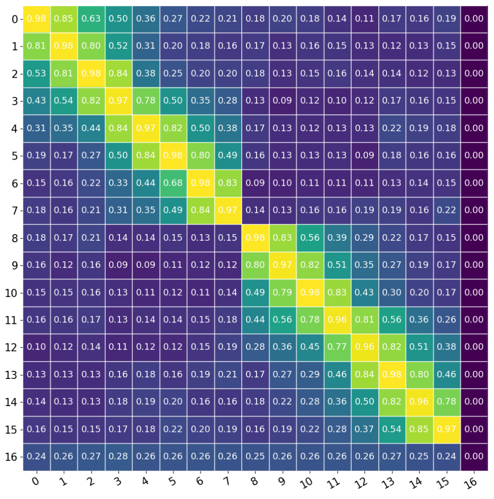


Figure 3: A visualization of a GRG learned on the grid-world domain (g_{16} is the back-up "*random*" goal).

methods are taken as performance upper/lower bounds.

- DQN. The vanilla DQN implementation that directly maps the observation to the optimal action. To make it goal-adaptive, the input observation image contains a channel of the obstacle map and a channel of the designated goal position if it presents. It is empirically shown to be better than embedding the goal with a one-hot vector (see supplementary material).
- H-DQN. It is a widely adopted hierarchical method [17] modified for our partially observable goal-driven task where both the high-level network and the low-level network adopt a vanilla DQN implementation. The high-level network takes the whole observation as the input to propose a sub-goal that is visible. To be goal-adaptive, the goal is embedded into the high-level network in the form of a one-hot encoded vector. The low-level network is the same as the method DQN (also with ours).

Baseline comparisons. We specify the maximum number of actions that all methods can take as 100, and for hierarchical methods, i.e. H-DQN and our method, the maximum number of actions that the low-level network can take at each time is 10. We evaluate all methods in terms of the Success Rate (SR), the Average Steps over all successful cases compared to the Minimal Steps over these cases (AS

Table 2: The ablation studies of our method on the unseen gird-world maps.

	Seen Goals			Unseen Goals			Overall		
Method	SR↑	AS / MS↓	SPL↑	SR↑	AS / MS↓	SPL↑	SR↑	AS / MS↓	SPL↑
Ours	**0.57**	**28.71 / 9.03**	**0.33**	**0.70**	24.19 / 8.73	**0.45**	**0.74**	**24.02 / 8.65**	**0.46**
-relation	0.26	33.20 / 6.16	0.10	0.35	31.93 / 6.84	0.14	0.40	29.39 / 6.14	0.18
-termination	0.55	31.36 / 8.81	0.27	0.58	27.91 / 8.11	0.32	0.64	25.56 / 7.88	0.37
-high-level	0.56	29.97 / 9.03	0.31	0.65	**23.63 / 8.65**	0.42	0.66	22.86 / 7.86	0.41

/ MS), and the Success weighted by inverse Path Length (SPL) following [1] and calculated as $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(l_i, p_i)}$. Here S_i is a binary indicator of success in experiment i, l_i and p_i are the minimal steps and the steps actually taken by the agent. We randomly sample seen goals, unseen goals and all goals over the unseen grid-world maps, each having 100 samples that yield 100 tasks respectively. We run each method using 5 random seeds. Table 1 reports the results.

As is shown in Table 1, we can observe that our method outperforms all baseline methods in terms of generalization ability on the unseen grid-world maps as expected. On one hand, the performance of DQN leaves much to be desired for both seen goals and unseen goals. Whereas H-DQN achieves comparable performance to our method for seen goals, but it struggles to generalize towards unseen goals. On the other hand, our proposed method generalizes well to both seen goals and unseen goals, since our GRG captures the underlying relations of all goals, even if some of the goals are not set as the designated goals in the training stage. Figure 3 shows a visualization of the learned GRG, which captures the goal relations well.

Ablation studies. To investigate *how* GRG helps to solve the partially observable goal-driven task, we conduct ablation studies for each component. The GRG has two roles: In the high-level network, it weighs each candidate sub-goal by its relation to the final goal before calculating its Q-value. In the low-level network, it is used for early termination. We disable each role and denote them as "-relation" and "-termination" respectively. The results reported in Table 2 clearly show that both of them contribute to the performance of our proposed method, whereas weighing the candidate sub-goals by relations contributes more. Moreover, to show the necessity of the high-level network, we present "-high-level" that removes the high-level network, leaving only the GRG and the low-level network in place. In such a way, a sub-goal is proposed purely based on the graph planning over GRG without taking the current observation into consideration. The results in Table 2 show that it is slightly worse than our proposed method; from which we can infer that 1) the high-level network captures as much information as the GRG; 2) observations still matter since the graph only captures the expected relations; and 3) the performance gap could be wider in complex real-world environments.

4.2. Robotic Object Search

Robotic object search is a challenging goal-driven visual navigation task [36, 38, 23, 26, 37, 27, 5]. It requires an agent to search for and navigate to an instance of a user-specified object category in indoor environments with only its first-person view RGB image.

A previous method SCENE PRIORS [36] also incorporates object relations as scene priors to improve the robotic object search performance in the AI2-THOR [15] environments. Unlike ours, it extracts the object relations from the Visual Genome [16] corpus and incorporates the relations through Graph Convolutional Networks [14]. Therefore, we compare our method with it in the AI2-THOR environments. AI2-THOR consists of 120 single functional rooms, including kitchens, living rooms, bedrooms and bathrooms, in which we take the first-person view semantic segmentation and depth map as the agent's pre-processed observation. As such, the goal position can be represented by the corresponding channel of the semantic segmentation (a.k.a. Φ). In addition, we adopt the A3C [21] algorithm for our low-level network and define the maximum steps it can take at each time as 10. We follow the experimental setting in [36] to implement both SCENE PRIORS [36] and our HRL-GRG. We report the results in Table 4 where we compare the two methods in terms of their performance improvement over the RANDOM method. Table 4 indicates an overfitting issue of the SCENE PRIORS method as reported in [36] as well. At the same time, we observe a superior generalization ability of our method especially to the unseen goals.

To further demonstrate the efficacy of our method in more complex environments, we conduct robotic object search on the House3D [34] platform. Different from AI2-THOR, each house environment in the House3D has multiple functional rooms that are more likely to occlude the user-specified target object, thus stressing upon the ability of inferring the target object's location on the fly to perform the task well.

We consider a total of 78 object categories in the House3D environment to form our goal space. The agent moves for-

Table 3: The performance of all methods in the House3D [34] environment for the robotic object search task.

	Single Environment				Multiple Environments			
	Seen Goals		Unseen Goals		Seen Env.		Unseen Env.	
Method	SR↑	SPL↑	SR↑	SPL↑	SR↑	SPL↑	SR↑	SPL↑
RANDOM	0.20	0.05	0.23	0.04	0.39	0.03	0.60	0.05
DQN	0.58	0.27	0.18	0.05	0.42	0.06	0.39	0.04
A3C	0.53	0.18	0.27	0.09	0.48	0.03	0.47	0.03
HRL	0.77	0.15	0.05	0.00	0.43	0.05	0.28	0.02
Ours	**0.88**	**0.33**	**0.79**	**0.21**	**0.76**	**0.20**	**0.62**	**0.10**

Table 4: The performance improvement of SCENE PRIORS [36] (top) and our HRL-GRG (bottom) over the RANDOM method in the AI2-THOR [15] environment for the robotic object search task (without stop action).

		Seen Goals		Unseen Goals	
		SR↑	SPL↑	SR↑	SPL↑
Seen Env.	[36]	+0.25	+0.16	+0.08	+0.07
	Ours	**+0.37**	**+0.24**	**+0.33**	**+0.23**
Unseen Env.	[36]	+0.18	+0.11	+0.12	+0.06
	Ours	**+0.33**	**+0.21**	**+0.38**	**+0.23**

ward / backward / left / right 0.2 meters, or rotates 90 degrees for each action step. We adopt the encoder-decoder model from [7] to predict both the semantic segmentation and the depth map from the first-person view RGB image and we take both predictions as the agent's partial observation. Furthermore, we adopt the A3C [21] algorithm for the low-level network. We compare our method with the baseline methods introduced in Section 4.1 while we also adopt the A3C algorithm for the low-level network in H-DQN and hereby denoted as HRL. In addition, we include the vanilla A3C approach.

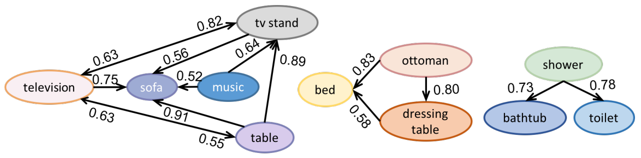


Figure 4: The object relations captured by our GRG in the House3D [34] environment for the robotic object search task. Only a small number of objects as nodes and the edges with the weight ≥ 0.5 are shown.

We set the maximum steps for all the aforementioned methods to solve the object search task in the House3D environment as 1000, and the maximum steps that the low-level networks of the hierarchical methods (HRL and ours) can take as 50. To better investigate each method's properties, we first train and evaluate in a single environment, and show the results in Table 3 (left part). Similar to the grid-world domain, the baseline methods lack generalization ability towards achieving the unseen goals, even though they perform fairly well for the seen ones. The placement of many objects is subject to the users' preference that may require the environment-specific training process. Still, it is desirable for a method to generalize towards the objects in the unseen environments where the placement of the objects is consistent with that in the seen ones (e.g., the objects that are always placed in accordance with their functionalities). We train all the methods in four different environments and test the methods in four other unseen environments. The results presented in Table 3 (right part) show that all the baselines struggle with the object search task under multiple environments even during the training stage. In comparison, our method achieves far superior performance with the help of the object relations captured by our GRG (samples shown in Figure 4).

5. Conclusion

In this paper, we present a novel hierarchical reinforcement learning approach equipped with a GRG formulation for the partially observable goal-driven task. Our GRG captures the underlying relations among all goals in the goal space through a Dirichlet-categorical model and thus enables graph-based planning. The planning outputs are further incorporated into our two-layer hierarchical RL for proposing sub-goals and early low-level layer termination. We validate our approach on both the grid-world domain and the challenging robotic object search task. The results show our approach is effective and is exceptional in generalizing to unseen environments and new goals. We argue that the joint learning of GRG and HRL boosts the overall performance on the tasks we perform in our experiments, and it may push forward future research ventures in combining symbolic reasoning with DRL.

Acknowledgements. This work is partially supported by the NSF grant #1750082, and Samsung Research.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 7

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018. 2

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[4] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 5

[5] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020. 2, 7

[6] 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. 2

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 8

[8] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering. In *Conference on Robot Learning*, pages 53–62, 2018. 1

[9] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pages 3314–3325, 2018. 2

[10] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017. 2

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. 1, 2, 5

[12] Dongqi Han, Kenji Doya, and Jun Tani. Variational recurrent models for solving partially observable control tasks. *arXiv preprint arXiv:1912.10703*, 2019. 1, 2

[13] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. *arXiv preprint arXiv:1806.02426*, 2018. 1, 2

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2, 7

[15] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 2, 5, 7, 8

[16] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017. 2, 7

[17] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016. 6

[18] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019. 1, 2

[19] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017. 5

[20] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 1

[21] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016. 1, 2, 5, 7, 8

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 1, 2, 4, 5

[23] Arsalan Mousavian, Alexander Toshev, Marek Fišer, Jana Košecká, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8846–8852. IEEE, 2019. 2, 7

[24] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018. 5

[25] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, pages 14843–14854, 2019. 1

[26] Tai-Long Nguyen, Do-Van Nguyen, and Thanh-Ha Le. Reinforcement learning based navigation with semantic knowledge of indoor environments. In *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–7. IEEE, 2019. 2, 7

[27] Yiding Qiu, Anwesan Pal, and Henrik I Christensen. Target driven visual navigation exploiting object relationships. *arXiv preprint arXiv:2003.06749*, 2020. 2, 7

[28] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018. 2

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 1

[30] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 1

[31] Stephen Tu. The dirichlet-multinomial and dirichlet-categorical models for bayesian inference. *Computer Science Division, UC Berkeley*, 2014. 2

[32] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019. 2

[33] Yuechen Wu, Zhenhuan Rao, Wei Zhang, Shijian Lu, Weizhi Lu, and Zheng-Jun Zha. Exploring the task cooperation in multi-goal visual navigation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 609–615. AAAI Press, 2019. 2

[34] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018. 2, 5, 7, 8

[35] Yi Wu, Yuxin Wu, Aviv Tamar, Stuart Russell, Georgia Gkioxari, and Yuandong Tian. Bayesian relational memory for semantic visual navigation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2769–2779, 2019. 2

[36] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018. 2, 5, 7, 8

[37] Xin Ye, Zhe Lin, Joon-Young Lee, Jianming Zhang, Shibin Zheng, and Yezhou Yang. Gaple: Generalizable approaching policy learning for robotic object searching in indoor environment. *IEEE Robotics and Automation Letters*, 4(4):4003–4010, 2019. 7

[38] Xin Ye, Zhe Lin, Haoxiang Li, Shibin Zheng, and Yezhou Yang. Active object perceiver: Recognition-guided policy learning for object searching on mobile robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6857–6863. IEEE, 2018. 2, 7

[39] Xin Ye and Yezhou Yang. Efficient robotic object search via hiem: Hierarchical policy learning with intrinsic-extrinsic modeling. *IEEE Robotics and Automation Letters*, 2021. 2

[40] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017. 2