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.

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 in complex real-world scenarios [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 under the real-world scenarios [13]. Typically, for real-world 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 the real-
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].

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, and report the observed results that validate 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 introduced an off-policy correction method. [27] and [21] proposed 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] defined the sub-goal space in the space of entities and relations, such as the “reach” relation they used for their Atari game experiment. Sub-tasks and their relations were provided as inputs in [28] and [29]. Closer related to our work, [30] adopted \{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] chose \{navigate, scan, detect, manipulate, answer\} as the possible sub-tasks, while the reinforcement learning methods were mainly applied for learning high-level policy, i.e., planning over the pre-trained or fixed sub-tasks. More recently, [32] integrated symbolic planning as an additional sub-task.

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] proposed the option-critic framework to autonomously discover the specified number of temporal abstractions. [33] learned the temporal abstractions through advantage-weighted information maximization. [23] addressed 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 expressive 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 task [34]. [5] proposed learning depth prediction and action policy jointly for visual navigation. [6] introduced a target-driven deep reinforcement learning framework to enable a robot to reach an image-specified location, which has also been studied in [35]. The Embodied Question Answering or Interactive Question Answering task is introduced and addressed in [1], [30]–[32], [36], [37], where the robot needs to navigate an environment to answer an unconstrained natural language question. [14], [17], [38]–[41] focused on the robotic object search and/or object approaching task. [42], [15] and [16] focused on natural language guided visual navigation. Among all of these cited work, most of them plan over the atomic actions for navigation [1], [5], [6], [14], [16], [17], [35], [37]–[42] without hierarchical modeling, and thus lack interpretability. While [31] and [32] studied hierarchical policies, they manually constructed the hierarchy with high-level and low-level controllers, and a standard reinforcement learning method is applied on high-level layer to decide which low-level controller to invoke. [30] also studied...
hierarchical reinforcement learning to generate explainable hierarchical policies. Along with these mentioned methods that build upon human specified sub-goal space, we further consider the likelihood that these manual sub-goals may not lead to the optimal steps towards the final goal, and we model it explicitly in our framework.

Notably, many of the previous works assume that the robot can access the full information of the environments during the training time, either by defining the reward function with the distance between the robot’s current location and the target location [14]–[16], and/or adopting shortest path as the supervised signal for pre-training [1], [16], [30], [31], [36], [42]. 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 to excel.

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 \( \pi_h \) and a low-level policy \( \pi_l \). 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 \( sg_t \sim \pi_h(s_t, g) \) 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 \( \text{term}(s_{t+N_t}, g, sg_t) \); 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 \{\text{approach obj}|\text{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 current view is a more general and relatively trainable task shown by [39]. 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 \{\text{move forward} / \text{backward} / \text{left} / \text{right}, \text{turn left} / \text{right}\}.

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.
Following the option-critic framework [25], we unroll the state action value $Q_h$ [43] for our high-level layer, i.e., $E[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, g = g, s_0 = sg] = Q_h^g(s, g, sg)$.

Following the option-critic framework [25], we unroll the state action value $Q_h$ as,

\[
Q_h^g(s, g, sg) = \sum_a \pi(a|s, g, sg) E[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, g = g, s_0 = sg, a_0 = a]
\]

where the state action value $Q_l^g(s, g, sg, a)$ 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 $s$, $Q_l^g(s, g, sg, a)$ can be further formulated as,

\[
Q_l^g(s, g, sg, a) = \sum_{s'} P(s' | s, a)[r(s, a, s', g) + \gamma U(g, sg, s')] ,
\]

\[
U(g, sg, s') = (1 - \text{term}(s', g, sg))Q_h^g(s', g, sg) + \text{term}(s', g, sg)V_l^g(s', g),
\]

\[
V_l^g(s', g) = \sum_{sg'} \pi_l(g'|s', g)V_h^g(s', g, sg') .
\]

We parameterize $Q_h^g(s, g, sg)$, $Q_l^g(s, g, sg, a)$ and term$(s, g, sg)$ with $\theta_h^g$, $\theta_l^g$ and $\theta_t$ respectively. Then the high-level policy $\pi_h(g|s, g) = 1(s = \text{argmax}_a Q_h^g(s, g, sg))$, and the low-level policy $\pi_l(a|s, g, sg) = 1(a = \text{argmax}_a Q_l^g(s, g, sg, a))$. We adopt the DQN [12] based method to learn $Q_h^g(s, g, sg)$ and $Q_l^g(s, g, sg, a)$ in which we update both of the values towards the 1-step extrinsic return $R_l^i = r(s, a, s', g) + \gamma U(g, sg, s')$, and consequently $\theta_h^g$ and $\theta_l^g$ can be updated by Equation 3 and 4. In addition, $\theta_t$ can be updated by Equation 5 as demonstrated by [25].

\[
\theta_h^g \leftarrow \theta_h^g - \nabla_{\theta_h^g} [R_l^i - Q_h^g(s, g, sg)]^2 .
\]

\[
\theta_l^g \leftarrow \theta_l^g - \nabla_{\theta_l^g} [R_l^i - Q_l^g(s, g, sg, a)]^2 .
\]

\[
\theta_t \leftarrow \theta_t - \nabla_{\theta_t} [R_l^i - \text{term}(sg, sg)Q_h^g(s', g, sg) - V_l^g(s', g)].
\]

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 rewards through the ε-greedy [12] exploration policy and collect the experience samples to effectively train $\theta_h^g$, $\theta_l^g$ and $\theta_t$. On the contrary, encountering the intrinsic rewards is much more possible 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 $\theta_h^g$, $\theta_l^g$ and $\theta_t$.

We hereby define a proxy low-level policy $\pi_l^a(s, g)$ aiming to achieve the proposed sub-goal $sg$. Similarly, we learn the proxy low-level policy by optimizing the discounted cumulative intrinsic rewards:

\[
Q_l^a(s, g, sg, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, g_0 = sg, a_0 = a].
\]

We unroll $Q_l^a(s, g, sg, a)$ following Equation 7 and we adopt the DQN method [12] to learn it by updating its parameter $\theta_l^a$ with Equation 8 where $R_l^i = r^l(s, a, s', sg) + \gamma V_l^a(s', sg)$ is the 1-step intrinsic return and $\pi_l^a(a|s, g) = 1(a = \text{argmax}_a Q_l^a(s, g, sg, a))$.

\[
Q_l^a(s, g, sg, a) = \sum_{s'} P(s' | s, a)[r^l(s, a, s', sg) + \gamma V_l^a(s', sg)]
\]

\[
V_l^a(s', sg) = \sum_a \pi_l^a(a|s', sg)Q_l^a(s', sg, a).
\]

\[
\theta_l^a \leftarrow \theta_l^a - \nabla_{\theta_l^a} [R_l^i - Q_l^a(s, g, sg, a)]^2 .
\]

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 $s_g$ with the proxy policy $\pi^p(a|s, s_g)$, we introduce a hyper-parameter $\alpha \in [0, 1]$ as the probability that the low-level layer adopts the proxy policy $\pi^p(a|s, s_g)$ to explore the environment and collect the experience samples. The experience samples are used to batch train $\theta^e_l$, $\theta^f_l$, $\theta_l$ and $\theta^t_l$ with Equation 3, 4, 5 and 6 respectively. In practice, $\alpha$ decays from 1 to 0 across the training episodes to enable our low-level layer to act optimally towards the goal with the policy $\pi_l(a|s, g, s_g)$ 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 $\theta^e_h$, $\theta^f_h$, $\theta_i$ and $\theta^t_i$ to handle the high-dimensional inputs and approximate $Q^h_l(s, g, s_g)$, $Q^t_l(s, g, s_g, a)$, $\text{term}(s, g, s_g)$ and $Q^i_l(s, g, s_g, a)$.

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 [44] 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 $\theta^e_h$ 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 $1 \times 1$, 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 $\theta^e_h$ feeds the embedding fusion into one additional fully connected layer to approximate $Q^h_l(s, g, s_g)$. To save the number of parameters, our termination network $\theta_t$ shares most parameters with the high-level network $\theta^e_h$ except the last fully connected layer where it adopts a new one to approximate $\text{term}(s, g, s_g)$.

For the low-level network $\theta^f_l$ and $\theta^t_l$, we take the sub-goal specified channel of the predicted semantic segmentation and the predicted depth map as the inputs. The low-level network $\theta^f_l$ takes the one-hot vector of the target object as an additional input. Similar to our high-level network, each input of $\theta^f_l$ and $\theta^t_l$ is fed into a fully connected layer before being concatenated together to generate an embedding fusion with a new fully connected layer. The embedding fusion is further fed into an additional fully connected layer to approximate $Q^t_l(s, g, s_g, a)$ and $Q^i_l(s, g, s_g, a)$.

We follow Equation 3, 4, 5 and 6 to learn $Q^h_l(s, g, s_g)$, $Q^t_l(s, g, s_g, a)$, $\text{term}(s, g, s_g)$ and $Q^i_l(s, g, s_g, a)$ 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 [44] (shown in Fig. 2 [upper left] for semantic segmentation and depth prediction as we mentioned in Sec. III-D). 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 each 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** [45]. The vanilla A3C implementation that has been widely adopted for the navigation task in the previous work [6], [17], [39]–[41]. 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 $\theta^e_h$.

- **DQN** [12]. The vanilla DQN implementation that adopts a similar network architecture as our high-level network $\theta^e_h$ 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 $\text{term}(s, g, s_g) = 0$ and $\alpha = 1$ to disable both the termination network $\theta_t$ and the low-level network $\theta^t_l$.

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 $\theta^t_l$, **HIEM-low** sets $\alpha = 1$ to disable the low-level network $\theta^t_l$, and **HIEM-term** sets $\text{term}(s, g, s_g) = 0$ to disable the termination network $\theta_t$. 
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 from [video link] for animated demonstrations.

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. We 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 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 it reaches the goal state (success case) or 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.

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:

\[
\frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\infty} \gamma^t r_{t+1} = \frac{1}{N} \sum_{i=1}^{N} 1(\text{success})\gamma^{\#\text{steps}} * 1, \quad (9)
\]

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 \( \gamma \), 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) [46], which is calculated as \( \frac{1}{N} \sum_{i=1}^{N} S_i \frac{l_i}{\max(l_i, p_i)} \).
A non-optimal low-level policy would drive extrinsic modeling in learning an optimal policy. Though our 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.

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 further 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.

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 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 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,” arXiv preprint arXiv:1907.11770, 2019.
[3] D. Miskin, A. Dosovitskiy, and V. Kolm, “Benchmarking classic and learned navigation in complex 3d environments,” arXiv preprint arXiv:1901.10915, 2019.
[4] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A brief survey of deep reinforcement learning,” arXiv preprint arXiv:1708.03866, 2017.
