Efficient Robotic Object Search Via HIEM: Hierarchical Policy Learning With Intrinsic-Extrinsic Modeling

Xin Ye and Yezhou Yang

Abstract—**Despite the significant success at enabling robots with autonomous behaviors makes deep reinforcement learning a promising approach for robotic object search task, the deep reinforcement learning approach severely suffers from the nature sparse reward setting of the task. To tackle this challenge, we present a novel policy learning paradigm for the object search task, based on hierarchical and interpretable modeling with an intrinsic-extrinsic reward setting. More specifically, we explore the environment efficiently through a proxy low-level policy which is driven by the intrinsic rewarding sub-goals. We further learn our hierarchical policy from the efficient exploration experience where we optimize both of our high-level and low-level policies towards the extrinsic rewarding goal to perform the object search task well. Experiments conducted on the House3D environment validate and show that the robot, trained with our model, can perform the object search task in a more optimal and interpretable way.**

Index Terms—**Reinforcement learning, semantic scene understanding, vision-based navigation.**

I. Introduction

ROBOTIC object search is a task where a robot (with an on-board camera) is expected to take reasonable steps to approach a user-specified object in an unknown indoor environment. It is an essential capability for assistant robots and could serve as an enabling step for other tasks, such as the Embodied Question Answering [1]. Classical map-based approaches like simultaneous localization and mapping (SLAM) have been studied to address this problem for a long time, but it is also well-known that SLAM-based approaches rely heavily on sensor inputs and thus suffer from sensor noises [2], [3]. Recently, (deep) reinforcement learning (RL) has demonstrated its power at enabling robots with autonomous behaviors [4], such as navigating over an unknown environment [5], [6], manipulating objects with robot's end effectors [7]–[9], and motion planning [10], [11]. Under the RL setting, a robot learns the optimal behavioral policy by maximizing the expected cumulative rewards given the samples collected from its physical and/or virtual interactions with the environment. The rewards serve as the reinforcement signals for the robot to update its policy.

A pressing challenge to train a robot to perform object search with RL is the sparse reward issue, due to the fact that the environment and/or the location of the target object are typically unknown. With well-designed reward functions, such as the ones in Atari games [12], the learned policies are shown to achieve extremely promising performance. However, it is a well-known challenge designing the reward function for the real-world applications [13]. Typically, for applications such as object search or target-driven visual navigation, prior research constructs the reward function in terms of the distance between the robot's current location and the object location under a strict assumption that the full information of the environment is known [14]–[16]. For an unknown environment, a straightforward way is to set a high reward when the robot reaches the final goal state while at all other intermediate states, the reward is either zero or a small negative value [6]. More recently, [17] presented a relatively denser reward function which is based on the bounding box of the target object from the robot's detection system, but the reward is still not defined among the situations where the target object is not detected. In such a sparse reward setting where the reward is only defined for a small subset of the states, the robot struggles to learn the object search policy as it is unlikely to encounter and sample the very few rewarding states without a well-designed goal-oriented exploration strategy, especially dealing with complex environments.

Hierarchical RL (HRL) paradigm is thus formulated considering its efficient strategy for exploration [18] and superiority under the sparse reward setting [19]–[21]. HRL aims to learn multiple layers of policies. The higher layer breaks down the task into several easier sub-tasks and proposes corresponding sub-goals for the lower layer to achieve. Typically, the sub-goals are aliases to the states that mandates the lower layer to reach, as defined in [20], [22] for tasks with low dimensional state spaces. Unfortunately, these methods are not directly applicable for the object search task in which the state observations are directly taken from the high dimensional RGB images. It is utterly difficult and seemingly impractical for the higher layer to output homogeneous images as sub-goals. On the other hand, reconstructing a concise low dimensional sub-goal space from the observation space without compromising the optimality of the learned policy demands elaborate efforts [23], [24].

Manuscript received October 15, 2020; accepted February 16, 2021. Date of publication March 25, 2021; date of current version April 9, 2021. This letter was recommended for publication by Associate Editor E. Ricci and Editor E. Marchand upon evaluation of the reviewers' comments. This work was supported by the NSF Robust Intelligence Program Project #1750082, and Samsung Research. *(Corresponding author: Xin Ye.)*

The authors are with the Active Perception Group, the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ USA (e-mail: xinye1@asu.edu; yz.yang@asu.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3068906, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3068906

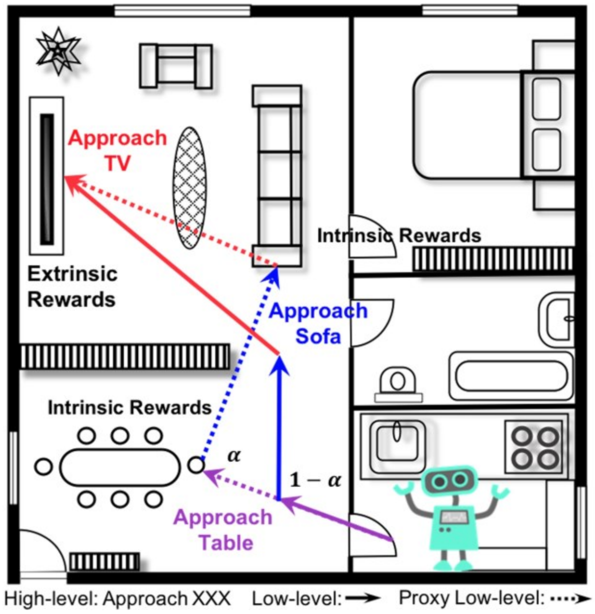


Fig. 1. An example of our *HIEM* framework. When our high-level policy proposes a sub-goal, our proxy low-level policy is invoked with the probability of α to explore the environment by optimizing towards the sub-goal, and our low-level policy learned from the exploration experience is invoked with the probability of 1 _α to collaborate with the high-level policy to better achieve the goal.

In this paper, we put forward a novel two-layer hierarchical policy learning paradigm for the object search task. Our hierarchical policy builds on a simple yet effective and interpretable low dimensional sub-goal space. To obtain an optimal hierarchical policy given the small sub-goal space, we model the object search task with both goal dependent extrinsic rewards and sub-goal dependent intrinsic rewards. To be specific, our high-level policy plans over the sub-goal space in order to achieve the final goal by maximizing the extrinsic rewards. When a sub-goal is given following the high-level policy, a proxy low-level policy is then invoked for the robot to explore the environment. The proxy low-level policy maximizes the intrinsic rewards in order to achieve the proposed sub-goal. Meanwhile, our low-level policy learns from the exploration experience and optimizes towards the final goal. It is invoked eventually to collaborate with our high-level policy to form an optimal hierarchical object search sequence. Moreover, inspired by [25], the low-level policy learns to terminate at valuable states that further improves our hierarchical object search performance. We dub our framework as *HIEM:* Hierarchical policy learning with Intrinsic-Extrinsic Modeling (see Fig. 1). We validate *HIEM* with extensive sets of experiments on the House3D [26] simulation environment which contains thousands of 3D houses with a diverse set of objects and natural layouts resembling the real-world. The observed results demonstrate the efficiency and efficacy of our system over other state-of-the-art ones.

II. Related Work

Our work is closely related to two major research thrusts: hierarchical RL and target-driven visual navigation.

Hierarchical reinforcement learning. Previous work has studied hierarchical reinforcement learning in many different ways. One is to come up with efficient methods to accelerate the learning process of the general hierarchical reinforcement learning scheme. As in [22], the authors introduce an off-policy correction method. [27] and [21] propose to use Hindsight Experience Replay to facilitate learning at multiple time scales. Though these methods' performance are impressive, they typically assume the sub-goal space for the higher level policy is the state space. However, in the object search task, the RL system takes the image as the state representation, these methods are not directly applicable since the higher layer can hardly propose an image as a sub-goal for the lower layer to achieve.

Other methods designate a separate sub-goal space for hierarchical reinforcement learning. For example, [19] defines the sub-goal space in the space of entities and relations, such as the "reach" relation they use for their Atari game experiment. Sub-tasks and their relations are provided as inputs in [28] and [29]. Closer related to our work, [30] adopts *{exit-room, find-room, find-object, answer}* as the sub-goal space to learn a hierarchical policy for the Embodied Question Answering task. For the same task, [31] chooses *{navigate, scan, detect, manipulate, answer}* as the possible sub-tasks, while the reinforcement learning methods are mainly applied for learning high-level policy, i.e. planning over the pre-trained or fixed sub-tasks.

On the other side, attempts have been made to learn a set of low-level skills automatically to achieve the goal. These low-level skills are also referred to as temporal abstractions. [25] proposes the option-critic framework to autonomously discover the specified number of temporal abstractions. [32] learns the temporal abstractions through advantage-weighted information maximization. [23] addresses the sub-goal representation learning problem. With the learned representation, their hierarchical policies are shown to approach the optimal performance within a bounded error.

Motivated by aforementioned ones, we designate a simple yet effective sub-goal space that makes the hierarchy better interpretable. Meanwhile, to make the optimal policy expressible and learnable with the specified sub-goal space, we also leverage the benefits from the automatic temporal abstraction learning methods, which ultimately yields a hybrid system.

Target-driven visual navigation. Deep reinforcement learning has been studied extensively for the target-driven visual navigation tasks [33]. These tasks can be categorized in terms of the description of the navigation target. [6], [17] and [34] specify the navigation target by the image taken at the target location. The robotic object search task studied in [14], [35]–[37] and the room navigation task introduced in [26], [38] take the semantic label of the target object and room as the navigation target. The Embodied Question Answering [1], [30], [31] and the Vision-and-Language Navigation [16], [39] address the problem where the navigation target is provided with an unconstrained natural language. Here, we study the robotic object search task where the navigation target is an object specified by a semantic label. Unlike the previous work that plans over the atomic actions for navigation [14], [26], [35]–[37], we learn a hierarchical policy that performs the robotic object search task in a more interpretable way. While [30], [31] and [38] also study hierarchical policies, their low-level policies focus only on the sub-tasks without keeping the final navigation target in mind, thus may yield less optimal policies towards the final navigation target.

Notably, many of the previous works address the sparse reward issue by introducing additional supervision under the assumption that the robot can access the full information of the environments during the training time, such as defining the reward function with the distance between the robot's current location and the target location (a.k.a. reward shaping) [14], [26], adopting shortest path as the supervised signal for pre-training [1], [16], and/or gradually increasing the distance between robot's starting location and the target location (a.k.a. curriculum learning) [30], [34]. Nevertheless, for applications in real-world environments, collecting all the information is unarguably expensive and sometimes impractical. *We would like to stress upon the point that our model does not assume any environment information available even during the training stage, which makes our object search task significantly more challenging.*

III. Our Approach

First, we define the robotic object search task. Formally speaking, when a target object is specified and provided with a semantic label, the robot is asked to search and approach the object from its random starting position. The RGB image from the robot's on-board camera is the only source of information for decision making. None of the environment information, such as the map of the environment or the location of the target object could be accessed. Once the area of the target object in the robot's viewpoint (the image captured by its camera) is larger than a predefined threshold, the robot stops and we consider it as a success. In this work, we present a novel two-layer hierarchical policy for the robot to perform the object search task, motivated by how human beings typically conduct object search. In the following sections, we first describe the hierarchy of policies. Then we introduce two kinds of reward functions, i.e. extrinsic rewards and intrinsic rewards, and we make use of these two reward functions to formulate the solution. Finally, we describe the network architecture adopted for learning the two-layer hierarchical policy.

A. Hierarchy of Policies

Our hierarchical policy has two levels, a high-level policy π_h and a low-level policy π_l. At time step t, the robot takes the image captured by its camera as the current state s_t. Given a target object or goal g, the high-level layer proposes a sub-goal $sg_t \sim \pi_h(sg|s_t, g)$ and the low-level layer takes over the control. The low-level layer then draws an atomic action $a_t \sim \pi_l(a|s_t, g, sg_t)$ to perform. The robot will receive a new image/state s_{t+1}. The low-level layer repeats N_t times till 1) the low-level layer terminates itself following the termination signal $term(s_{t+N_t}, g, sg_t)$; 2) the low-level layer achieves the sub-goal sg_t. 3) the low-level layer has performed a predefined maximum number of atomic actions. Either way, the low-level layer terminates at state s_{t+N_t}, and then returns the control back to the high-level layer, and the high-level layer proposes another sub-goal. This process repeats until 1) the goal g is achieved, i.e. the robot finds the target object successfully; 2) a predefined maximum number of atomic actions has been performed.

For the object search task, we define the sub-goal space as *{approach $obj|obj$ is visible in the robot's current view}*. We argue three reasons for the sub-goal space definition, a) approaching an object that shows in the robot's view is a more general and relatively trainable task shown by [35]. It also aligns well with the objective of the hierarchical reinforcement learning by breaking down the task into several easier sub-tasks; b) approaching a related object may increase the probability of seeing the target object. As soon as the target object is captured in the robot's current view, the task becomes an object approaching task; c) as also suggested by [19], specifying sub-goals over entities and relations can provide an efficient space for exploration in a complex environment. Moreover, in case there is no object visible in the robot's current view, we supplement a back-up "random" sub-goal invoking a random low-level policy. The atomic action space for the low-level layer is defined for navigation purpose, namely *{move forward / backward / left / right, turn left / right}* in which the *move* action updates the robot's location only and the *turn* action drives the robot's rotation only.

B. Extrinsic Rewards and Intrinsic Rewards

We define two kinds of reward functions. The extrinsic rewards r^e are defined for our object search task, thus are goal dependent. Further, we also introduce the intrinsic rewards r^i for the low-level sub-tasks. The intrinsic rewards are hereby sub-goal dependent. We specify the two reward functions respectively as follows.

Extrinsic rewards r^e. Without loss of generality, to encourage the robot to finish the object search task, we provide a positive extrinsic reward (in practice, 1) when the robot reaches the final goal state. At all other intermediate states, the extrinsic rewards are set to 0. Formally, $r^e_t(s_{t-1}, a_{t-1}, s_t, g) = 1$ if and only if s_t is a goal state of the goal g, otherwise $r^e_t(s_{t-1}, a_{t-1}, s_t, g) = 0$.

Intrinsic rewards r^i. To facilitate the robot perform the sub-task, i.e. approaching the object specified in the proposed sub-goal sg which shows in the robot's current view, we adopt the similar binary rewards. To be specific, the intrinsic reward $r^i_t(s_{t-1}, a_{t-1}, s_t, sg) = 1$ if and only if s_t is a goal state of the sub-goal sg, otherwise $r^i_t(s_{t-1}, a_{t-1}, s_t, sg) = 0$.

C. Model Formulation

We formulate the object search task in terms of the two rewards introduced in Sec. III-B. When the robot starts from an initial state s_0, it proposes a sub-goal sg_0 aiming to achieve the final goal g (locating and approaching the target object). To achieve the final goal, we can optimize the discounted cumulative extrinsic rewards, expected over all trajectories starting at state s_0 and sub-goal sg_0, which is $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r^e_{t+1} | s_0, g, sg_0]$. If and only if the robot takes minimal steps to the goal state, the discounted cumulative extrinsic rewards are thus maximized.

The discounted cumulative extrinsic rewards is also known as the state action value Q^e_h [40] for our high-level layer, i.e. $\mathbb{E}[\sum_{t=0}^{\infty}\gamma^t r^e_{t+1}|s_0 = s, g = g, sg_0 = sg] = Q^e_h(s, g, sg)$. Following the option-critic framework [25], we unroll the $Q^e_h(s, g, sg)$ as,

$$\begin{aligned} Q^e_h(s, g, sg) &= \sum_a \pi_l(a|s, g, sg)\mathbb{E}\sum_{t=0}^{\infty}\gamma^t r^e_{t+1}|s_0 = s, g = g, sg_0 = sg, a_0 = a \\ &= \sum_a \pi_l(a|s, g, sg)Q^e_l(s, g, sg, a), \end{aligned} \quad (1)$$

where the state action value $Q^e_l(s, g, sg, a)$ for our low-level layer is the discounted cumulative extrinsic rewards after taking action a under the state s, goal g and sub-goal sg. Given the transition probability $P(s'|s, a)$ which denotes the probability of being state s' after taking action a at state s, $Q^e_l(s, g, sg, a)$ can be further formulated as,

$$\begin{aligned} Q^e_l(s, g, sg, a) &= \sum_{s'} P(s'|s, a)[r^e(s, a, s', g) + \gamma U(g, sg, s')], \\ U(g, sg, s') &= (1 - term(s', g, sg))Q^e_h(s', g, sg) \\ &\quad + term(s', g, sg)V^e_h(s', g), \\ V^e_h(s', g) &= \sum_{sg} \pi_h(sg|s', g)Q^e_h(s', g, sg). \end{aligned} \quad (2)$$

We parameterize $Q^e_h(s, g, sg)$, $Q^e_l(s, g, sg, a)$ and $term(s, g, sg)$ with θ^e_h, θ^e_l and θ_t respectively. Then the high-level policy $\pi_h(sg|s, g) = \mathbb{1}(sg = \mathrm{argmax}_{sg} Q^e_h(s, g, sg))$, and $\pi_l(a|s, g, sg) = \mathbb{1}(a = \mathrm{argmax}_a Q^e_l(s, g, sg, a))$ is our low-level policy. We adopt the DQN [12] based method to learn $Q^e_h(s, g, sg)$ and $Q^e_l(s, g, sg, a)$ in which we update both of the values towards the 1-step extrinsic return $R^e_1 = r^e(s, a, s', g) + \gamma U(g, sg, s')$, and consequently θ^e_h and θ^e_l can be updated by Equation 3 and 4. In addition, θ_t can be updated by Equation 5 as demonstrated by [25].

$$\theta^e_h \leftarrow \theta^e_h - \nabla_{\theta^e_h}[R^e_1 - Q_{\theta^e_h}(s, g, sg)]^2. \quad (3)$$

$$\theta^e_l \leftarrow \theta^e_l - \nabla_{\theta^e_l}[R^e_1 - Q_{\theta^e_l}(s, g, sg, a)]^2. \quad (4)$$

$$\theta_t \leftarrow \theta_t - \nabla_{\theta_t} term_{\theta_t}(s', g, sg)(Q^e_h(s', g, sg) - V^e_h(s', g)). \quad (5)$$

Since the robot may start at a position far away from the target object, it is unlikely for the robot to encounter the sparse extrinsic rewarding states through the ϵ-greedy [12] exploration policy and collect the experience samples to effectively train θ^e_h, θ^e_l and θ_t. On the contrary, encountering the intrinsic rewarding states is much more possibly as an object shows in the robot's current view is usually nearby. Therefore, training the robot to achieve a sub-goal is more accessible. Then, by iteratively asking the robot to achieve suitable sub-goals, i.e. to approach related objects, the robot is more likely to observe the target object and collect the valuable experience samples to train θ^e_h, θ^e_l and θ_t.

We hereby define a proxy low-level policy $\pi^p_l(a|s, sg)$ aiming to achieve the proposed sub-goal sg. Similarly, we learn the proxy low-level policy by optimizing the discounted cumulative intrinsic rewards $Q^i_l(s, sg, a)$. We adopt the DQN method [12] to learn it by updating its parameter θ^i_l with Equation 6, where $R^i_1 = r^i(s, a, s', sg) + \gamma \max_a Q^i_l(s', sg, a)$ is the 1-step intrinsic return. As a result, the proxy low-level policy $\pi^p_l(a|s, sg) = \mathbb{1}(a = \mathrm{argmax}_a Q^i_l(s, sg, a))$.

$$\theta^i_l \leftarrow \theta^i_l - \nabla_{\theta^i_l}[R^i_1 - Q_{\theta^i_l}(s, sg, a)]^2. \quad (6)$$

For our low-level layer to balance between exploitation by achieving the goal g with the policy $\pi_l(a|s, g, sg)$ and the exploration by achieving the sub-goal sg with the proxy policy $\pi^p_l(a|s, sg)$, we introduce a hyper-parameter $a \in [0, 1]$ as the probability that the low-level layer adopts the proxy policy $\pi^p_l(a|s, sg)$ to explore the environment and collect the experience samples. The experience samples are used to batch train θ^e_h, θ^e_l, θ_t and θ^i_l with Equation (3), (4), (5) and (6) respectively. In practice, a decays from 1 to 0 across the training episodes to enable our low-level layer to act optimally towards the goal with the policy $\pi_l(a|s, g, sg)$ eventually.

D. HIEM Network Architecture

Since the image captured by the robot's on-board camera serves as the robot's current state, we adopt deep neural networks as θ^e_h, θ^e_l, θ_t and θ^i_l to handle the high dimensional inputs and approximate $Q^e_h(s, g, sg)$, $Q^e_l(s, g, sg, a)$, $term(s, g, sg)$ and $Q^i_l(s, sg, a)$.

Fig. 2 illustrates our network architecture. For the object search task, semantic segmentation and depth map are necessary for the robot to detect the target object and avoid collision during the navigation. Therefore, we first adopt the encoder-decoder network [35] to predict the semantic segmentation and the depth map from the robot's observation. We take the predicted results as the inputs to our policy networks to avoid the need of visual domain adaption [14]. The predicted results of the 4 history observations are fed into our high-level network θ^e_h in addition to a one-hot vector representing the target object. The channel size of the segmentation input is first reduced to 1 through a convolutional layer with 1 filter of kernel size 1×1, and then the three inputs are fed into three different fully connected layers respectively and their outputs are further concatenated into a joint vector before attaching another fully connected layer to generate an embedding fusion. Our high-level network θ^e_h feeds the embedding fusion into one additional fully connected layer to approximate $Q^e_h(s, g, sg)$. To save the number of parameters, our termination network θ_t shares most parameters with the high-level network θ^e_h except the last fully connected layer where it adopts a new one to approximate $term(s, g, sg)$.

For the low-level network θ^e_l and θ^i_l, we take the sub-goal specified channel of the predicted semantic segmentation and the predicted depth map as the inputs. The low-level network θ^e_l takes the one-hot vector of the target object as an additional input. Similar to our high-level network, each input of θ^e_l and θ^i_l is fed into a fully connected layer before being concatenated together to generate an embedding fusion with a new fully connected

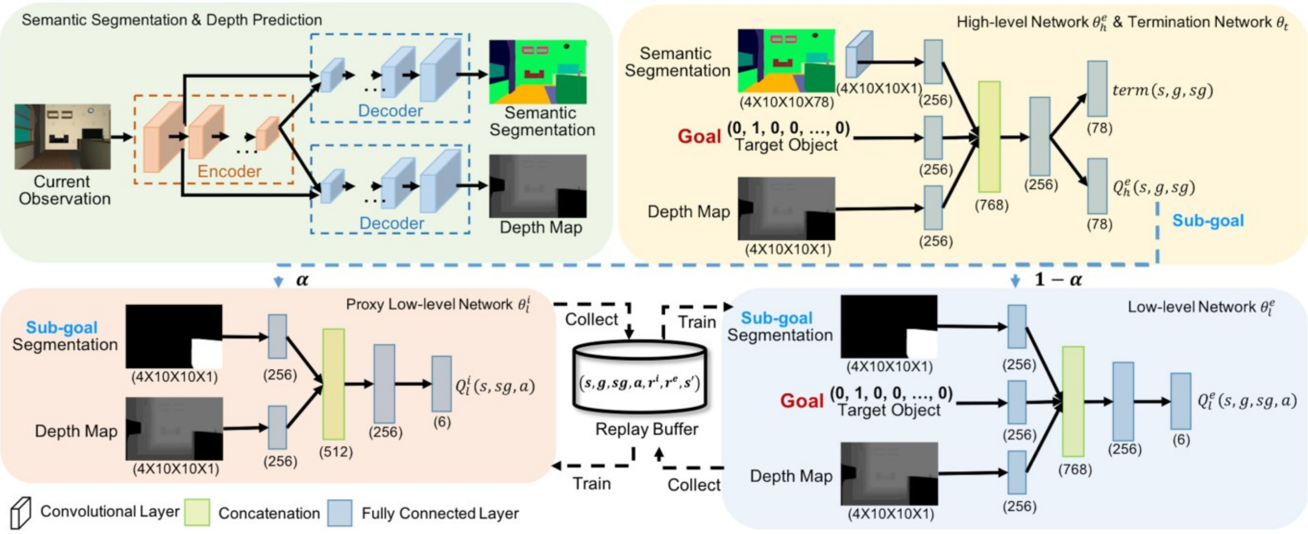


Fig. 2. Network architecture of our hierarchical reinforcement learning model.

layer. The embedding fusion is further fed into an additional fully connected layer to approximate $Q_l^e(s, g, sg, a)$ and $Q_l^i(s, sg, a)$.

We follow Equation 3, 4, 5 and 6 to learn $Q_h^e(s, g, sg)$, $Q_l^e(s, g, sg, a)$, $term(s, g, sg)$ and $Q_l^i(s, sg, a)$ respectively.

IV. Experiments

A. Dataset

We validate our framework on the simulation platform House3D [26]. House3D consists of rich indoor environments with diverse layouts for a virtual robot to navigate. In each indoor environment, a variety of objects are scattered at many locations, such as *television, sofa, desk*. While navigating, the robot has a first-person view RGB image as its observation. The simulator also provides the robot with the ground truth semantic segmentation and depth map corresponding to the RGB image. The RGB images, as well as the semantic segmentation and depth maps can be used as the training data to learn the encoder-decoder network [35] (shown in Fig. 2 upper left) for semantic segmentation and depth prediction as we mentioned in Sec. III-D. We refer interested readers to [35] for more details. In addition, the trained model, specifically the semantic segmentation prediction, can be used as the robot's detection system.

To validate our proposed method in learning hierarchical policy for object search, we conduct the experiments in an indoor environment where the objects' placements are in accordance with the real-world scenario. For example, the *television* is placed close to the *sofa*, and is likely occluded by the *sofa* at many viewpoints. In such a way, to search the target object *television*, the robot could approach *sofa* first to increase the likelihood of seeing the *television*.

We consider discrete actions for the robot to navigate in this environment. Specifically, the robot moves forward / backward / left / right 0.2 meters, or rotates 90 degrees every time. We also discretize the environment into a certain number of reachable locations, as shown in Fig. 3.

B. Experimental Setting

We compare the following methods and variants:

Oracle and **Random**. At each time step, the robot ignores its observation and performs the optimal action and a random action respectively.

A3C [41]. The vanilla A3C implementation that has been wildly adopted for the navigation task in the previous work [6], [17], [35]–[37]. It learns the action policy $\pi(a|s, g)$ and the state value $V^e(s, g)$ with a similar network architecture as our high-level network θ_h^e.

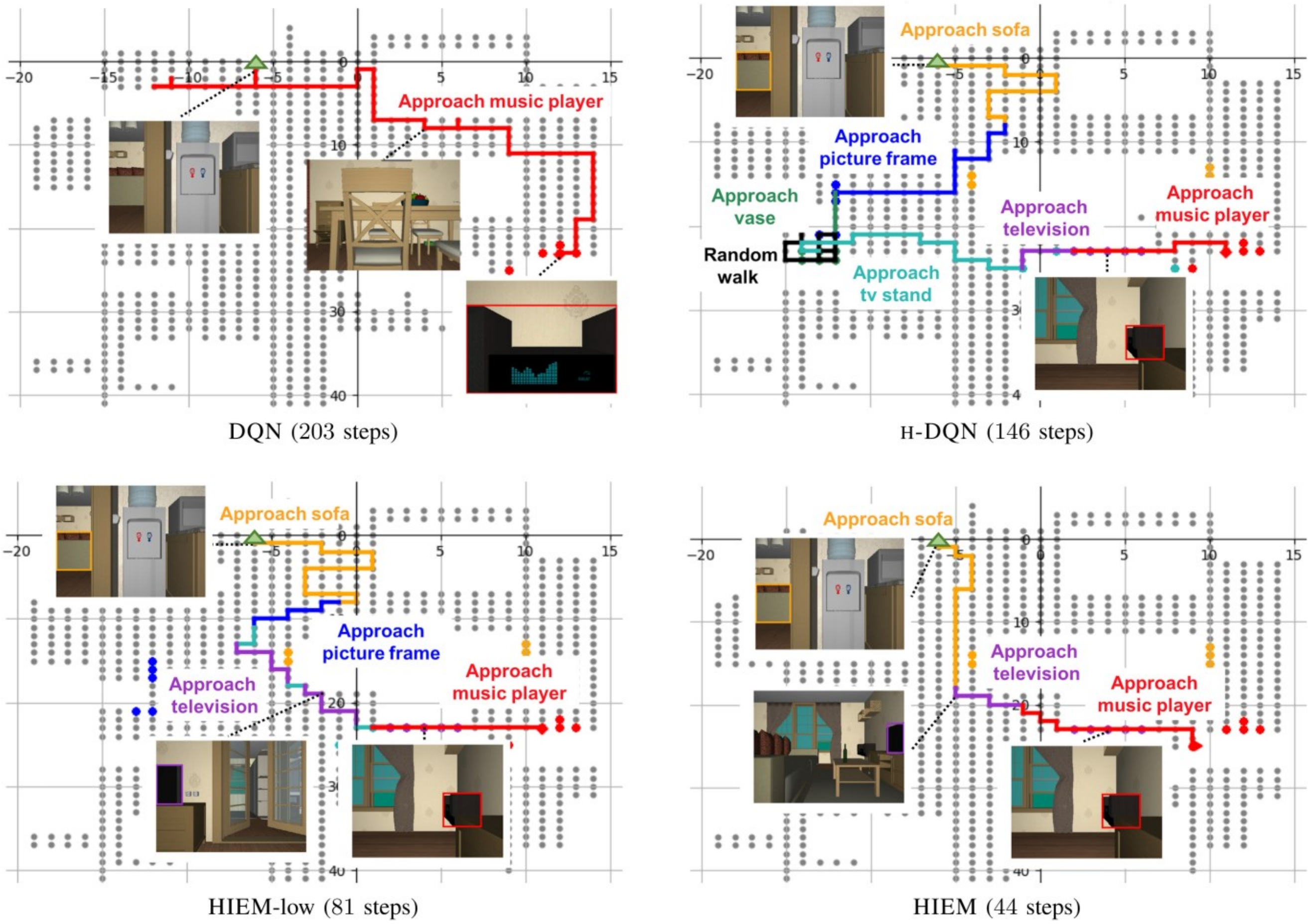
DQN [12]. The vanilla DQN implementation that adopts a similar network architecture as our high-level network θ_h^e to predict the state action value $Q^e(s, g, a)$.

OC [25]. The Option-Critic implementation that learns a hierarchical policy autonomously by maximizing the discounted cumulative extrinsic rewards where only the number of the options needs to be manually set. We set it as 4 in our experiments.

h-DQN [19] with our proposed sub-goal space. It is equivalent to our method when we set $term(s, g, sg) = 0$ and $a = 1$ to disable both the termination network θ_t and the low-level network θ_l^e.

HIEM. Our method follows Sec III. To further identify the role of each component of our method, we conduct ablation studies by disabling one component at a time. Specifically, **HIEM-proxy** sets $\alpha = 0$ to disable the proxy low-level network θ_l^i, **HIEM-low** sets $\alpha = 1$ to disable the low-level network θ_l^e, and **HIEM-term** sets $term(s, g, sg) = 0$ to disable the termination network θ_t.

For fair comparisons, all the methods share similar network architectures and hyperparameters, and they all take the predicted semantic segmentation and the depth map as the inputs.



DQN (203 steps)

H-DQN (146 steps)

HIEM-low (81 steps)

HIEM (44 steps)

Fig. 3. Trajectories generated by DQN [12], H-DQN [19] and our method HIEM-low and HIEM for searching the target object *music player* (red dots) from the same starting position (green triangle) which is 39 steps away. Different colors represent different sub-goals in which the colored lines and dots denote the corresponding sub-goal-oriented trajectories and sub-goal states respectively. Our method HIEM generates a more concise and interpretable trajectory. We refer readers to the supplemental video demo for animated demonstrations.

To be specific, for DQN networks in the method DQN, H-DQN and HIEM, we adopt the Double DQN [42] technique where we train the main network every 100 time steps with a batch of size 64 and we update the target network every 100 000 time steps. The exploration rate decreases from 1 to 0.1 over 10 000 time steps. For the A3C network, we set the weight of the entropy regularization term as 0.01 and we update the network for every 10 time steps unrolled. We adopt RMSProp optimizer of learning rate 1×10^{-4} to train each method to search 6 different target objects (78 in total) from random starting positions in the environment. During testing time, we randomly sample 100 starting positions and the corresponding target objects. We set the maximum number of atomic actions that all methods can take as 500, and for the method H-DQN and HIEM, the maximum number of atomic actions that the low-level layer can take at each time is 25. The robot stops either when it reaches the goal state (success case) or when it runs out of 500 atomic action steps (failure case). We implement all the methods using Tensorflow toolbox and conduct all the experiments with Nvidia V100 GPUs and 16 Intel Xeon E5-2680 v4 CPU cores. In general, each training takes around 2 days.

C. Experimental Results and Discussion

Since we formulate the object search problem as maximizing the discounted cumulative extrinsic rewards, we take the Average discounted cumulative extrinsic Rewards (AR) as one of the evaluation metrics, calculated by:

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{\infty}\gamma^{t} r^{e}_{t+1} = \frac{1}{N}\sum_{i=1}^{N} 1(success)\gamma^{\#steps} * 1, \quad (7)$$

where $\gamma \in (0,1]$ is the discount factor. From the perspective of the evaluation metric, it can also be seen as a trade-off between the success rate metric and the average steps metric. With the higher value of γ, the average steps metric weighs less, and vice versa. In our experiments, we set $\gamma = 0.99$.

In addition, we also report the following widely used evaluation metrics. Success Rate (SR). Average Steps over all successful cases compared to the Minimal Steps over these cases (AS / MS). Success weighted by inverse Path Length (SPL) [43], which is calculated as $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(l_i,p_i)}$. Here, S_i is the binary indicator of success in episode i, l_i and p_i are the lengths of the shortest path and the path actually taken by the robot. We

TABLE I
THE PERFORMANCE OF ALL METHODS FOR THE OBJECT SEARCH TASK. (SR: SUCCESS RATE; AS / MS: AVERAGE STEPS / MINIMAL STEPS OVER ALL SUCCESSFUL CASES; SPL: SUCCESS WEIGHTED BY INVERSE PATH LENGTH; AR: AVERAGE DISCOUNTED CUMULATIVE EXTRINSIC REWARDS.)

Method	SR↑	AS / MS↓	SPL↑	AR↑
ORACLE	1.00	25.63 / 25.63	1.00	0.79
RANDOM	0.19	188.11 / 7.05	0.03	0.08
A3C	0.13	93.23 / 4.00	0.03	0.08
DQN	0.47	120.74 / 16.09	0.20	0.26
OC	0.14	99.29 / 5.14	0.06	0.09
H-DQN	0.74	182.15 / 23.62	0.17	0.23
Ours				
HIEM-proxy	0.40	95.08 / 15.03	0.12	0.22
HIEM-low	0.99	76.81 / 25.55	0.47	0.56
HIEM-term	**1.00**	49.42 / 25.63	0.65	0.66
HIEM	**1.00**	**41.18 / 25.63**	**0.72**	**0.70**

TABLE II
AVERAGE SPL ACHIEVED BY ALL METHODS ON 4 RANDOM ENVIRONMENTS

Method	RANDOM	A3C	DQN	OC	H-DQN	**HIEM**
Avg SPL	0.03	0.03	0.35	0.03	0.11	**0.54**

adopt the number of the action steps as the path length. As a result, SPL also trades-off success rate against average steps.

Table I shows comparisons of all the methods in performing the object search task. It demonstrates the superiority of our method over all metrics, and also highlights the following observations.

The intrinsic rewards help to explore. Comparing to H-DQN and our methods (HIEM, HIEM-low, HIEM-term) which model the object search task with both extrinsic and intrinsic rewards, all the other methods where no intrinsic rewards is involved achieve unsatisfactory success rate. It indicates that under the sparse extrinsic rewards setting, the robot struggles to reach the goal state even with the hierarchical policy OC or HIEM-proxy, while our intrinsic rewards effectively encourage the robot to explore the environment and encounter the goal state. In fact, the intrinsic rewards guide our proxy low-level network to approach a visible object, and only after the proxy low-level network achieves good performance can it collaborate with our high-level network to help explore.

Our intrinsic-extrinsic modeling contributes to a more optimal policy. Though our intrinsic rewards help to explore the environment and improve the success rate, they are limited in improving the policy in terms of the optimality, as suggested by the higher AS and lower SPL and AR that H-DQN and HIEM-low achieve in comparison with HIEM. Different from H-DQN or HIEM-low that models the low-level layer with the intrinsic rewards solely, our HIEM adopts the novel intrinsic-extrinsic modeling and yields a more optimal policy, demonstrating the role of our intrinsic-extrinsic modeling in learning an optimal policy.

Early termination to the non-optimal low-level policy is necessary. A non-optimal low-level policy would drive the robot to an undesirable state that in consequence hurts the object search performance. The issue is shown to be mitigated by terminating the low-level policy at a valuable state in HIEM-low and HIEM when comparing them with H-DQN and HIEM-term respectively. Furthermore, we also observe that the termination function helps more to less optimal low-level policy as more improvements are achieved from H-DQN to HIEM-low.

We also report in Table II the average SPL achieved by all methods on 4 random environments. It further validates the superiority of our HIEM on other environments as well. We depict sample qualitative results in Fig. 3, which shows that our method yields a more concise and interpretable trajectory compare to other methods for the object search task.

V. CONCLUSION AND FUTURE WORK

In this paper, we present a novel two-layer hierarchical policy learning framework for the robotic object search task. The hierarchical policy builds on a simple yet effective and interpretable low dimensional sub-goal space, and is learned with both extrinsic and intrinsic rewards to perform the object search task in a more optimal and interpretable way. When our high-level layer plans over the specified sub-goal space, the low-level layer plans over the atomic actions to collaborate with the high-level layer to better achieve the goal. This is efficiently learned with the experience samples collected by our proxy low-level policy, a policy optimizes towards the proposed sub-goals. Moreover, our low-level layer terminates at valuable states which further approximates the optimal policy. The empirical and extensive experiments together with the ablation studies on House3D platform demonstrate the efficacy and efficiency of our presented framework. The presented HIEM framework further paves several possible avenues for future study. A promising one is by incorporating the Goals Relational Graph (GRG) [44] to integrate top-down human knowledge together with the human specified sub-goal space to facilitate the object search with improved efficiency.

We want to mention that the current work assumes the robot can access the environment for training before being deployed in the same one for object search. In other words, we do not aim for the generalization ability towards novel environments, but our success sheds light on how to generalize well. Specifically, an optimal object search policy in an environment is determined by the map of the environment. In order to generalize a learned object search policy to a new environment where the map is unknown and no extra exploration or training process is allowed, the robot must be able to infer the map from its observation and/or from its external memory or knowledge. While the large high-resolution map is extremely challenging to infer, inferring a small part of it and a low-resolution object arrangement are still tractable, which in consequence makes both of our low-level policy and high-level policy more likely to generalize well. We deem it as our future work.

REFERENCES

[1] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 2054–2063.

[2] N. Kojima and J. Deng, "To learn or not to learn: Analyzing the role of learning for navigation in virtual environments," 2019, *arXiv:1907.11770*.

[3] D. Mishkin, A. Dosovitskiy, and V. Koltun, "Benchmarking classic and learned navigation in complex 3d environments," 2019, *arXiv:1901.10915*.

[4] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," 2017, *arXiv:1708.05866*.

[5] P. Mirowski *et al.*, "Learning to navigate in complex environments," in *Proc. Int. Conf. on Learn. Representations*, 2017.

[6] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3357–3364.

[7] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3389–3396.

[8] I. Popov *et al.*, "Data-efficient deep reinforcement learning for dexterous manipulation," 2017, *arXiv:1704.03073*.

[9] A. Rajeswaran *et al.*, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Proc. of Robotics: Science and Systems*, 2018.

[10] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1343–1350.

[11] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3052–3059.

[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn.*. ACM, 2004.

[14] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 8846–8852.

[15] X. Wang, W. Xiong, H. Wang, and W. Yang Wang, "Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 37–53.

[16] X. Wang *et al.*, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6629–6638.

[17] X. Ye, Z. Lin, H. Li, S. Zheng, and Y. Yang, "Active object perceiver: Recognition-guided policy learning for object searching on mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 6857–6863.

[18] O. Nachum, H. Tang, X. Lu, S. Gu, H. Lee, and S. Levine, "Why does hierarchy (sometimes) work so well in reinforcement learning?" 2019, *arXiv:1909.10618*.

[19] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.

[20] H. M. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé III, "Hierarchical imitation and reinforcement learning," in *Proc. Int. Conf. on Mach. Learn.*, 2018, pp. 2917–2926.

[21] A. Levy, R. Platt, and K. Saenko, "Hierarchical reinforcement learning with hindsight," 2018, *arXiv:1805.08180*.

[22] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3307–3317.

[23] O. Nachum, S. Gu, H. Lee, and S. Levine, "Near-optimal representation learning for hierarchical reinforcement learning," in *Proc. Int. Conf. on Learn. Representations*, 2019.

[24] Z. Dwiel, M. Candadai, M. J. Phielipp, and A. K. Bansal, "Hierarchical policy learning is sensitive to goal space design," 2019, *arXiv:1905.01537*.

[25] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1726–1734.

[26] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3 d environment," 2018, *arXiv:1801.02209*.

[27] A. Levy, R. Platt, and K. Saenko, "Hierarchical actor-critic," 2017, *arXiv:1712.00948*.

[28] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *Proc. 34th Int. Conf. Mach. Learn.-Volume 70*. JMLR. org, 2017, pp. 166–175.

[29] S. Sohn, J. Oh, and H. Lee, "Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7156–7166.

[30] A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Neural modular control for embodied question answering," in *Proc. Conf. on Robot Learn.*, 2018, pp. 53–62.

[31] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: Visual question answering in interactive environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4089–4098.

[32] T. Osa, V. Tangkaratt, and M. Sugiyama, "Hierarchical reinforcement learning via advantage-weighted information maximization," in Proc. *Int. Conf. on Learn. Representations*, 2019.

[33] X. Ye and Y. Yang, "From seeing to moving: A survey on learning for visual indoor navigation (vin)," 2020, *arXiv:2002.11310*.

[34] J. Kulhánek, E. Derner, T. de Bruin, and R. Babuška, "Vision-based navigation using deep reinforcement learning," in *Proc. Eur. Conf. Mobile Robots.*, 2019, pp. 1–8.

[35] X. Ye, Z. Lin, J.-Y. Lee, J. Zhang, S. Zheng, and Y. Yang, "Gaple: Generalizable approaching policy learning for robotic object searching in indoor environment," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4003–4010, Oct. 2019.

[36] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," in *Proc. Int. Conf. on Learn. Representations*, 2019.

[37] R. Druon, Y. Yoshiyasu, A. Kanezaki, and A. Watt, "Visual object search by learning spatial context," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1279–1286, Apr. 2020.

[38] Y. Wu, Y. Wu, A. Tamar, S. Russell, G. Gkioxari, and Y. Tian, "Bayesian relational memory for semantic visual navigation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 2769–2779.

[39] P. Anderson *et al.*, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3674–3683.

[40] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[41] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[42] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp.2094–2100.

[43] P. Anderson *et al.*, "On evaluation of embodied navigation agents," 2018, *arXiv:1807.06757*.

[44] X. Ye and Y. Yang, "Hierarchical and partially observable goal-driven policy learning with goals relational graph," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021.