

Vision-guided Policy Learning for Complex Tasks

by

Xin Ye

A Dissertation Presented in Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

Approved July 2021 by the  
Graduate Supervisory Committee:

Yezhou Yang, Chair  
Yi Ren  
Theodore Pavlic  
Deliang Fan  
Siddharth Srivastava

ARIZONA STATE UNIVERSITY

August 2021

## ABSTRACT

The field of computer vision has achieved tremendous progress over recent years with innovations in deep learning and neural networks. The advances have unprecedentedly enabled an intelligent agent to understand the world from its visual observations, such as recognizing an object, detecting the object’s position, and estimating the distance to the object. It then comes to a question of how such visual understanding can be used to support the agent’s decisions over its actions to perform a task. This dissertation aims to study this question in which several methods are presented to address the challenges in learning a desirable action policy from the agent’s visual inputs for the agent to perform a task well. Specifically, this dissertation starts with learning an action policy from high dimensional visual observations by improving the sample efficiency. The improved sample efficiency is achieved through a denser reward function defined upon the visual understanding of the task, and an efficient exploration strategy equipped with a hierarchical policy. It further studies the generalizable action policy learning problem. The generalizability is achieved for both a fully observable task with local environment dynamic captured by visual representations, and a partially observable task with global environment dynamic captured by a novel graph representation. Finally, this dissertation explores learning from human-provided priors, such as natural language instructions and demonstration videos for better generalization ability.

*To my loving parents,  
for their constant support and encouragement.*

## ACKNOWLEDGEMENTS

I would like to take this chance to thank my advisor Dr. Yezhou Yang, who guides my research with great patience throughout my doctoral study. I still remember that I even needed help to draw an arrow on an image five years ago, and I couldn't imagine how this dissertation would be possible without his continuous encouragement and feedback. I am fortunate to work with him at Active Perception Group, where I have the freedom to explore the research problems that are of my interest, and meanwhile, I have been trained to improve my research, writing, and presentation skills greatly. I also appreciate all the internship and collaboration opportunities he provided to me.

I would also like to thank my committee members, Dr. Yi Ren, Dr. Theodore Pavlic, Dr. Deliang Fan, and Dr. Siddharth Srivastava for their service on my committee and the valuable suggestions provided. Their insightful questions and comments on my comprehensive exam and proposal defense have inspired me to think more deeply and critically about my ongoing and future research work. I also thank all my collaborators, Dr. Wenlong Zhang, Dr. Yiwei Wang, Dr. Zhe Lin, Dr. Haoxiang Li, Dr. Joohyung Lee, and many others for their expertise to improve my work.

I am grateful to have my friends Rui Zhang, Ze Gong, Ying Qin, Rui Shi, Zhaosong Huang to share happiness during the stressful doctoral study. I also thank all my lab mates, especially Mohammad Farhadi, Zhiyuan Fang, Shibin Zheng for their help in my work and life, and I want to give my special thanks to Zhiyuan Fang for a lovely cat he gave me that accompanies me through the difficult quarantine time.

Finally, thanks to my parents for their continuous care and love and thanks to myself for never giving up.

## TABLE OF CONTENTS

	Page
LIST OF TABLES .....	vii
LIST OF FIGURES .....	ix
CHAPTER	
1 INTRODUCTION .....	1
1.1 Problem Statement .....	1
1.2 Related Work .....	5
1.3 Contributions .....	7
1.4 Dissertation Outline .....	9
2 REWARD FUNCTIONS FROM VISUAL UNDERSTANDING .....	11
2.1 Introduction .....	11
2.2 Related Work .....	12
2.3 Our Approach .....	13
2.3.1 Object Recognition Module with Target Object Given .....	14
2.3.2 Recognition-guided Action Policy Learning .....	16
2.4 Experiments .....	18
2.4.1 Dataset .....	19
2.4.2 Experimental Results and Discussion .....	21
2.5 Conclusion .....	24
3 EFFICIENT EXPLORATION WITH HIERARCHICAL POLICY .....	26
3.1 Introduction .....	26
3.2 Related Work .....	28
3.3 Our Approach .....	30
3.3.1 Hierarchy of Policies .....	31
3.3.2 Extrinsic and Intrinsic Rewards .....	32

CHAPTER	Page
3.3.3	Model Formulation . . . . . 32
3.3.4	Network Architecture . . . . . 35
3.4	Experiments . . . . . 36
3.4.1	Dataset . . . . . 36
3.4.2	Experimental Setting . . . . . 37
3.4.3	Experimental Results and Discussion . . . . . 39
3.5	Conclusion . . . . . 42
4	GENERALIZABLE POLICY LEARNING FOR FULLY OBSERVABLE TASK . . . . . 45
4.1	Introduction . . . . . 45
4.2	Related Work . . . . . 47
4.3	Our Approach . . . . . 49
4.3.1	Overview . . . . . 49
4.3.2	Semantic Segmentation and Depth Prediction . . . . . 51
4.3.3	Approaching Policy Learning . . . . . 52
4.4	Experiments . . . . . 55
4.4.1	Dataset . . . . . 55
4.4.2	Semantic Segmentation and Depth Prediction . . . . . 56
4.4.3	Approaching Policy Learning . . . . . 57
4.4.4	Real World Experiment . . . . . 63
4.4.5	Analysis and Discussion . . . . . 64
4.5	Conclusion . . . . . 66
5	GENERALIZABLE POLICY LEARNING FOR PARTIALLY OBSERV- ABLE TASK . . . . . 68

CHAPTER	Page
5.1	Introduction..... 68
5.2	Related Work ..... 71
5.3	Hierarchical RL with GRG ..... 73
5.3.1	Overview ..... 73
5.3.2	Goals Relational Graph (GRG) ..... 74
5.3.3	Goal-driven High-level Network ..... 76
5.3.4	Termination-aware Low-level Network ..... 78
5.4	Experiments..... 80
5.4.1	Grid-world Domain ..... 80
5.4.2	Robotic Object Search ..... 87
5.5	Conclusion ..... 92
6	TOWARDS LEARNING FROM HUMAN-PROVIDED PRIORS FOR GENERALIZABILITY ..... 95
6.1	Environment Dynamics from Natural Language Instructions ..... 95
6.2	Environment Dynamics from Human Demonstrations..... 100
7	CONCLUSION ..... 105
	REFERENCES ..... 108

## LIST OF TABLES

Table		Page
2.1	The Performance Metrics of All the Methods in the Simulation Platform. (AL: Average Trajectory Length; SR: Success Rate) .....	22
2.2	The Performance Metrics for the Real World Experiments. (AL: Average Trajectory Length; SR: Success Rate) .....	23
3.1	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.) .....	40
3.2	Average SPL Achieved by All Methods on 4 Random Environments....	42
4.1	The Success Rate Drop from the Trained Objects to New Objects (Setting 1): S1), and from the Trained Environments to New Environments (Setting 2): S2). .....	61
4.2	Average Number of Steps Taken by All Methods on Two Settings.....	62
4.3	Success Rate of Method b) Where the Target Object Channel Is Blocked (with Predicted Inputs and Within 5X Minimal Steps). .....	62
4.4	Experimental Results of Our Method (Method e)) on AVD (Ammirato <i>et al.</i> , 2017). .....	63
5.1	The Performance of DQN vs DQN_ONEHOT and DQN_FULL on the Unseen Grid-world Maps. ....	81
5.2	Hyperparameters of all the methods for the grid-world domain.....	83
5.3	The Performance of All Methods on the Unseen Grid-world Maps.....	83
5.4	The Ablation Studies of Our Method on the Unseen Grid-world Maps..	85



Table	Page
5.5 The Performance Improvement of SCENE PRIORS (Yang <i>et al.</i> , 2018) (Top) and Our HRL-GRG (Bottom) over the RANDOM Method in the AI2-THOR (Kolve <i>et al.</i> , 2017b) Environment for the Robotic Object Search Task (Without Stop Action).....	88
5.6 The Performance of All Methods in the House3D (Wu <i>et al.</i> , 2018) Environment for the Robotic Object Search Task. ....	89
5.7 Hyperparameters of all the methods for the robotic object search task..	89

## LIST OF FIGURES

Figure	Page
1.1 An Illustration of Our Policy Learning Problem Focused on This Dissertation. ....	2
1.2 An Illustration of a Visual Navigation Task. ....	3
1.3 The Outline of the Dissertation. ....	10
2.1 An Illustration of Our Robotic Object Search Task. Red Dot: The Random Initial Location; White Line and Error: The Generated Robot Trajectory from Our Turtle-bot Experiment; Upper Left Image: The Final View of the Robot with the Target Object Detection (the Target Object in This Experiment Is “shear”). ....	13
2.2 The Architecture of the Object Recognition Network. ....	15
2.3 An Illustration of Our Vision-guided Reward Function. ....	18
2.4 The Deep Reinforcement Learning Architecture. ....	19
2.5 Sample Testing and Training Images From our Real World Dataset. . .	20
3.1 An Example of Our <i>HIEM</i> Framework. When Our High-level Policy Proposes a Sub-goal, Our Proxy Low-level Policy Is Invoked with the Probability of $\alpha$ 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 $1 - \alpha$ to Collaborate with the High-level Policy to Better Achieve the Goal. ....	29
3.2 Network Architecture of Our Hierarchical Reinforcement Learning Model.	35

Figure	Page
3.3 Trajectories Generated by DQN (Mnih <i>et al.</i> , 2015), H-DQN (Kulkarni <i>et al.</i> , 2016) and Our Method HIEM-low and HIEM for Searching the Target Object <i>Music Player</i> (Red Dots) from the Same Starting Position (Green Triangle) Which Is 39 Steps Away. Our Method HIEM Generates a More Concise and Interpretable Trajectory. . . . .	43
4.1 An Overview of Our GAPLE System. . . . .	49
4.2 An Illustration of the Adopted Model Based on <i>DeepLabv3+</i> (Chen <i>et al.</i> , 2018) to Predict Semantic Segmentation and Depth Map from a Single RGB Image. . . . .	51
4.3 The Architecture of Our Deep Reinforcement Learning Model for Action Policy Training. . . . .	55
4.4 A Sample Environment from House3D and Some Target Object Candidates. . . . .	56
4.5 Some Qualitative Semantic Segmentations and Depth Predictions from Our Feature Representation Module. . . . .	57
4.6 Successful Approaching Rates. Upper: Setting 1: Generalization Ability Across Target Objects (on Trained Objects and on New Objects). Lower: Setting 2: Generalization Ability Across Environments (on Trained Environments and on New Environments). . . . .	60
4.7 An Example of the Mobile Robot Approaches the Target Object “white-board” Using the Method (d)). Upper View: RGB Input; Lower View: Depth Map Generated. . . . .	64
4.8 Pair-wise Feature Distances W.R.T. Physical Distances. . . . .	66

Figure	Page
5.1 Illustrations of the Grid-world Domain and the Robotic Object Search Task (Left), and an Overview of Our Method (Right). . . . .	69
5.2 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. . . . .	78
5.3 A Visualization of a GRG Learned on the Grid-world Domain ( $g_{16}$ Is the Back-up “ <i>Random</i> ” Goal). . . . .	84
5.4 Trajectories Generated by Our Method on the Unseen Grid-world Maps for Both the Seen Goals (a) (b) and the Unseen Goals (c) (d). The Different Colors Represent Different Sub-goals and the Corresponding Sub-goal-oriented Trajectories Where the Red One Denotes the Designated Final Goal. . . . .	86
5.5 Trajectories Generated by Our Method for the Robotic Object Search Task on AI2-THOR (Kolve <i>et al.</i> , 2017b). . . . .	90
5.6 The Object Relations Captured by Our GRG in the House3D (Wu <i>et al.</i> , 2018) Environment for the Robotic Object Search Task. Only a Small Number of Objects as Nodes and the Edges with the Weight $\geq .5$ Are Shown. . . . .	91
5.7 Trajectories Generated by Our Method for the Robotic Object Search Task on House3D (Wu <i>et al.</i> , 2018). . . . .	93
6.1 An Illustration of the Baseline VLN Model EnvDrop (Tan <i>et al.</i> , 2019). . . . .	98
6.2 An Illustration of Our Proposed VLN Model. . . . .	99

6.3	Action Representation. Top Two Rows Are Two Examples of Primitive Actions. The Upper One Is from the 50 Salads (Stein and McKenna, 2013) Dataset and the Lower One Is from the MANIAC Dataset (Aksoy <i>et al.</i> , 2017). Bottom Row Is an Abstract Representation of One Example Primitive Action. ....	102
6.4	The GRG Learned on 50 Salads Dataset (Stein and McKenna, 2013)...	103

## INTRODUCTION

**1.1 Problem Statement**

Recent work equipped with deep learning technique has made tremendous progress on computer vision tasks, such as object recognition (Krizhevsky *et al.*, 2017), semantic segmentation (Long *et al.*, 2015) and depth estimation (Liu *et al.*, 2015). These advancements have unprecedentedly enabled an intelligent agent to understand the world from its visual observations. However, how such visual understanding could be used to support the agent’s decisions over its actions to perform a task has not been efficiently explored. On the other hand, reinforcement learning has long been demonstrated its power at enabling agents with autonomous behaviors (Arulkumaran *et al.*, 2017), such as navigating over an unknown environment (Mirowski *et al.*, 2016; Zhu *et al.*, 2017), manipulating objects with robot’s end effectors (Gu *et al.*, 2017; Popov *et al.*, 2017; Rajeswaran *et al.*, 2017), and motion planning (Chen *et al.*, 2017b; Everett *et al.*, 2018). Equipped with deep neural networks, deep reinforcement learning algorithms are able to directly take high dimensional sensory inputs as states to learn action policies. While it is straightforward to take the agent’s visual observations as the states, learning an effective action policy from them that is also interpretable, generalizable and robust still remains a challenge. Here, in this dissertation, we study the problem of learning action policies from the agent’s visual observations and we approach it by understanding the visual observations in order to let the agent to perform the task well.

Formally speaking, a task can be defined as a 10-tuple  $\langle S, A, T, G, R, \Omega, O, G_d,$

$\Phi, \gamma >$ , in which  $S$  is a set of valid states,  $A$  is a set of actions that can be performed by the agent,  $T : S \times A \times S \rightarrow [0, 1]$  is a state transition probability function describing an environment dynamic,  $G \subseteq S$  is a set of goal states indicating the task is completed,  $R : S \times G \rightarrow \mathbb{R}$  is a reward function that gives feedback of how well the task is performed. For complex tasks, the true state  $s \in S$  may not always be accessible, thus requiring the agent to estimate it from the available observation  $o \in \Omega$ .  $\Omega$  is a set of observations that are determined by conditional observation probability  $O : S \times \Omega \rightarrow [0, 1]$ . Similarly, the goal state  $g \in G$  may not be available either and therefore a goal description  $g_d \in G_d$  is instead used to convey the task goal. The goal recognition function  $\Phi : \Omega \times G_d \rightarrow [0, 1]$  defines how likely a goal description  $g_d$  is satisfied by an observation  $o$ . 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  $\Delta$ .  $\gamma \in (0, 1]$  is a discount factor. To perform the task well, the agent needs to learn an optimal action policy  $\pi : \Omega \times G_d \rightarrow A$  to select an action  $a_t$  at the state  $s_t$  given the observation  $o_t$  and the goal description  $g_d$  so that the expected discounted cumulative rewards  $\mathbb{E}[\sum_t^{\infty} \gamma^t r_{t+1}(s_t, a_t, s_{t+1}, g) | s_t, g]$  is maximized. An illustration is shown in Figure 1.1.

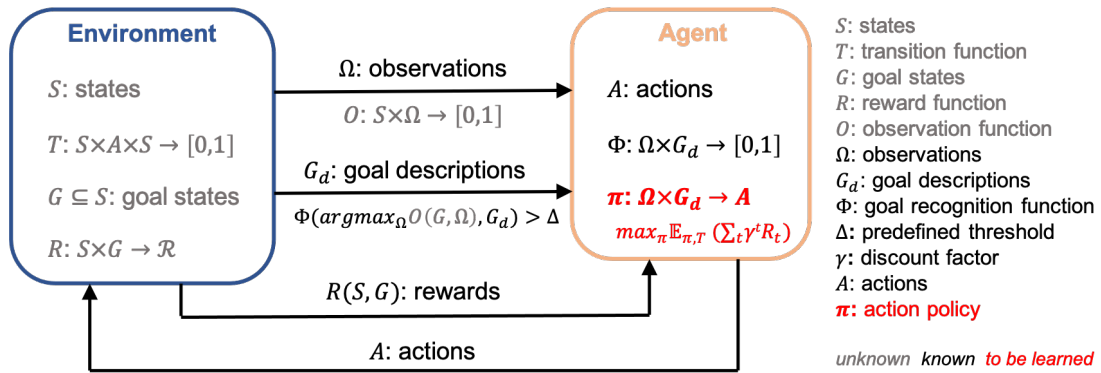


Figure 1.1: An illustration of our policy learning problem focused on this dissertation.

Taking visual navigation as an example (see Figure 1.2), where an intelligent

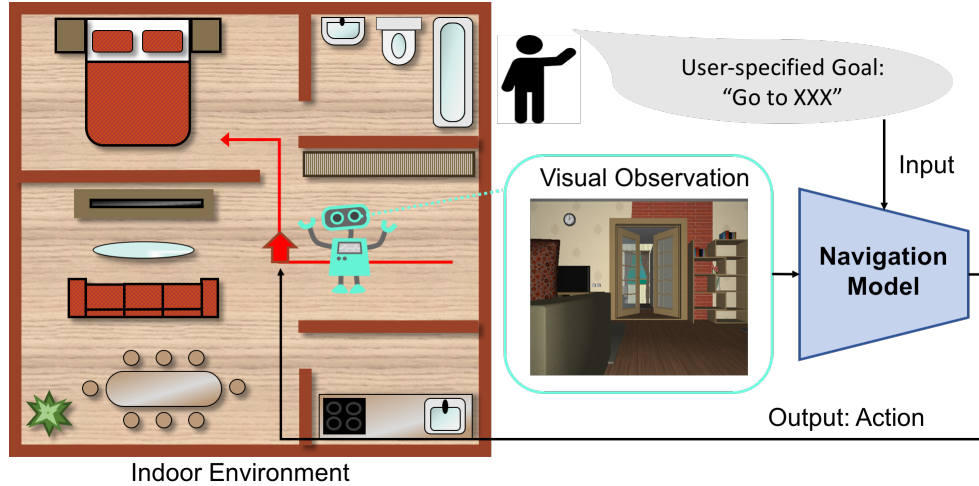


Figure 1.2: An illustration of a visual navigation task.

agent (a.k.a. robot) is instructed to navigate towards a user-specified goal in an environment based on its first-person visual observations (typically the RGB images captured by its on-board camera). The environment information, such as geometric/topological/occupancy/semantic map of the environment, is typically unavailable even for training since collecting the ground-truth environment information is unarguably expensive and sometimes impractical for real-world applications. As a result, the transition function  $T$ , together with agent's states  $S$  and the goal states  $G$  defined by the agent's positions and the goal positions in the environment are not available and can only be inferred from the agent's visual observations  $\Omega$  and the goal descriptions  $G_d$ . The goals are usually described by labels (e.g. semantic labels of objects), images (e.g. visual observations at goal locations) and natural language (e.g. navigation instruction). Typically, there should be goal recognition functions  $\Phi$  determining if the current visual observations satisfy the goal descriptions. For example, when asking the agent to navigate to an object by specifying the navigation goal with the object category, the object recognition function that describes how likely if the object presents in the observations can be used as  $\Phi$ . The objective of the visual navigation



task is to achieve the specified goal  $G$  as soon as possible, requiring an elaborate design of the reward function  $R$ .

There are several challenges presented in learning the action policy  $\pi : \Omega \times G_d \rightarrow A$ . First, the observations  $\Omega$ , especially the visual observations, together with the goal descriptions  $G_d$  are always the high dimensional inputs to the policy function  $\pi$ . Therefore, large amounts of parameters would be introduced in the function  $\pi$  that require great quantities of data to learn. The collection of the large training data needs either extensive feedbacks or efficient sampling process while both of them are challenging: 1) Although prohibitively expensive human-labeled action policies are not required for reinforcement learning algorithms, reward functions as feedbacks still need to be well designed. However, unlike video games where the game scores are the natural sources of the rewards, for most tasks, designing a good reward function which is both correct (i.e. leads to the desired outcomes) and learnable (i.e. rewards happen early and often enough) is extremely difficult (Abbeel and Ng, 2004). 2) As a result of the 1), a task, especially a complex task, typically yields a sparse reward setting. In consequence, the agent may completely fail to learn and improve its sample efficiency as it is unlikely to encounter and sample the very few rewarding states without a good exploration strategy.

Second, the observations  $\Omega$  and/or goal descriptions  $G_d$  are often the high dimensional projections of the true latent states, making the agent tend to capture spurious features and overfit to the training tasks easily, which is not desirable. Ideally, the learned policy should be robust to the observation variances and even be adaptive to the environment dynamics so that we can generalize it towards unseen environments and new goals rather than training a new policy for each single environment and goal. To achieve this objective, we need to estimate the agent's states from the observations and/or inferring the goal states from the goal descriptions, and the desired state

representations should be relevant to the underlying state transition functions and reflect the environment dynamics. It is extremely challenging because 1) the observations and the goal descriptions as the policy inputs may only capture the partial information of the environments; 2) the limited number of training tasks cannot well represent the underlying task distributions.

## 1.2 Related Work

### Visual Understanding

In the last few decades, deep learning based approaches have been extensively studied and applied in the field of computer vision (Guo *et al.*, 2016). Their state-of-the-art performance make them promising in helping an agent to acquire various visual understanding abilities. For example, the image classification approaches (Simonyan and Zisserman, 2014; Szegedy *et al.*, 2015; He *et al.*, 2015; Krizhevsky *et al.*, 2017) which label input images with a probability over a set of pre-defined classes can tell the agent what presents in its visual observations. The object detection algorithms (Ren *et al.*, 2016; Redmon *et al.*, 2016) enable the agent to localize a target object in its visual observations. Furthermore, even the pixel-level classes can be predicted if with the advanced semantic segmentation methods (Long *et al.*, 2015; Chen *et al.*, 2017a). Similarly, the pixel-level depth values can also be well estimated (Liu *et al.*, 2015; Fu *et al.*, 2018). In addition, to achieve a better and deeper understanding of the visual observations, there are a large amount work focus on the tasks that are at the intersection of computer vision and natural language processing. For example, the image captioning task that asks to automatically generate a natural language description of an image (Yang *et al.*, 2011; You *et al.*, 2016; Rennie *et al.*, 2017; Aneja *et al.*, 2018; Hossain *et al.*, 2019), the natural language object retrieval task

that requires to localize the natural language specified objects in an image, and the visual question answering tasks in which a free-form natural language question about a given image is asked (Antol *et al.*, 2015; Goyal *et al.*, 2017; Yi *et al.*, 2018). All of these work have achieved superior performance and played significant roles in visual understanding, yet they only work on static and fixed visual observations. In this dissertation, we focus on the agent that can take actions to actively perceive and we study the problem of how the visual understanding can help the agent act to perform tasks.

## **Embodied AI Agents**

To transform the agent from a passive observer to an active perceiver, the community begins to show increased interest in embodied AI tasks in which an AI agent needs to take actions in either physical or virtual environments to tackle the tasks. Embodied AI tasks encompasses a variety of tasks including the visual navigation (Gupta *et al.*, 2017; Zhu *et al.*, 2017; Yang *et al.*, 2018; Ye and Yang, 2020), the object manipulation (Yang *et al.*, 2015; Bütepage *et al.*, 2019) and the natural language based instructions following (Anderson *et al.*, 2018b; Fu *et al.*, 2019). Among all of these tasks, the visual navigation task has drawn extensive attentions due to a number of simulated platforms, such as AI2THOR (Kolve *et al.*, 2017b), House3D (Wu *et al.*, 2018), and Matterport3D (Chang *et al.*, 2017), have been developed that enable and accelerate the learning process of the visual navigation system. Given different forms of the goal descriptions, the visual navigation task can be further categorized into point-goal navigation (Gupta *et al.*, 2017), object-goal navigation (Yang *et al.*, 2018), image-goal navigation (Zhu *et al.*, 2017) and natural language-goal navigation (Das *et al.*, 2018a; Anderson *et al.*, 2018b) where the goals are described as the points in a coordinate frame, the semantic labels of objects, the images taken at the goal

positions and the natural language questions or instructions respectively. To learn an action policy to perform such tasks, most of the previous work assume the environment information, such as the map information is available during the training process. In such a case, they can generate additional supervisions, such as defining the reward function with the distance between the robot’s current location and the target location (a.k.a. reward shaping) (Wu *et al.*, 2018; Mousavian *et al.*, 2019), adopting shortest path as the supervised signal for pre-training (Gupta *et al.*, 2017; Fried *et al.*, 2018; Das *et al.*, 2018a; Wang *et al.*, 2019), and/or gradually increasing the distance between robot’s starting location and the target location (a.k.a. curriculum learning) (Das *et al.*, 2018b; Kulhánek *et al.*, 2019). However, such environment information is available only in the simulated platforms and is very difficult to get in the real world. As a result, they only have limited training environments and their models are very likely to overfit to the training environments. Some methods entirely bypass the explicit vision modules and directly map the agent’s visual observations to actions with the underlying assumption that the visual understanding capabilities can be automatically learned as needed (Zhu *et al.*, 2017; Yang *et al.*, 2018). Unlike these methods, we study the embodied AI tasks in a more general and practical way where we don’t assume any expensive or even unrealistic supervisions is available. In addition, we study how the visual understanding capabilities from the explicit vision modules can better help the agent to learn the action policy for the tasks.

### 1.3 Contributions

In this dissertation, we present several novel methods to address the challenges in learning the action policy  $\pi : \Omega \times G_d \rightarrow A$  for a visual task. First, to enable the policy learning from the high dimensional inputs, we improve the data collection efficiency in two ways, i.e. introducing additional supervisions/feedbacks and developing an ef-

efficient exploration strategy for the agent. Specifically, 1) we propose a denser reward function by understanding the visual observations of the task. While previous work typically designs or learns a reward function with ground-truth environment information, human-labeled action policy or human expert demonstrations that require significant effort and time, we here leverage advanced computer vision techniques to reward the agent by performing visual prediction tasks. The visual prediction tasks can be trained with dataset that either is available or requires less human labors. The experimental results demonstrate that our reward function, though not accurate, does help policy learning from visual observations. 2) We also propose a novel hierarchical policy learning paradigm for efficient exploration in high dimensional state space. Instead of taking the high dimensional visual observations as the sub-goal space which is extremely challenging to learn, we build a low dimensional sub-goal space by understanding the visual observations. The proposed hierarchical policy learning paradigm is demonstrated to help the agent sample efficiently even in the sparse reward setting and thereby learn to perform the task in a more optimal and interpretable way.

Second, to make the learned policy generalizable towards new task instances that might have different visual observations, goal descriptions and even different environment dynamics, we present novel methods learning the state representations that unravels the underlying state transition functions. To be specific, 1) we extract task-relevant visual representations from a vision module to learn a generalizable action policy for fully observed task, i.e. task that goals are visible in visual observations. The vision module is trained in a supervised way so that it can deal with the visual variance more efficiently. In addition, our extracted visual representations are able to capture the local environment dynamics, making the learned policy generalize better for the fully observed task. 2) We present a two-layer hierarchical reinforcement learning approach equipped with a Goals Relational Graph (GRG) for tackling the

partially observable task, i.e. task where goals are not always observable which is very common as the environments typically can only be partially observed by the visual observations. Our GRG captures the underlying relations of all goals in the goal space as a representation of the high-level environment dynamics that helps our high-level network generalize. Our low-level network which focuses on the fully observable task can utilize the our method introduced in 1) to generalize as well. As a result, our overall system exhibits superior generalization performance on both unseen environments and new goals. 3) We propose a promising research direction which is to learn from human-provided priors the high-level environment dynamics. While it is impractical for human experts to provide the conventional detailed low-level environment dynamics, specifying the high-level one with natural language (Anderson *et al.*, 2018b; Qi *et al.*, 2020) or demonstration videos (Ye *et al.*, 2019b) is efficient. Meanwhile, inferring the high-level environment dynamics is also more tractable. With the inferred environment dynamics, the action policy can be learned to be generalizable.

#### 1.4 Dissertation Outline

The rest of the dissertation starts with a study in Chapter 2 on defining a denser reward function by understanding the visual observation of the task (Ye *et al.*, 2018). It is then followed by a study in Chapter 3 on developing a hierarchical reinforcement learning paradigm for efficient exploration (Ye and Yang, 2021a). After an action policy can be well learned from visual observations for a training task, we further present methods to make the learned policy generalizable towards new task instances. In Chapter 4, we report the learning of generalizable policy for fully observable task by capturing local environment dynamics (Ye *et al.*, 2019a). In Chapter 5, we explore the learning of generalizable policy for partially observable task by capturing global environment dynamics (Ye and Yang, 2021b). Finally, we highlight the possibility of

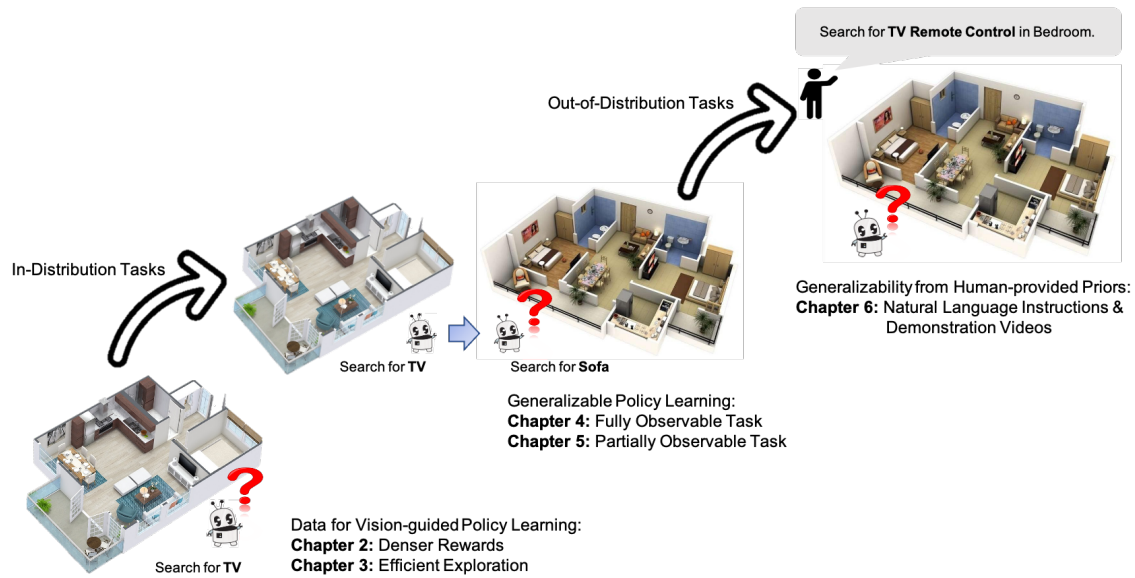


Figure 1.3: The outline of the dissertation.

inferring environment dynamics from human-provided priors for generalizable policy learning in Chapter 6. All the work are concluded in Chapter 7. Figure 1.3 depicts the outline of this dissertation.

## REWARD FUNCTIONS FROM VISUAL UNDERSTANDING

**2.1 Introduction**

Reinforcement learning (RL) has demonstrated its power at enabling robots with autonomous behaviors (Arulkumaran *et al.*, 2017), such as navigating over an unknown environment (Mirowski *et al.*, 2016; Zhu *et al.*, 2017), manipulating objects with robot’s end effectors (Gu *et al.*, 2017; Popov *et al.*, 2017; Rajeswaran *et al.*, 2017), and motion planning (Chen *et al.*, 2017b; Everett *et al.*, 2018). 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. However, for most of the tasks, there is no natural source for the reward signals. Instead, the reward signals need to be hand-crafted and carefully designed to accurately represent the task. Typically, it is also necessary to manually tweak the rewards until desired behavior is observed.

The difficulty of manually defining a reward function motivates many researchers to extract the reward function automatically by observing a human expert performing the task (Abbeel and Ng, 2004; Arora and Doshi, 2018). Unlike these works that leverage expert knowledge to convert a task description into a compact reward function, we study how an agent can reward itself by understanding its visual observations. Specifically, we focus on robotic object search task in this chapter in which a robot needs to locate a user-specified target object in indoor environments using only visual observations. To facilitate the robot to perform the task, we propose an object



recognition module and build informative rewards upon the recognition results. With our proposed reward function, the robot learns a better action policy.

## 2.2 Related Work

Deep reinforcement learning has been studied extensively for the target-driven visual navigation tasks (Ye and Yang, 2020). These tasks can be categorized in terms of the description of the navigation target. Zhu *et al.* (2017) and Kulhánek *et al.* (2019) specify the navigation target by the image taken at the target location. The robotic object search task studied in Mousavian *et al.* (2019), Ye *et al.* (2019a), Yang *et al.* (2018) and Druon *et al.* (2020), and the room navigation task introduced in Wu *et al.* (2018) and Wu *et al.* (2019b) take the semantic label of the target object and room as the navigation target. The Embodied Question Answering (Das *et al.*, 2018a,b; Gordon *et al.*, 2018) and the Vision-and-Language Navigation (Anderson *et al.*, 2018b; Wang *et al.*, 2019) address the problem where the navigation target is provided with an unconstrained natural language. Among all of these works, many of them adopt a sparse reward function where a high reward is defined only at the goal state and at all other intermediate states, the reward is either zero or a small negative value (Zhu *et al.*, 2017; Yang *et al.*, 2018; Kulhánek *et al.*, 2019). Other works usually define the reward function with the distance between the agent’s current location and the target location under the strict assumption that the full information of the environment is known (Mousavian *et al.*, 2019; Wu *et al.*, 2018; Wang *et al.*, 2019). Here, we introduce an informative reward function by adopting an object recognition module on the agent’s visual inputs while we do not assume any environment information available.

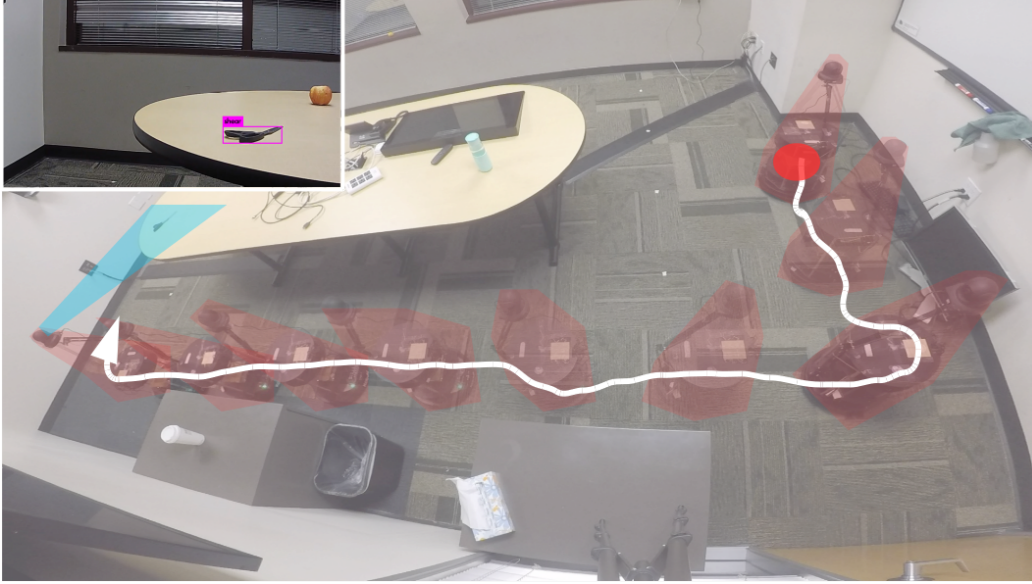


Figure 2.1: An illustration of our robotic object search task. Red dot: the random initial location; White line and error: the generated robot trajectory from our turtlebot experiment; Upper left image: the final view of the robot with the target object detection (the target object in this experiment is “shear”).

### 2.3 Our Approach

Our objective is to learn an action policy for the robot to locate a user-specified target object in indoor environments using only visual inputs. Here, the environment information, such as the map of the environment or the location of the target object is unknown. The robot makes action decisions only upon a single stream image sequences from an on-board camera mounted in the robot and an RGB image depicting the target object with no contextual information. As a result, the state observations are the RGB images captured by the robot’s on-board camera, and the goal descriptions are the images of the target objects. With the learned policy, when the user provides the robot an image of a target object, the robot is expected to take a relatively small number of steps to approach the target object from its random

starting position. Moreover, the robot is expected to return a bounding box of the target object in its viewpoint (as shown in Fig 2.1). To learn the action policy, we first propose an object recognition module to enable the robot to detect the given object in its viewpoint. The recognition results are then used to reward the robot in order to learn the policy with our recognition-guided action policy learning module. We describe the two modules respectively in the following sections.

### 2.3.1 Object Recognition Module with Target Object Given

A standard object detection process typically consists of two steps: 1) detect the candidate regions of objects in an image; and 2) predict a class label to each region (a bounding box representation is typically used). However, under our setting, the target object is given, and the object recognition module only needs to detect whether the specific object exists in the current range of view at any time step. Following this observation, we simply take the target object image as the first input to our detection network, along with the whole image of robot’s current view as the second input, and predicts the bounding box coordinates of the target object in the current view if there is one. Fig. 2.2 shows an illustration of the proposed network architecture.

**Network architecture:** As shown in Fig. 2.2, the module takes both the current view (full image frame) and the target object image and feed them into a shared, ResNet-50 network (He *et al.*, 2016) with a siamese-like architecture and extracts the 1024 dimensional features at the output of the *res4f* layer for both inputs. Here, we take the pre-trained ResNet model (trained on ImageNet) and fix them during our subsequent training and testing pipelines. A fully-connected layer is attached after each of the 1024 dimensional feature inputs and projects them down into a 512 dimensional vector with a single fully connected layer. The two 512 dimensional vectors are then concatenated and aggregated into a 512 dimensional joint vector by

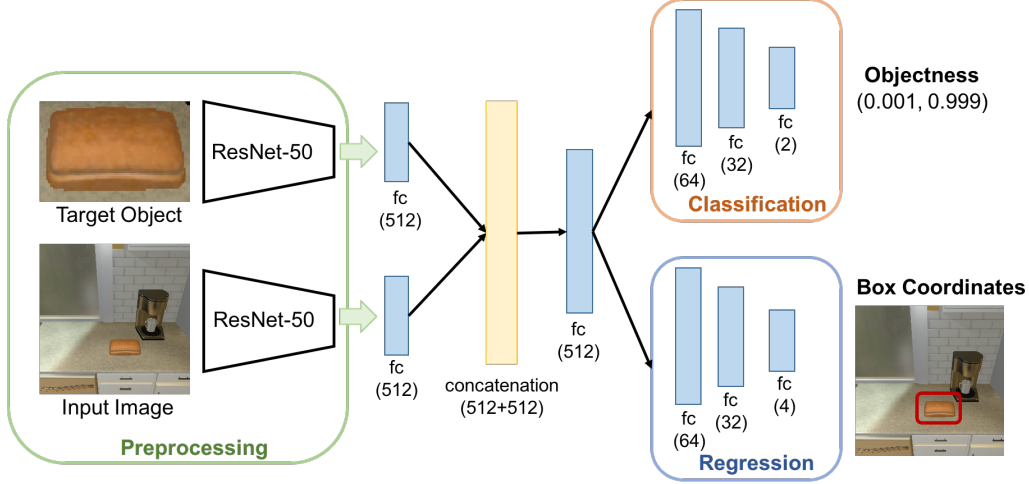


Figure 2.2: The architecture of the object recognition network.

an additional fully-connected layer. Finally, we feed the 512 dimensional joint vector into a classification layer and a regression layer, each of which includes another two fully-connected layers. The classification layer is designed to predict whether the target object appears in the input whole image or not, and the regression layer is designed to predict the 4 parameters of the target object bounding box, i.e. the center coordinate  $(x, y)$ , the width  $w$  and the height  $h$  of the bounding box.

**Loss function:** Eq. 2.1 shows the loss function we designed for the object recognition module.

$$\mathbf{L} = \sum_i -[p_i^* \log(p_i) + (1 - p_i^*) \log(1 - p_i)] + \lambda \sum_i \mathbf{1}_i^{obj} \|\mathbf{b}_i - \mathbf{b}_i^*\|_2^2 \quad (2.1)$$

The first term denotes the cross-entropy loss for binary classification, where  $p_i^*$  is the ground-truth label of the  $i$ th image and its value equals to 1 if the object appears in the image, and 0 if it doesn't.  $p_i$  is the predicted probability of the object appears in the image  $i$  that is the output from the previous classification layer. The second term in the loss function is the  $L_2$  loss between ground truth bounding box coordinates  $b_i^*$  and that predicted coordinates  $b_i$ , where  $b_i = (x_i, y_i, w_i, h_i)$  (likewise for  $b_i^*$ ).  $\mathbf{1}_i^{obj}$  is an

indicator function indicates that if the object is in the image  $i$ . In this formulation, the regression loss will only be activated when the target object is detected in the current view.  $\lambda$  is the weight factor that balances between these two losses. In practice we found a weight value of 0.5 works well, so we fix it throughout all our reported experiments. The object detection network is trained by minimizing the overall loss function with the standard stochastic gradient decent (SGD) optimization.

### 2.3.2 Recognition-guided Action Policy Learning

With the guidance from the object recognition module, we build upon a basic deep reinforcement learning based framework to learn a mapping from the robot’s visual observations to its action space as an action policy. Since the goal of the robot is to find the image specified target object, we define goal states as those positions where the robot’s visual observations contain a bounding box of the target object. At the same time, the size of the bounding box needs to be larger than a predefined threshold to determine a success. In practice, we found the size of the fifth largest bounding box (among all the ground-truth bounding box instances) is a reasonable threshold, and it yields 5 goal states (top 5 images with the largest target object detected). To learn the action policy, we first define an informative reward function in terms of the recognition results from the object recognition module. The reward function guides the robot towards the goal states. We then approximate the robot’s action policy with a deep neural network and learn it with the A3C algorithm (Mnih *et al.*, 2016).

**Reward function:** At each state, we apply our object recognition module on the state observation and we define the reward as the one proportional to the size (the area) of the detected bounding box that contains the target object, if there is one. For all states where the robot cannot detect the object, we set the reward to zero.

Since there might exist many states where the robot can detect a bounding box with a smaller-than-the-threshold size, the robot can easily stuck in these states, since they all yield positive rewards. In order to encourage the robot to keep searching states with possibly larger detected bounding boxes, we further set reward as zero to a state, if at this state the robot do detect a bounding box but the size of it is smaller than the bounding boxes that have been detected earlier in one training episode. In other words, our proposed reward function keeps the records of the largest box size in the current episode and accumulates discounted rewards in an incremental way with respect to the records. Figure 2.3 shows an example.

More formally, let  $a_t$  be the size of the bounding box the robot detects at time step  $t$ , the reward at this time step  $r_t = ka_t$  if and only if  $a_t > a_{t-1}, a_{t-2}, \dots, a_0$ , otherwise  $r_t = 0$ .  $k$  is a constant. As a result, the discounted cumulative reward for one episode will be  $\gamma^{i_1}ka_{i_1} + \gamma^{i_2}ka_{i_2} + \dots + \gamma^{i_t}ka_{i_t}$ , where  $\gamma$  is the discount factor for time penalty and  $a_{i_1} < a_{i_2} < \dots < a_{i_t}$  ( $i_1 < i_2 < \dots < i_t$ ). The rationale supporting this design is that this reward function encourages more exploration only around these aforementioned and potential trapping states, instead of having a higher uniform exploration rate across the whole state space, regardless of whether these states being worth exploring or not. In this way, we can achieve both faster convergence and more meaningful exploration paths at the same time.

**Network architecture:** Fig. 2.4 depicts the architecture of our proposed deep reinforcement learning-based model for approximating the action policy. The model shares the similar architecture with Zhu *et al.* (2017) and takes both the observed visual view and the target object image as the inputs. Unlike Zhu *et al.* (2017), our model does not share the weights between the two fully connected layers when fusing the target and the observation images. One primary motivation for this is that under our setting, the input image of the target object could come from very different

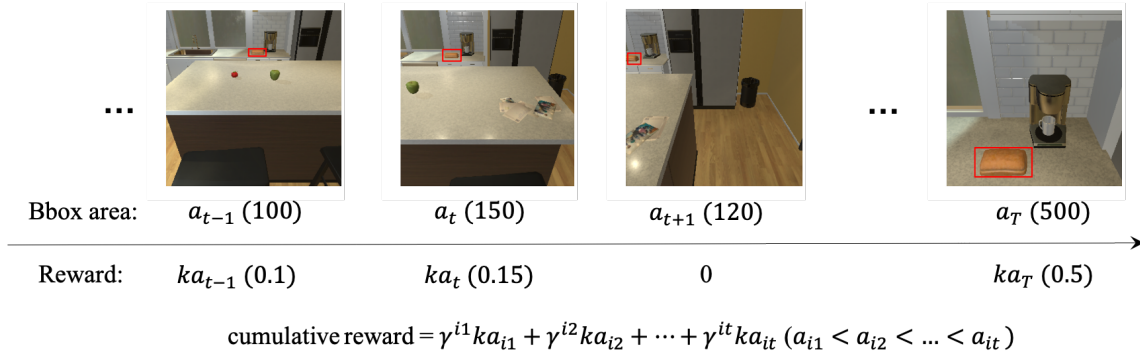


Figure 2.3: An illustration of our vision-guided reward function.

domains from the object observed from the scene image so that it can't be projected to the same embedding space as the observation image. In addition, we further feed the object location information generated from the object recognition module into the embedding fusion layer. Here, the object location information is encoded as a 5 by 5 binary image that specify the object's location. The last scene-specific layer is used to predict the action policy  $\pi(a|s, g)$  and the value  $V(s, g)$  for each scene. The action policy  $\pi(a|s, g)$  is a 8 dimensional probabilities over 8 actions, namely  $\{move\ forward/backward/left/right, tilt\ up/down, turn\ left/right\}$ .

**Loss function:** We adopt the A3C algorithm (Mnih *et al.*, 2016) to update the network and learn the action policy. A3C optimizes the policy outputs by minimizing the loss function  $L_p = -\mathbb{E}[\sum_{t=1}^T (R_t - V(s_t, g)) \log \pi(a_t | s_t, g)]$ , where  $R_t = \sum_{i=0}^{T-t} \gamma^i r_{t+i} + V(s_{T+1}, g)$ . The value output is updated by minimizing the loss  $L_v = \mathbb{E}[(R_t - V(s_t, g))^2]$ . Finally, the overall loss is  $L = L_p + \lambda L_v$  where  $\lambda$  is a constant coefficient and we set it as 0.5 in our experiments.

## 2.4 Experiments

We evaluate our framework in both simulation and real environments. In the simulated environment, we set up a variety of experiments with multiple target objects

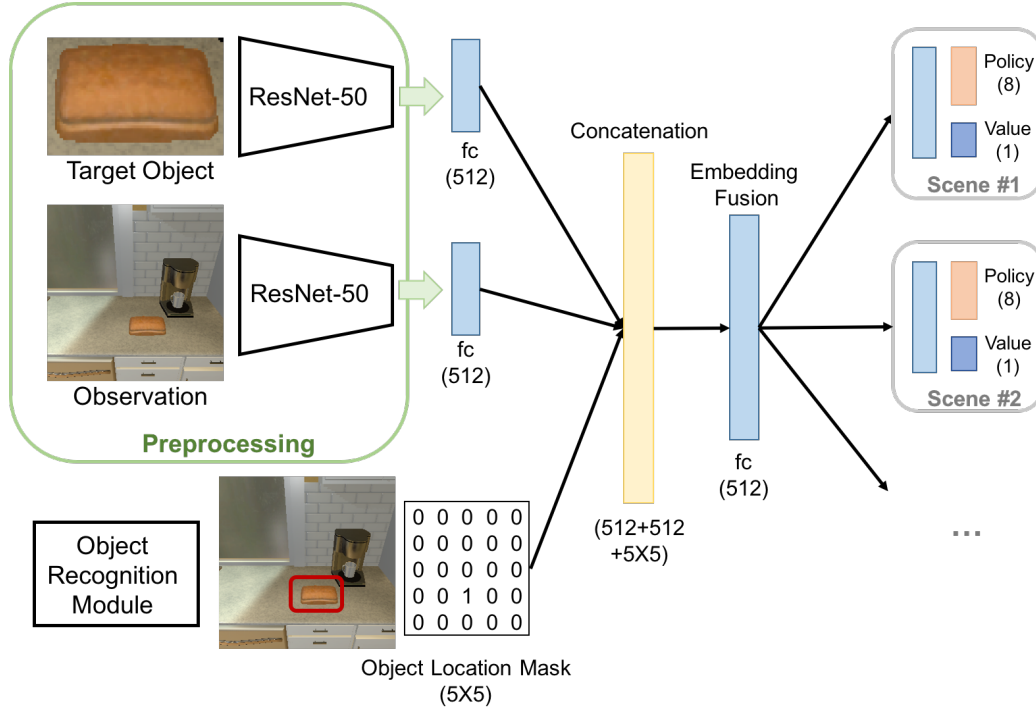


Figure 2.4: The deep reinforcement learning architecture.

in four indoor scenes. We further implement our framework on a real mobile robot platform (a turtle-bot with a pan-tilt camera) and demonstrate its efficacy in a real indoor scene (a conference room) in finding the target objects.

### 2.4.1 Dataset

**Simulation platform** is adopted from the THOR Challenge platform<sup>1</sup>. This platform provides 30 photo-realistic indoor scenes from AI2-THOR dataset (Kolve *et al.*, 2017a) for training and validating autonomous robotic systems to navigate and search for objects in these virtual environments. The platform serves as the testbed for our system. More specifically, the 30 indoor scenes include 15 kitchens and 15 living rooms, where 20 of them are training scenes and the rest 10 are reserved for

<sup>1</sup><http://vuchallenge.org/thor.html>



validation. The available actions for the robot are predefined as 8 discrete actions, namely  $\{move\ forward/backward/left/right, tilt\ up/down, turn\ left/right\}$ . Following the simulation setting, each virtual scene can be discretized into a set of images taken from each robot state, and the whole set of images characterize the overall state space of the robot. Moreover, the platform also provides a set of target images for each scene. These target images contain only objects without any background, which can well-support our experiments.

In order to train the object recognition module, we need to further augment the data with annotated object bounding boxes. Without the loss of generality, we select 4 scenes randomly and for each scene we retrieve all images from every robot state. We further select 4 objects that can be found in the scene and take their corresponding target images as the input target images to our system. Finally, we manually labeled the bounding box for each target object in each scene image, and treat them as the ground-truth bounding boxes in training our object recognition module. We adopted the standard cross-validation mechanism to mitigate the risk of over-fitting.

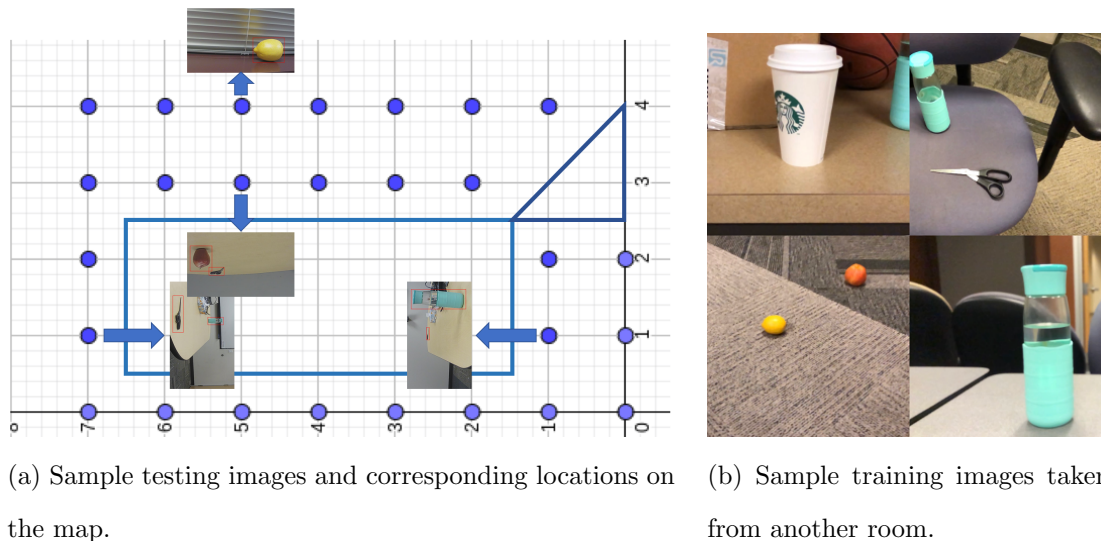


Figure 2.5: Sample testing and training images from our real world dataset.

**Real world scenario:** we equipped a turtle-bot with a pan/tilt camera to conduct the experiment. The action set available for the turtle-bot consists of,  $\{move\ forward/backward, rotate\ left/right, look\ up/down\}$ . We discretized the scene space (which is a conference room) into 27 locations. At each location, the robot takes 8 RGB images with its pan/tilt camera (turn left/right, look up/down). This leads to a total of 216 RGB images for our experiment, representing all possible states of the turtle-bot in this real world scenario. For the objects presented in the scene, we collected another group of 150 images each for training the object recognition module. Then, we train the deep neural networks in our framework with reinforcement learning. In testing, we set the turtle-bot at a random starting location in the conference room, and search for a given object with the trained model. Fig. 2.5 shows a few sample testing and training data.

#### 2.4.2 Experimental Results and Discussion

Table 2.1 and Table 2.2 report the performance of all the methods in performing the robotic object search task at the simulation platform and the real world respectively. We designed four experiments as following.

1) Locate 1 object in 1 simulated scene. In order to test whether our proposed framework can successfully find the target object or not in one scene, we first conduct this experiment.

2) Locate 4 objects in 1 simulated scenes. To validate the generalization ability of our method to multiple objects and avoid over-fitting to one specific target object, we train one model to find any given one of 4 different objects.

3) Locate a total of 16 objects in 4 different simulated scenes. To verify that the models have the generalization ability across different scenes, we trained one model which has 4 scene-specific branches as shown in Fig. 2.4 to learn 4 action policies for

Table 2.1: The performance metrics of all the methods in the simulation platform.  
(AL: Average trajectory Length; SR: Success Rate)

Methods	Experiment Scenarios					
	1 obj in 1 scene		4 objs in 1 scene		16 objs in 4 scenes	
	AL	SR	AL	SR	AL	SR
Random Walk	2050.3	60.0%	1911.6	72.5%	2057.6	84.3%
Reward Func. 1	2880.2	50.0%	~	~	~	~
Reward Func. 2	-	0.0%	~	~	~	~
Our Reward Func. (high exploration)	1957.6	80.0%	1643.2	92.5%	1430.7	87.5%
Our Reward Func. (low exploration)	52.9	100.0%	30.1	75.0%	593.0	75.0%

4 scenes.

4) Turtle-bot experiment in a real world scenario. We conduct an experiment using a turtle-bot to search for an object in a real world scene (a conference room) to validate the real-world efficacy of our method.

For each experimental setting, we compare our method with the following baseline and variants to demonstrate the superiority of our method. Note that we don't compare our method with the classical search algorithms, such as  $A^*$ , depth-first or breadth-first search since we assume the global map of the environment is unknown. The robot will simply remain on the current state if the action it takes cause collision. This non-deterministic transition characteristic makes these deterministic algorithms unusable.

Table 2.2: The performance metrics for the real world experiments. (AL: Average trajectory Length; SR: Success Rate)

Experiment Scenario	Methods					
	Random Walk		Our Reward Func. (high exploration)		Our Reward Func. (low exploration)	
	AL	SR	AL	SR	AL	SR
turtle-bot experiment	820.9	99.0%	176.8	100%	63.3	100%

1) Random walk. In this baseline method, the robot randomly takes an action from its available action set at each state. We take this method as our baseline.

2) Reward function 1. We adopt our model architecture but with a reward function different from the one defined in Zhu *et al.* (2017) in training. The reward function is defined to give a positive reward (10) at goal states, and a small negative reward ( $-0.01$ ) for all the other states.

3) Reward function 2. We trained our model with another intuitive reward function. We define the reward as the one proportional to the area of the object’s bounding box at each state, no matter if the size of the current bounding box is smaller than the previous one. This means that if  $a_0, a_1, \dots, a_s$  where  $a_i (0 \leq i \leq s) \geq 0$  are the areas of bounding boxes the robot have seen during one episode, the total reward for this episode is  $ka_0 + \gamma ka_1 + \dots + \gamma^s ka_s$  and  $\gamma$  is the discount factor.

4) Our reward function with high exploration rate. We use our reward function defined in Sec. 2.3.2 to train our model, with a relatively high exploration rate.

5) Our reward function with a low exploration rate. We reduce the exploration rate to train our model without any other change compared to method 4).

We report the performance of all methods with two metrics, namely the average number of actions that need to be taken to find the target object (also known as the average trajectory length), and the success rate of the robot finally finds the target object. For a fair comparison, for each target object, we randomly initialize the robot’s starting position and run each method for 10 episodes. An episode ends when the robot finds the target object or it has already taken 5000 steps (and it claims a failure). Only when the robot finds the target object successfully, we count the episode as a successful trail, and the corresponding trajectory length will then contribute to the average trajectory length metric. “-” indicates the model does not converge or during testing the robot is trapped in sub-optimal states for good. Here we want to mention a caveat: because both the reward function 1 and 2 do not perform well even on the 1 object 1 scene scenario and due to the limited computing resource we have, we did not test them for the other three scenarios (we use “~” in the table).

We observed that: 1) trained model based on our proposed reward function outperforms other baseline methods with a significant margin (less steps to find the target object with a higher success rate; 2) our experiments show that reward function 1 and 2’s performances are worse than random walk, or can not converge, which indicates they are not suitable for the general robotic object search task; 3) model trained with multiple objects and multiple scenes takes more steps to achieve higher success rates with the high exploration rate, indicating that the model lacks of good generalization ability.

## 2.5 Conclusion

In this chapter, we present a denser reward function building upon a vision module that enables a robot to develop an action policy to perform the robotic object better.

Experiments conducted on both public AI2-THOR (Kolve *et al.*, 2017a) platform and a new indoor environment dataset (a conference room), in simulation and on a physical robot executing the challenging object searching task validated the proposed approach.

The study presented in this chapter also opens several avenues for future research. First, some other visual prediction tasks that achieve state of the art performance in the field of computer vision can also be adopted to define reward functions. For example, in a dynamic environment, the robot can predict the movements of other agents or human collaborators from its visual observations to avoid collisions with them (Wang *et al.*, 2017). Moreover, the visual prediction errors can also be taken as indications of insufficient exploration of the environments to drive the robot to explore more efficiently (Pathak *et al.*, 2017).

## EFFICIENT EXPLORATION WITH HIERARCHICAL POLICY

**3.1 Introduction**

A pressing challenge to train an agent to perform complex tasks with reinforcement learning (RL) methodology is the sparse reward issue. With well-designed reward functions, such as the ones in Atari games (Mnih *et al.*, 2015), the learned policies are shown to achieve extremely promising performance. However, it is still a challenge designing the reward function for the real-world applications (Abbeel and Ng, 2004). Typically, for applications such as target-driven visual navigation, prior research constructs the reward function in terms of the distance between the agent’s current location and the target location under a strict assumption that the full information of the environment is known (Mousavian *et al.*, 2019; Wang *et al.*, 2018, 2019). For an unknown environment, a straightforward way is to set a high reward when the agent reaches the final goal state while at all other intermediate states, the reward is either zero or a small negative value (Zhu *et al.*, 2017). In such a sparse reward setting where the reward is only defined for a small subset of the states, the agent struggles to learn a good action 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 (Nachum *et al.*, 2019) and superiority under the sparse reward setting (Kulkarni *et al.*, 2016; Le *et al.*, 2018; Levy *et al.*, 2018). HRL aims to learn multiple layers of policies. The higher layer breaks down the task into several eas-

ier 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 Le *et al.* (2018) and Nachum *et al.* (2018b) for tasks with low dimensional state spaces. Unfortunately, these methods are not directly applicable for the tasks where the underlying states are unknown and the state observations are 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 (Nachum *et al.*, 2018a; Dwiel *et al.*, 2019).

In this chapter, we put forward a novel two-layer hierarchical policy learning paradigm for robotic object search task. The robotic object search task asks a robot to search and approach an object in an unknown environment where the state observations are the RGB images captured by the robot’s on-board camera and the goal descriptions are the semantic labels of the target objects. 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 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 Bacon *et al.* (2017), 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. 3.1). We validate *HIEM* with extensive sets of experiments on the House3D (Wu *et al.*, 2018) 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.

### 3.2 Related Work

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 Nachum *et al.* (2018b), the authors introduce an off-policy correction method. Levy *et al.* (2017b) and Levy *et al.* (2018) 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 tasks like 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, Kulkarni *et al.* (2016) 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 Andreas *et al.* (2017) and Sohn *et al.* (2018). Closer related to our work, Das *et al.* (2018b) 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, Gordon

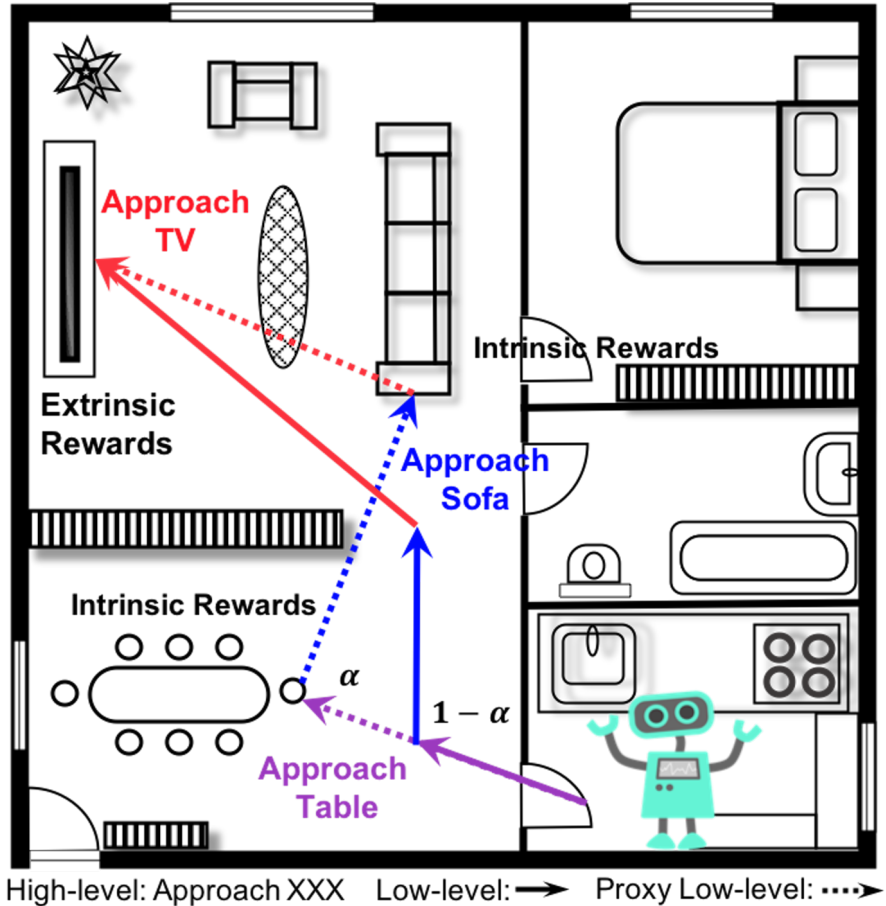


Figure 3.1: An example of our *HIEM* framework. When our high-level policy proposes a sub-goal, our proxy low-level policy is invoked with the probability of  $\alpha$  to explore the environment by optimizing towards the sub-goal, and our low-level policy learned from the exploration experience is invoked with the probability of  $1 - \alpha$  to collaborate with the high-level policy to better achieve the goal.

*et al.* (2018) 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 tem-

poral abstractions. Bacon *et al.* (2017) proposes the option-critic framework to autonomously discover the specified number of temporal abstractions. Osa *et al.* (2019) learns the temporal abstractions through advantage-weighted information maximization. Nachum *et al.* (2018a) 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.

### 3.3 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 chapter, 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.

### 3.3.1 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  $sg_t \sim \pi_h(sg|s_t, g)$  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 Ye *et al.* (2019a). 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 Kulkarni *et al.* (2016), 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  $\{move\ forward / backward / left / right, turn\ left / right\}$ .

### 3.3.2 Extrinsic 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,  $r_t^e(s_t, g) = 1$  if and only if  $s_t$  is a goal state of the goal  $g$ , otherwise  $r_t^e(s_t, g) = 0$ .

**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  $r_t^i(s_t, sg) = 1$  if and only if  $s_t$  is a goal state of the sub-goal  $sg$ , otherwise  $r_t^i(s_t, sg) = 0$ .

### 3.3.3 Model Formulation

We formulate the object search task in terms of the two rewards introduced in Sec. 3.3.2. 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

$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}^e | s_0, g, sg_0]$ . 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  $Q_h^e$  (Sutton and Barto, 2018) for our high-level layer, i.e.  $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}^e | s_0 = s, g = g, sg_0 = sg] = Q_h^e(s, g, sg)$ . Following the option-critic framework (Bacon *et al.*, 2017), we unroll the  $Q_h^e(s, g, sg)$  as,

$$\begin{aligned} Q_h^e(s, g, sg) &= \sum_a \pi_l(a|s, g, sg) \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}^e | s_0 = s, g = g, sg_0 = sg, a_0 = a] \\ &= \sum_a \pi_l(a|s, g, sg) Q_l^e(s, g, sg, a), \end{aligned} \quad (3.1)$$

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

$$\begin{aligned} Q_l^e(s, g, sg, a) &= \sum_{s'} P(s'|s, a) [r^e(s', g) + \gamma U(g, sg, s')], \\ U(g, sg, s') &= (1 - term(s', g, sg)) Q_h^e(s', g, sg) + term(s', g, sg) V_h^e(s', g), \\ V_h^e(s', g) &= \sum_{sg'} \pi_h(sg'|s', g) Q_h^e(s', g, sg'). \end{aligned} \quad (3.2)$$

We parameterize  $Q_h^e(s, g, sg)$ ,  $Q_l^e(s, g, sg, a)$  and  $term(s, g, sg)$  with  $\theta_h^e$ ,  $\theta_l^e$  and  $\theta_t$  respectively. Then the high-level policy  $\pi_h(sg|s, g) = \mathbf{1}(sg = \operatorname{argmax}_{sg} Q_h^e(s, g, sg))$ , and the low-level policy  $\pi_l(a|s, g, sg) = \mathbf{1}(a = \operatorname{argmax}_a Q_l^e(s, g, sg, a))$ . We adopt the DQN (Mnih *et al.*, 2015) based method to learn  $Q_h^e(s, g, sg)$  and  $Q_l^e(s, g, sg, a)$  in which we update both of the values towards the 1-step extrinsic return  $R_1^e = r^e(s', g) + \gamma U(g, sg, s')$ , and consequently  $\theta_h^e$  and  $\theta_l^e$  can be updated by Equation 3.3 and 3.4. In addition,  $\theta_t$  can be updated by Equation 3.5 as demonstrated by Bacon *et al.* (2017).

$$\theta_h^e \leftarrow \theta_h^e - \nabla_{\theta_h^e} [R_1^e - Q_{\theta_h^e}(s, g, sg)]^2. \quad (3.3)$$

$$\theta_l^e \leftarrow \theta_l^e - \nabla_{\theta_l^e} [R_1^e - Q_{\theta_l^e}(s, g, sg, a)]^2. \quad (3.4)$$

$$\theta_t \leftarrow \theta_t - \nabla_{\theta_t} term_{\theta_t}(s', g, sg)(Q_h^e(s', g, sg) - V_h^e(s', g)). \quad (3.5)$$

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  $\epsilon$ -greedy (Mnih *et al.*, 2015) exploration policy and collect the experience samples to effectively train  $\theta_h^e$ ,  $\theta_l^e$  and  $\theta_t$ . 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  $\theta_h^e$ ,  $\theta_l^e$  and  $\theta_t$ .

We hereby define a proxy low-level policy  $\pi_l^p(a|s, sg)$  aiming to achieve the proposed sub-goal  $sg$ . Similarly, we learn the proxy low-level policy by optimizing the discounted cumulative intrinsic rewards  $Q_l^i(s, sg, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}^i | s_0 = s, sg_0 = sg, a_0 = a]$ . We adopt the DQN method (Mnih *et al.*, 2015) to learn  $Q_l^i(s, sg, a)$  by updating its parameter  $\theta_l^i$  with Equation 3.6, where  $R_1^i = r^i(s', sg) + \gamma \max_a Q_l^i(s', sg, a)$  is the 1-step intrinsic return. As a result,  $\pi_l^p(a|s, sg) = \mathbf{1}(a = \operatorname{argmax}_a Q_l^i(s, sg, a))$ .

$$\theta_l^i \leftarrow \theta_l^i - \nabla_{\theta_l^i} [R_1^i - Q_{\theta_l^i}(s, sg, a)]^2. \quad (3.6)$$

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  $\pi_l^p(a|s, sg)$ , we introduce a hyper-parameter  $\alpha \in [0, 1]$  as the probability that the low-level layer adopts the proxy policy  $\pi_l^p(a|s, sg)$  to explore the environment

and collect the experience samples. The experience samples are used to batch train  $\theta_h^e$ ,  $\theta_l^e$ ,  $\theta_t$  and  $\theta_l^i$  with Equation 3.3, 3.4, 3.5 and 3.6 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  $\pi_l(a|s, g, sg)$  eventually.

### 3.3.4 Network Architecture

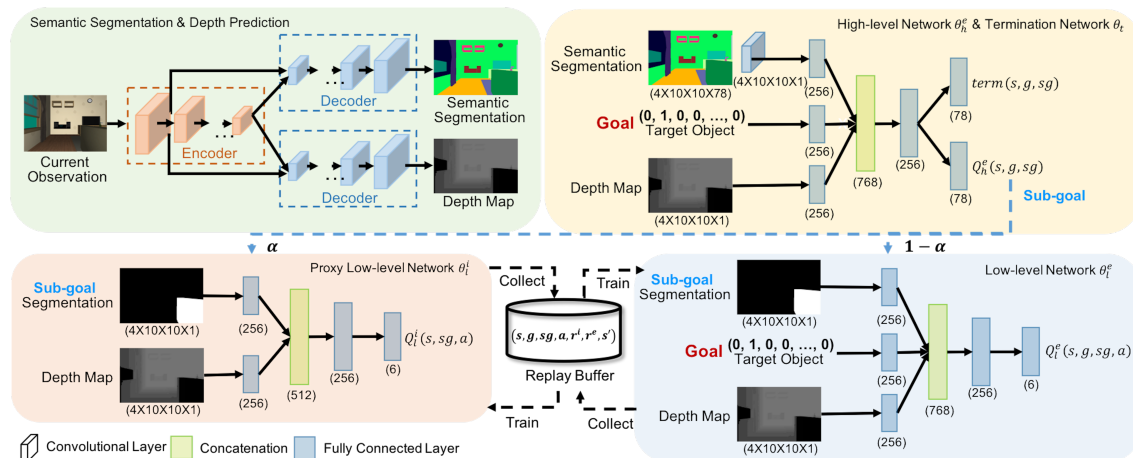


Figure 3.2: Network architecture of our hierarchical reinforcement learning model.

Since the image captured by the robot’s on-board camera serves as the robot’s current state, we adopt deep neural networks as  $\theta_h^e$ ,  $\theta_l^e$ ,  $\theta_t$  and  $\theta_l^i$  to handle the high dimensional inputs and approximate  $Q_h^e(s, g, sg)$ ,  $Q_l^e(s, g, sg, a)$ ,  $term(s, g, sg)$  and  $Q_l^i(s, sg, a)$ .

Fig. 3.2 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 (Ye *et al.*, 2019a) 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 (Mousavian *et al.*,



2019). The predicted results of the 4 history observations are fed into our high-level network  $\theta_h^e$  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  $\theta_h^e$  feeds the embedding fusion into one additional fully connected layer to approximate  $Q_h^e(s, g, sg)$ . To save the number of parameters, our termination network  $\theta_t$  shares most parameters with the high-level network  $\theta_h^e$  except the last fully connected layer where it adopts a new one to approximate  $term(s, g, sg)$ .

For the low-level network  $\theta_l^e$  and  $\theta_l^i$ , we take the sub-goal specified channel of the predicted semantic segmentation and the predicted depth map as the inputs. The low-level network  $\theta_l^e$  takes the one-hot vector of the target object as an additional input. Similar to our high-level network, each input of  $\theta_l^e$  and  $\theta_l^i$  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  $Q_l^e(s, g, sg, a)$  and  $Q_l^i(s, sg, a)$ .

We follow Equation 3.3, 3.4, 3.5 and 3.6 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.

## 3.4 Experiments

### 3.4.1 Dataset

We validate our framework on the simulation platform House3D (Wu *et al.*, 2018). 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 (Ye *et al.*, 2019a) (shown in Fig. 3.2 upper left) for semantic segmentation and depth prediction as we mentioned in Sec. 3.3.4. 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. 3.3.

### 3.4.2 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** (Mnih *et al.*, 2016). The vanilla A3C implementation that has been wildly adopted for the navigation task in the previous work (Zhu *et al.*, 2017; Ye *et al.*, 2018, 2019a; Yang *et al.*, 2018; Druon *et al.*, 2020). 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  $\theta_h^e$ .

**DQN** (Mnih *et al.*, 2015). The vanilla DQN implementation that adopts a similar network architecture as our high-level network  $\theta_h^e$  to predict the state action value  $Q^e(s, g, a)$ .

**OC** (Bacon *et al.*, 2017). 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** (Kulkarni *et al.*, 2016) with our proposed sub-goal space. It is equivalent to our method when we set  $term(s, g, sg) = 0$  and  $\alpha = 1$  to disable both the termination network  $\theta_t$  and the low-level network  $\theta_l^e$ .

**HIEM**. Our method follows Sec 3.3. 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  $\theta_l^i$ , **HIEM-low** sets  $\alpha = 1$  to disable the low-level network  $\theta_l^e$ , and **HIEM-term** sets  $term(s, g, sg) = 0$  to disable the termination network  $\theta_t$ .

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. To be specific, we adopt the same encoder-decoder network (Ye *et al.*, 2019a) to predict the semantic segmentation and the depth map from the raw RGB image of size  $600 \times 450$ . We further resize the both predictions to size  $10 \times 10$  and feed them to each method for policy learning. For DQN networks in the method DQN, h-DQN and HIEM, we adopt the Double DQN (Van Hasselt *et al.*, 2015) 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  $1 \times 10^{-4}$  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.

### 3.4.3 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:

$$\frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{\infty} \gamma^t r_{t+1}^e = \frac{1}{N} \sum_{i=1}^N \mathbf{1}(success) \gamma^{\#steps} * 1, \quad (3.7)$$

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  $\gamma$ , 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

Table 3.1: 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 $\uparrow$	AS / MS $\downarrow$	SPL $\uparrow$	AR $\uparrow$
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
<b>Ours</b>				
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	<b>1.00</b>	49.42 / 25.63	0.65	0.66
HIEM	<b>1.00</b>	<b>41.18 / 25.63</b>	<b>0.72</b>	<b>0.70</b>

over these cases (AS / MS). Success weighted by inverse Path Length (SPL) (Anderson *et al.*, 2018a), 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 adopt the number of the action steps as the path length. As a result, SPL also trades-off success rate against average steps.

Table 3.1 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 let the robot encounter the sparse 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.

Table 3.2: Average SPL achieved by all methods on 4 random environments.

Method	RANDOM	A3C	DQN	OC	H-DQN	<b>HIEM</b>
Avg SPL	0.03	0.03	0.35	0.03	0.11	<b>0.54</b>

We also report in Table 3.2 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. 3.3, which shows that our method yields a more concise and interpretable trajectory compare to other methods for the object search task.

### 3.5 Conclusion

In this chapter, 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

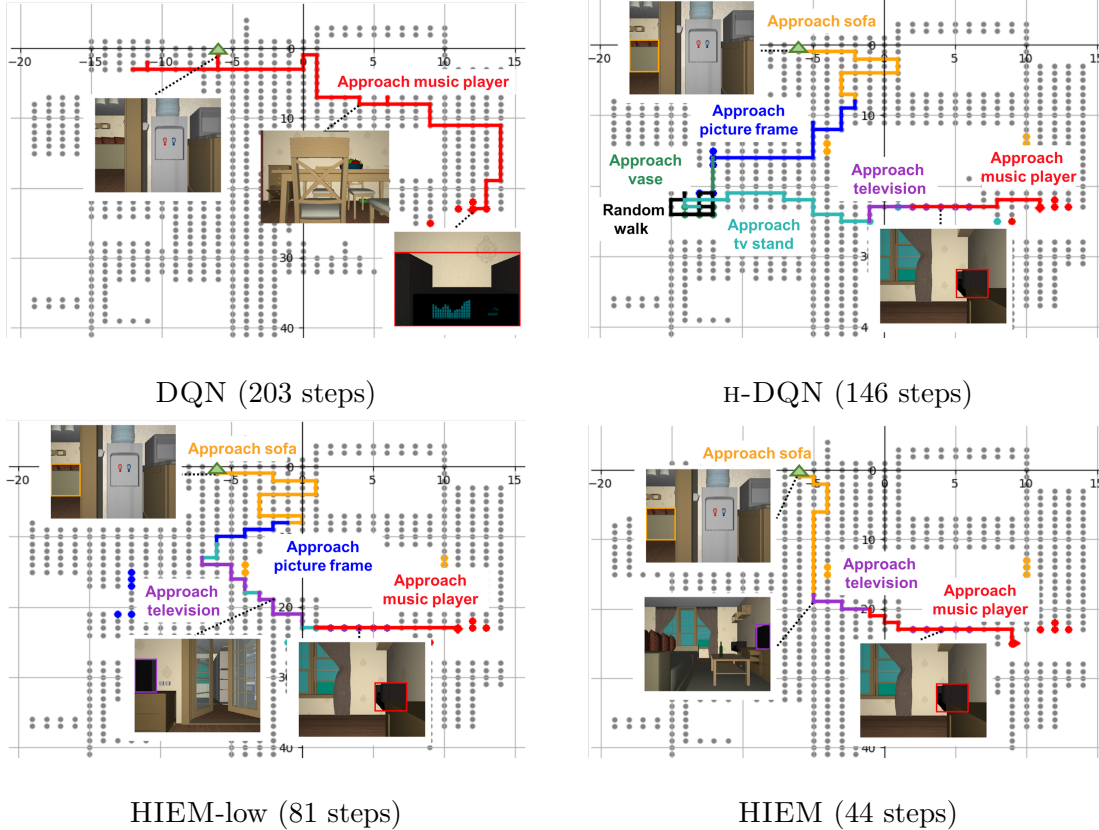


Figure 3.3: Trajectories generated by DQN (Mnih *et al.*, 2015), H-DQN (Kulkarni *et al.*, 2016) and our method HIEM-low and HIEM for searching the target object *music player* (red dots) from the same starting position (green triangle) which is 39 steps away. Our method HIEM generates a more concise and interpretable trajectory.

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 explore it in Chapter 5.

## GENERALIZABLE POLICY LEARNING FOR FULLY OBSERVABLE TASK

**4.1 Introduction**

With the current surge of deep reinforcement learning (Mnih *et al.*, 2015, 2013, 2016), a joint learning method of visual recognition and planning emerges as end-to-end learning (Zhu *et al.*, 2017; Ye *et al.*, 2018). Specifically, the robot learns an optimal action policy to reach a goal state by maximizing the reward it receives from the environment. Under the robotic object search task, the goal state is typically the location of the target object with a high reward assigned. Several recent work have attempted to fulfill the challenge and achieved certain promising results. For example, Zhu *et al.* (2017) adopted a target-driven deep reinforcement learning model to let robot find a specific visual scene. Ye *et al.* (2018) also proposed a recognition-guided deep reinforcement learning for robot to find a user-specified target object. Although these deep reinforcement learning models can be trained to navigate a robot to find a target scene or object in an environment, a time-consuming re-training process is needed every time the target or the environment alters. In other words, these systems suffer from an unsatisfiable generalization capability to transfer the previously learned action policy to a brand new target or a novel environment. Such defect extremely limits the applications of these methods in real-world scenarios as it is impractical to conduct the inefficient training process every single time.

We argue that the limitation is deeply rooted in the task itself. As the state observations typically come from the robot’s sensory inputs, they only capture the local information of its surrounding environments. Consequently, the goal states may

not be observable, yielding partially observable tasks that ask the robot to explore the environments and infer the whole state transition functions. To validate this argument, in this chapter, we start with a fully observable task where the goals are observable and we study generalizable policy learning for such a fully observable task.

For robotic object search task, while searching an object is indeed environment and object dependent, approaching an object after seen once should be a general capability. The insight could also be explained while observing human beings searching an object in a house. We first need to explore the house to locate the object once. After the object is captured with one sight, we are able to approach the target object with fairly few back and forth explorations. While the exploration policy varies a lot, the optimal approaching policy is indispensable, and provide a critical last step for a successful object search. Thus approaching policy is a much general capability of human beings, and thus in this chapter, we focus on the approaching policy learning. We define an approaching task as the robot is initialized in a state where the target object can be seen, and the goal is to take the minimal number of steps to approach the target object.

To tackle the challenge, we put forward a novel approach aiming at learning a generalizable approaching policy. We first treat a deep neural network as the policy approximator to map from visual signals to navigation actions, and adopt the deep reinforcement learning paradigm for model training. The trained model is expected to navigate the robot approaching a new target object in a new environment without any extra training effort. To learn an optimal action policy that can lead to a shortest path to approach the target object, previous methods typically attempts to map visual signal to an optimal action directly, no matter the signal contains clues towards reaching the goal state or not. In such a case, these methods inherently force the policy network to encode the local map information of the environment, which is

specific towards a certain scene. Thus, re-training or fine-tuning is needed to update the model parameters while facing a novel target object or a new environment.

Rather than learning a mapping from each visual signal directly to a navigation action, which has a much higher chance of encoding environment-dependent features, we present a method that first explicitly learns a general feature representations (scene depth and semantic segmentation map) to capture the task-relevant features solely from the visual signal. The representations serve as the input to the deep reinforcement learning model for training the action policy. To validate our proposed method’s ability to generalize the approaching behavior, empirical experiments are conducted on both simulator (House3D) and in a real-world scenario. We report the experimental results (a sharp increase of the generalization ability over baseline method) in Section 4.4.

## 4.2 Related Work

**Target-driven visual navigation.** Among plenty of methods for target-driven visual navigation, those ones with deep reinforcement learning are most relevant, as we will not provide any human guidance or map related information to the learning system. Recently, Mirowski *et al.* (2016) approached the target-driven deep reinforcement learning problem by jointly conducting depth prediction with other classification tasks. Zhu *et al.* (2017) proposed a target-driven framework to enable a robot to reach an image-specified target scene. Ye *et al.* (2018) introduced a recognition-guided paradigm for robot to find a target object in indoor environments. Both Das *et al.* (2018a) and Gordon *et al.* (2018) aim to let robot navigate in an indoor environment and collect necessary information to answer a question. Although these methods work well in their designed domain, the generalization ability towards new object and new environment is questionable.

**Generalization in deep reinforcement learning.** While generalization ability is a critical evaluation criteria in deep learning, it is less mentioned in the literature of deep reinforcement learning, where most of work focus on improving the training efficiency and the performance of the trained model in certain specific domains (Mnih *et al.*, 2013, 2015, 2016; Jaderberg *et al.*, 2016; Wang *et al.*, 2016; Ghosh *et al.*, 2017; Gu *et al.*, 2017). The authors of Dosovitskiy and Koltun (2016) proposed to predict the effect of different actions on future measurements, resulting in good generalization ability across environments and targets. However, it is based on the condition of training the model in the complex multi-texture environments. Pathak *et al.* (2017) adopted the error in predicting the consequences of robot’s actions as curiosity to encourage robot to explore the environment more efficiently. Yet it still needs a fine-tuning process when deploying the trained policy in a new environment.

**Semantic segmentation and depth prediction.** Semantic segmentation and depth prediction from a single image are two fundamental tasks in computer vision and have been extensively studied. Recently, convolutional neural network and deep learning based methods show dominating performance to both tasks (Liu *et al.*, 2009; Couprie *et al.*, 2013; Laina *et al.*, 2016; Chen *et al.*, 2018; Fu *et al.*, 2018). Instead of addressing them separately, Wang *et al.* (2015) proposed a unified framework for jointly predicting semantic and depth from a single image. Eigen and Fergus (2015) also adopted a single neural network to do semantic labeling, depth prediction and surface normal estimation. In work Jafari *et al.* (2017), the authors analyzed the cross-modality influences between semantic segmentation and depth prediction and then designed a network architecture to balance the cross-modality influences and achieve improved results. Despite the good performance these methods achieved, multi-step training process is still required, that leads to heavy computational load in learning and using these models. In this chapter, we adopt a *DeepLabv3+* (Chen

*et al.*, 2018) based model with which we can perform end-to-end training without a performance loss.

### 4.3 Our Approach

#### 4.3.1 Overview

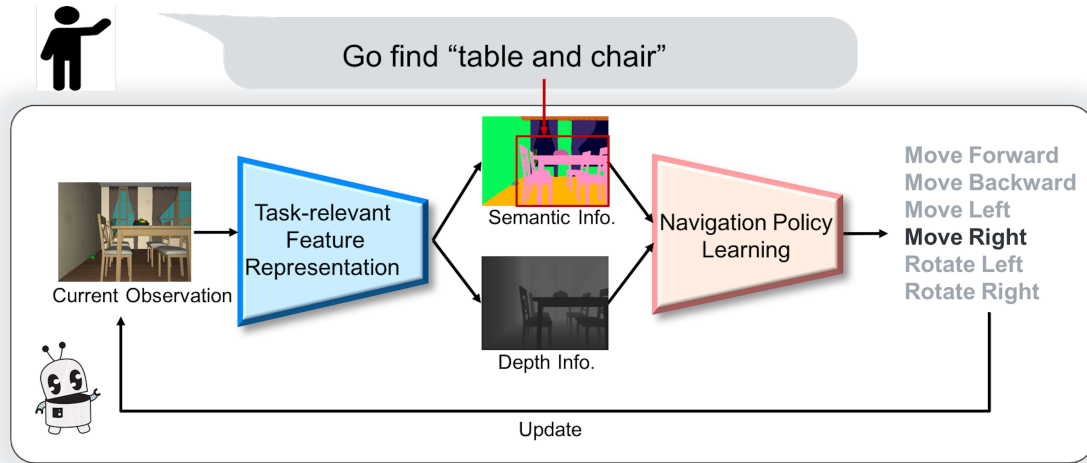


Figure 4.1: An overview of our GAPLE system.

We define the task as learning a generalizable action policy for a robot to approach a user-specified target object with minimal steps. The target object is specified as its semantic category. The robot’s on-board camera is the only sensor to capture RGB images, which serve as the robot’s observations. The robot starts at the location where the target object can be detected (fully observable task). With the current observation, the robot makes a decision upon which action to take. Afterwards, the robot receives a new observation and it repeats the decision process iteratively until it reaches a close enough location to the target object. Moreover, once the action policy is trained and deployed, the robot is expected to take reasonable number of steps to approach the user-specified target object as soon as the robot sees it, even the target

object is from a new category or the environment changes. The ground-truth map information for both training and testing environments is unknown.

Fig. 4.1 shows an overview of our system. Since the action decision depends on the robot’s current observation, the RGB image is the input to the system. Besides, to make the system be flexible to the appearance changes of the target object in the same semantic category, we further include its semantic label as part of the input.

To generalize well across various environments, the feature representations from the input image should be also general across all different environments, or so-called environment-independent. Although the deep neural network is well-suited for extracting task-relevant features (Donahue *et al.*, 2014), it tends to capture the environment-dependent features. For example, Zhu *et al.* (2017) also pointed out that a scene-specific layer is needed to capture the special characteristics like the room layouts. As a result, these models that integrally learns feature representation and navigation policies, can be easily over-fitted towards specific environments. To overcome this challenge, we propose to explicitly learn a more general feature representations. Consider our object approaching task as an example, the robot needs to capture the semantic information from its observation to identify the target object. At the same time, the depth information is crucial for the robot to navigate and avoid collisions. Thus, we adopt a feature representation module that captures both the semantic and the depth information from the input image. We further pipeline the outputs of our feature representation module as the inputs to our proposed navigation policy learning module for action policy training. The following sections introduce the feature representation module and the navigation policy learning module respectively.

### 4.3.2 Semantic Segmentation and Depth Prediction

We adopt the *DeepLabv3+* (Chen *et al.*, 2018) based model (as shown in Fig. 4.2) to jointly predict the semantic segmentation and depth map from a single RGB image. *DeepLabv3+* employs an encoder-decoder structure, where the encoder utilizes the spatial pyramid pooling to encode multi-scale contextual information. Specifically, it applies multiple filters or pooling operations that with different rates on the feature map computed by other pretrained models, such as ResNet-101 (He *et al.*, 2016) and Xception (Chollet, 2017). That allows the filters or the pooling operations to be able to consider different field-of-views, so that they can capture rich semantic information. During the decoding process, it gradually up-samples the encoder’s output and recovers the spatial information to capture the sharp object boundaries, which leads to better semantic segmentation results. We refer interested readers to Chen *et al.* (2018) for more details.

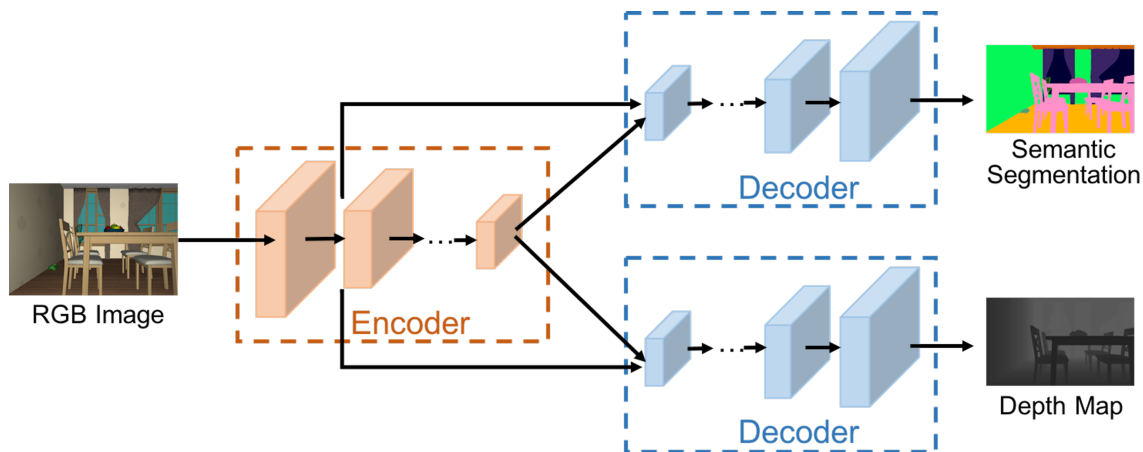


Figure 4.2: An illustration of the adopted model based on *DeepLabv3+* (Chen *et al.*, 2018) to predict semantic segmentation and depth map from a single RGB image.

For generating the depth map at the same time, we spawn another decoder branch. The motivation is that the depth information and semantic segmentation are corre-



lated. Either one can be used as a guidance to help predicting the other one according to Couprie *et al.* (2013) and Liu *et al.* (2009). Thus it is benefiting to jointly predict both of them (Wang *et al.*, 2015; Eigen and Fergus, 2015). Here, we adopt the exactly same architecture as the one for semantic segmentation except for the output layer. Rather than a classification layer that outputs labels for each corresponding pixel, we utilize a regression layer instead to predict depth value for each pixel.

$$\mathbf{L} = \frac{1}{N} \sum_i^N (-\mathbf{p}_i^* \log(\mathbf{p}_i)) + \lambda \frac{1}{N} \sum_i^N \|d_i - d_i^*\|_2^2 \quad (4.1)$$

Specifically, our system adopts the Xception-65 model (Chollet, 2017) pretrained on ImageNet (Russakovsky *et al.*, 2015) as initialization. We then define the loss function (Eq. 4.1) to train our model in an end-to-end manner. The first term is the cross entropy loss for semantic segmentation.  $\mathbf{p}_i^*$  is the one-hot encoded ground-truth semantic label for pixel  $i$ .  $\mathbf{p}_i$  is the corresponding predicted probabilities over all possible semantic labels. The second term is the mean-square error for depth prediction, where  $d_i^*$  denotes the ground truth depth value for pixel  $i$  and  $d_i$  represents the corresponding predicted depth value.  $N$  denotes the total number of pixels in the image and  $\lambda$  denotes a balancing factor. In practice,  $\lambda = 0.01$  achieves good performance empirically and we train our model by minimizing the loss function through the stochastic gradient decent (SGD) optimization.

### 4.3.3 Approaching Policy Learning

With the semantic segmentation and the depth information computed as the representations of the robot’s current observation, the robot is expected to make a decision of which action to take to approach the target object. Consider the challenge that the overall **state space** for robot is unknown and each state is of high dimension, we apply the deep reinforcement learning method.

First, we design a deep neural network as an estimator of the policy function. The **policy network** takes both semantic segmentation and depth information as inputs and outputs a probability distribution over all **valid actions** (also known as action policy). The robot picks a valid action either randomly or follows the distribution predicted by the policy network. After performing the action, the robot receives a **reward signal** as a measurement of how beneficial the performed action is towards the goal. This one-step reward (or likewise the accumulated rewards after taking multiple steps) serves as the weight factor of taking the performed action as the ground truth action for training. We further introduce each part of our setting in details here.

**State space:** Since we assume that the RGB image captured by robot’s camera is the only source of information, and both of the robot’s position and the target object’s location are unknown, the robot’s state can only be represented by the captured RGB image as well as the semantic label of the target object. As mentioned before, we represent the RGB image using semantic segmentation and depth map, and the semantic segmentation together with the semantic label of the target object can be further encoded as an attention mask. Afterwards, the attention mask and the depth map together represent the robot’s state (see left side of Fig. 4.3). In addition, the size of the attention field also encodes how close the robot to the target object. Thus, we set a threshold and set the goal states as those with an attention field larger than the threshold. In practice, we set it as the size of fifth largest attention field among all ground truth attention masks to yield five goal states. All the possible states form the state space.

**Action space:** To constrain the number of possible states, we only consider discrete actions. Without loss of generality, we consider some basic actions for the navigation purpose, namely “move forward”, “backward”, “left or right with a fixed

distance”, “rotate left or right with a constant angle”. In our experiments, we define a fixed distance as 0.2 meters and a constant angle as 90 degrees.

**Reward function:** We adopt the reward function designed by Ye *et al.* (2018) to avoid getting stuck in some suboptimal states. We define the reward as the one proportional to the size of the attention field if and only if the attention field is larger than all previous ones the robot has observed. Otherwise, the reward is set to be zero. Formally, the cumulative reward  $= \gamma^{i_1}ka_{i_1} + \gamma^{i_2}ka_{i_2} + \dots + \gamma^{i_t}ka_{i_t}$ , where  $k$  is a constant,  $a_i$  denotes the size of the attention field at time step  $i$ ,  $\gamma$  is the discount factor for time penalty and  $a_{i_1} < a_{i_2} < \dots < a_{i_t} (i_1 < i_2 < \dots < i_t)$ .

**Policy network:** Fig. 4.3 illustrates the overall policy learning architecture. The learning module takes the semantic segmentation and depth map as inputs. The semantic segmentation is then used to create an attention mask with the semantic label of the target object. We further resize both the attention mask and the depth map to the size of 10 by 10. For each input stream, we concatenate the features of 8 history frames. The two 800 dimensional vectors from two input streams are then concatenated into a 1600 dimensional joint vector before attaching a fully connected layer to generate a 512 dimensional embedding fusion. The embedding fusion is then fed into two separate branches. For both branches, we first attach a fully connected layer to project the 512 dimensional embedding fusion into a 20 dimensional vector. In the first branch, we further project the 20 dimensional vector into 6 action policy outputs (probability over actions) with an additional fully connected layer. In the second branch, we project the 20 dimensional vector into a single state value output, i.e. the expected cumulative reward the robot would receive at the current state.

We follow the training protocol from Zhu *et al.* (2017). It trains the model by running multiple threads in parallel with each thread takes one environment-target pair to train, and all threads update their weights to a global shared network asynchronously.

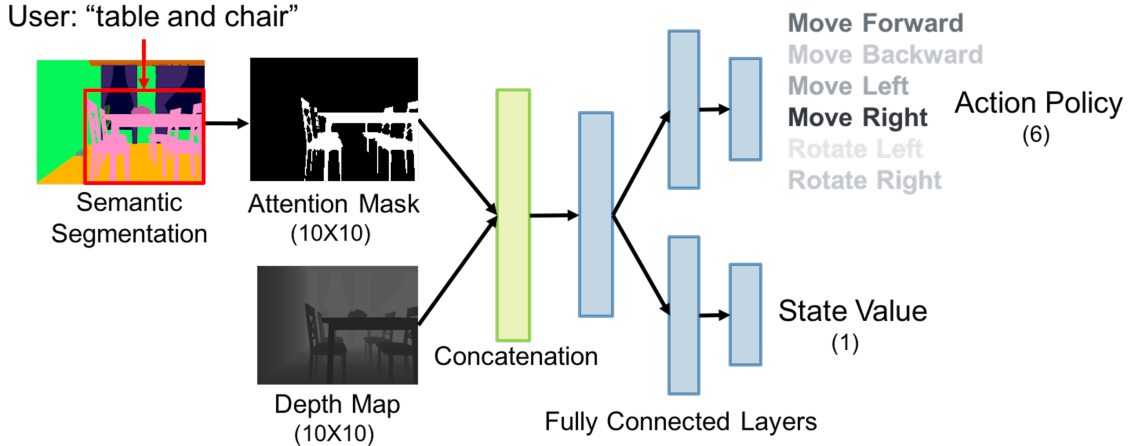


Figure 4.3: The architecture of our deep reinforcement learning model for action policy training.

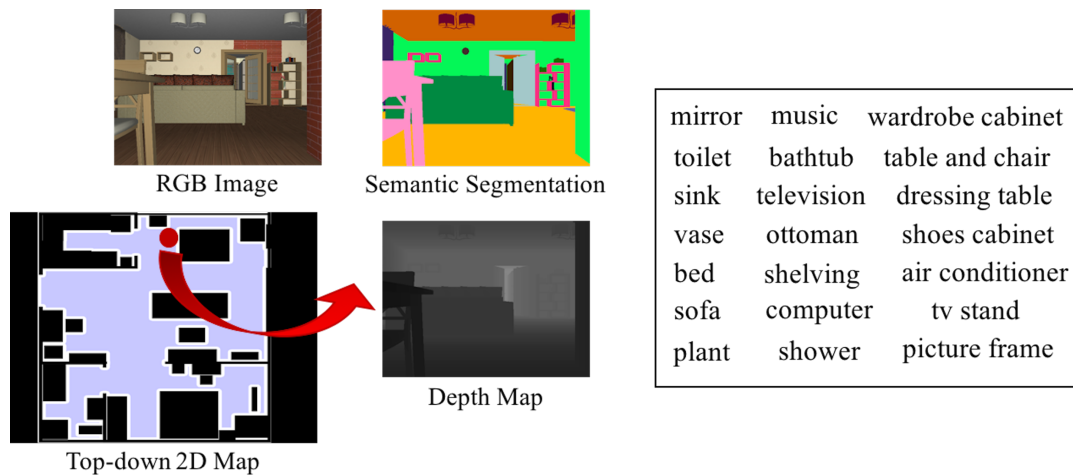
In case certain environment-target pairs are much easier to train that may cause the over-fitting problem, each thread chooses the less-trained environment-target pair before every training iteration to train all environment-target pairs equally (Chen *et al.*, 2007). We train the network with an RMSProp optimizer of learning rate  $1 \times 10^{-4}$ .

## 4.4 Experiments

### 4.4.1 Dataset

To train our model and test its generalization ability across different target objects and environments, we need a data efficient platform that has a diverse set of objects and environment types. Here, we adopt the publicly available simulation platform House3D (Wu *et al.*, 2018), and a renderer that builds on SUNCG dataset (Song *et al.*, 2013). Because House3D consists of a rich 3D indoor environments for a virtual robot to interact with. During the interaction, the robot has access to the first-person view RGB images, as well as the corresponding ground truth semantic segmentations and depth maps, which makes it well suited to the feature representation learning task

and the approaching policy learning task. Fig. 4.4 (a) depicts an example data.



(a) A sample environment

(b) Target object candidates

Figure 4.4: A sample environment from House3D and some target object candidates.

We constrain the robot to perform discrete actions in these virtual environments, i.e. moving 0.2 meters or rotating 90 degrees every time. It also discretizes the environment into a set of reachable locations. We select a total of 248 simulated environments that are suitable for testing. Additionally, to avoid ambiguity, we select the objects that only have one instance in an environment as the target objects for robot to approach. Fig. 4.4 (b) lists example target objects used.

#### 4.4.2 Semantic Segmentation and Depth Prediction

In order to train our feature representation module for semantic segmentation and depth prediction, we collect RGB images, as well as their corresponding ground truth captured at all discrete locations from 100 environments. We further delete the images that has over 80% background and randomly sample a total of 55,697 images for training. For semantic segmentation, 77 semantic labels are of our interest, with

all the remaining ones being classified as “background”.

We take the popularly used metrics, a.k.a. mean Intersection Over Union (mean IOU) and Root Mean Square Error (RMSE) to report the performance of our trained models in doing semantic segmentation and depth prediction respectively. Our model achieves 0.436 in mean IOU for semantic segmentation on validation dataset, and 0.0003 normalized RMSE for depth prediction. Fig. 4.5 shows several qualitative results.

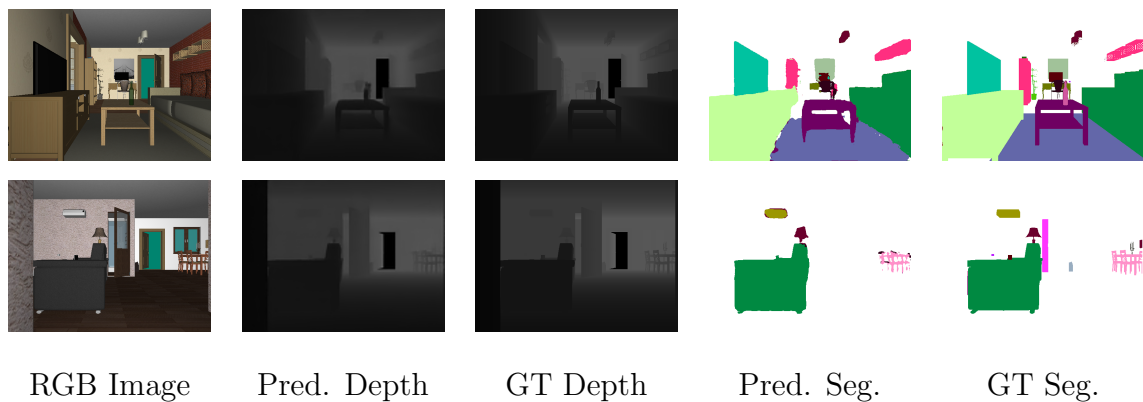


Figure 4.5: Some qualitative semantic segmentations and depth predictions from our feature representation module.

### 4.4.3 Approaching Policy Learning

To demonstrate the generalization ability of our proposed method across both target objects and environments, we compare our method with the following baselines and variants. Again, the map of the environment is unknown to all methods, except when calculating the minimal steps that robot needs to take to approach the target object.

**a)** Random method. At each state, the robot randomly choose an action to perform. Since the map is unknown, the action might yield collision. In that case,

the robot will simply stuck in the current state. The random method provides a performance lower-limit, which could calibrate how “intelligent” the other trained models are.

**b)** Method from Ye *et al.* (2018). It takes the output from the *res4f* layer of ResNet-50 network that pre-trained on ImageNet as the feature representation from both the target object and the robot’s current observation. The two channels of feature representations, as well as a binary attention mask that is generated using an object recognition module form the inputs to the deep reinforcement learning model. Here, we first use the ground truth attention mask to remove the influence of the noisy object recognition module. Then we test the method with the attention mask generated from our predicted semantic segmentation. Moreover, we adopt a single scene-specific branch for all the target objects and environments, and the trained model will be applied to unseen environments without extra training during the testing time.

**c)** Our method with ground truth semantic segmentation and depth map. We take the ground truth semantic segmentation and depth map as the inputs to our deep reinforcement learning model as described in Sec. 4.3.3, for the purpose of testing the performance upper-limit.

**d)** Our method with ground truth semantic segmentation and predicted depth map. This method is used to compare with the method b) as they both adopt ground truth semantic info to generate noise-free attention mask.

**e)** Our proposed method described in Sec. 4.3 that takes only an RGB image and the semantic label of the target object as the inputs, and outputs navigation actions.

We train and evaluate all the methods under two settings for object-wise generalization and environment-wise generalization respectively. **Setting 1):** training on 6 different target objects in 1 environment. The models trained on this setting

are then used to evaluate their generalization abilities across target objects. To be specific, during the testing, we use the trained models to approach 5 unseen target objects in the same environment and report their performances respectively. **Setting 2**): training on a total of 24 target objects in 4 different environments (6 each). We then test the trained models’ performances in approaching another 24 target objects in 4 novel and unseen environments. For both settings, the robot always starts at the position where the target object can be detected for both the training and the testing phases.

To conduct fair comparisons, for each testing environment-object pair, we randomly select 100 starting positions where the target object can be detected. The robot stops either it reaches close enough to the target object (a successful case) or it reaches 1000 steps (a failure case). We take two metrics to compare these methods, namely the success rate in terms of how many steps (relative to the minimal steps) are taken, and the average steps over the successful cases. Generally speaking, a higher success rate or a smaller number of average steps indicates a better approaching performance.

We trained each of these models with an Nvidia V100 (6 cards with 16g memory each) machine. For setting 1, each model’s training takes about 20 hours. For setting 2, the training takes about 40 hours to converge.

Fig. 4.6, Table 4.1 and Table 4.2 report the achieved success rate, success rate drop from trained objects/environments to new objects/environments, and average steps of all methods on the two settings respectively. From Fig. 4.6 and Table 4.1, the results indicate a clear generalization capability improvement of our method d) comparing with the method b) that both take ground truth attention mask and our method e) with the method b) that both take predicted attention mask. At the same time, the results align well with our expectation that our method with the predicted



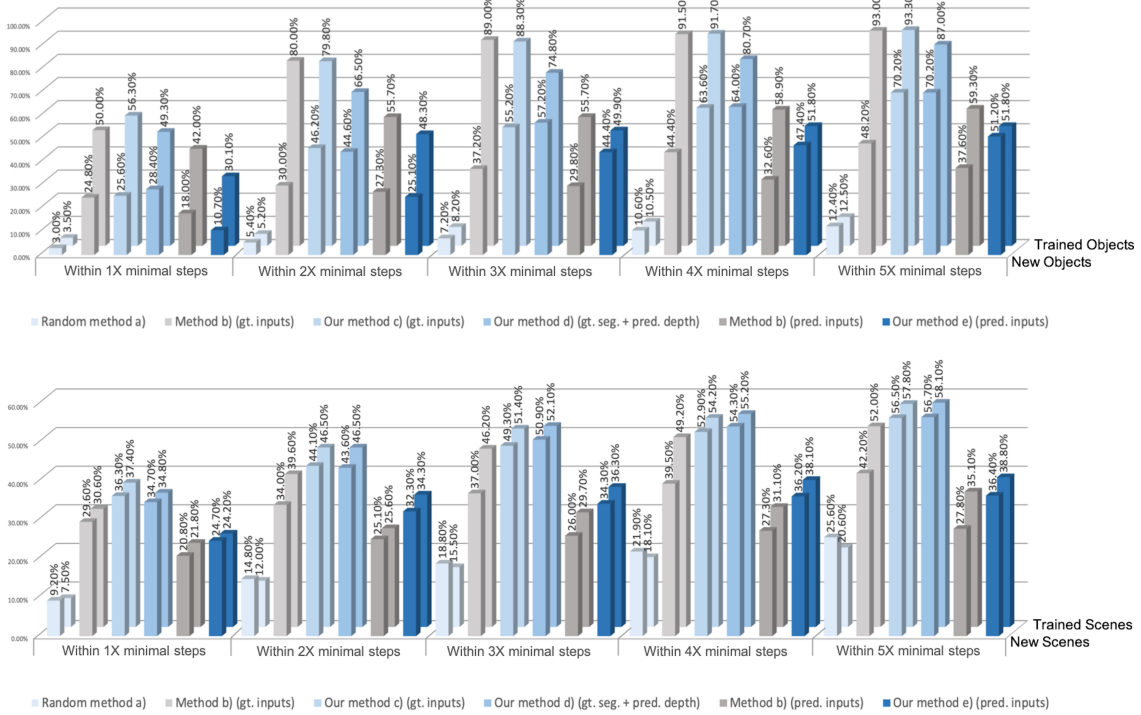


Figure 4.6: Successful approaching rates. Upper: Setting 1: generalization ability across target objects (on trained objects and on new objects). Lower: Setting 2: generalization ability across environments (on trained environments and on new environments).

semantic segmentation (method e)) performs worse than the method d), which also happens for method b). The reason is due to the recognition errors introduced from the predicted semantic segmentation that distracts the robot from approaching the target object. More specifically, the reward generated upon the area of the target object is not consistent due to the noisy detection. Moreover, the area of the robot’s attention (focusing on the target object) also needs to encode the goal states. With the noisy predicted semantic labels, the robot has a high likelihood to get stuck while it struggles to identify the correct goal states.

From Fig. 4.6, we also observe that method b) that with ground truth or predicted

Table 4.1: The success rate drop from the trained objects to new objects (setting 1): s1), and from the trained environments to new environments (setting 2): s2).

$\Delta$ success rate	1X minimal steps		3X minimal steps		5X minimal steps	
	s1	s2	s1	s2	s1	s2
Random a)	0.50%	-1.70%	1.00%	-3.30%	0.10%	-5.00%
b) (gt. inputs)	25.20%	1.00%	51.80%	9.20%	44.80%	9.80%
Ours c) (gt. inputs)	30.70%	1.10%	33.10%	2.10%	23.10%	1.30%
Ours d) (gt. s. + pred. d.)	20.90%	0.10%	17.60%	1.20%	16.80%	1.40%
b) (pred. inputs)	24.00%	1.00%	25.90%	3.70%	21.70%	7.30%
Ours e) (pred. inputs)	19.40%	-0.50%	5.50%	2.00%	0.60%	2.40%

attention mask achieves higher success rate than our method d) or e) respectively on trained objects under Setting 1). However, on new objects under Setting 1) and under Setting 2) where each model is trained with multiple different environments, method b) performs worse than our method, indicating that method b) may capture scene-specific features (pixel-based ResNet-50 features) and easily overfits to specific environments and objects. To further validate the overfitting concern is not solely due to the target object image input, we conduct experiments with blocking the target object input channel of method b), and we report the results in Table 4.3. The results indicate that the modified method b) remains to have concerns on generalizing well.

Moreover, comparing with our method c) that takes ground truth depth maps as inputs, our method d) with predicted depth maps achieves lower success rate especially under Setting 1). It is reasonable because the noisy depth maps may lead the robot to collisions or miss optimal actions. Nevertheless, as the results under Setting 2) indicate, such problem can be alleviated when the model is trained in

Table 4.2: Average number of steps taken by all methods on two settings.

methods	setting (1)		setting (2)	
	trained obj.	new obj.	trained env.	new env.
minimal	3.88	3.89	2.58	2.06
Random a)	213.71	202.15	166.62	109.85
Method b) gt.	5.88	7.85	4.34	3.64
Method b) pred.	4.70	5.82	6.15	5.94
Our method c)	5.85	13.99	6.14	5.14
Our method d)	8.45	13.43	7.96	5.96
Our method e)	10.31	16.44	3.77	4.22

Table 4.3: Success rate of method b) where the target object channel is blocked (with predicted inputs and within 5X minimal steps).

	trained obj. / env.	new obj. / env.	$\Delta$ success rate
setting (1)	63.80%	25.40%	38.40%
setting (2)	42.36%	27.60%	14.76%

multiple environments.

Table 4.2 reports the average number of steps taken among all successful trails. With the success rate reported in Fig. 4.6, it also matches our expectation that the average number of steps from our methods are generally larger than the method b). Here, the larger number of steps means that our methods also succeed in approaching the target object which needs a larger number of steps, while the method b) fails

these cases and they don't contribute to the average number of steps.

#### 4.4.4 Real World Experiment

We train and test our method on Active Vision Dataset (AVD) (Ammirato *et al.*, 2017). AVD is a set of real environments where a virtual robot can navigate in through discrete actions, i.e., move forward or backward, left or right 30 centimeters, rotate left or right 30 degrees. We follow the settings as introduced in Sec.4.4.3 to train and test our method (method e)), and report the quantitative results in Table 4.4. The experimental results on AVD further validate that our method e) achieves high generalization ability across both target objects and environments.

Table 4.4: Experimental results of our method (method e)) on AVD (Ammirato *et al.*, 2017).

metrics	setting (1)		setting (2)	
	trained obj.	new obj.	trained env.	new env.
success rate	68.00%	58.40%	72.60%	62.00%
average steps	37.86	45.23	33.19	44.39

To better visualize how our method performs, we also apply our method (method d)) in a real-world scenario (on a public dataset from Ye *et al.* (2018)). Without further fine-tuning the trained model, our trained model can still guide the robot to approach the target object. For this real-world experiment, we use the trained model from Laina *et al.* (2016) to predict the depth map and the ground truth bounding box to generate the attention mask. Fig. 4.7 shows an example of how the robot approaches the target object, which is a “whiteboard”.

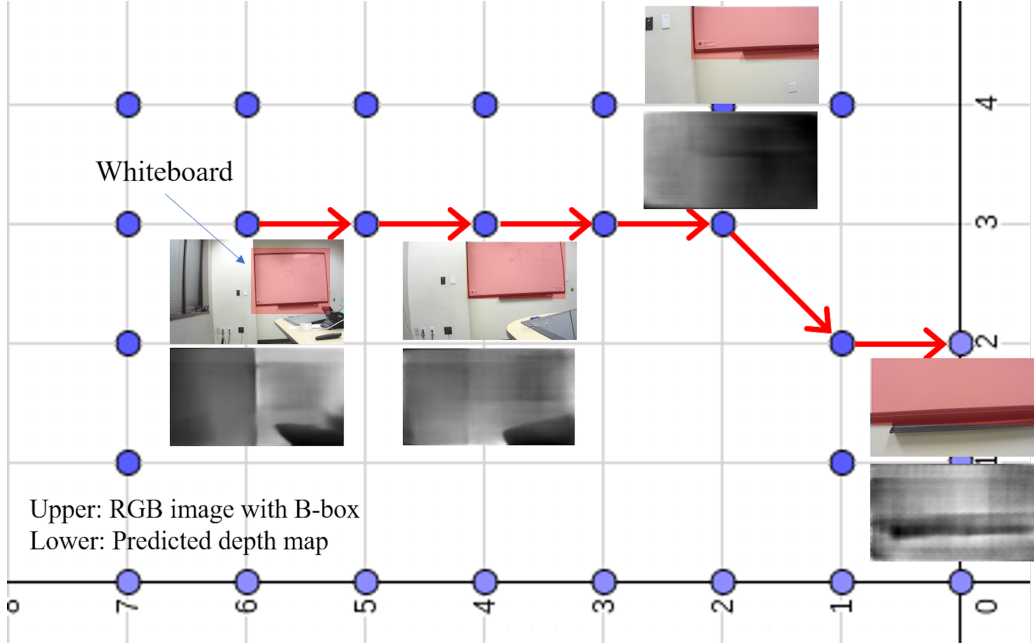


Figure 4.7: An example of the mobile robot approaches the target object “whiteboard” using the method (d). Upper view: RGB input; Lower view: Depth map generated.

#### 4.4.5 Analysis and Discussion

To better understand why our proposed method achieves better generalization ability, we further analyze from the feature representation perspective. Generally speaking, the goal of the deep reinforcement learning is to learn a policy model  $\pi$  that maps a state  $s$  to the most “beneficial” action  $a$  (or an action distribution from which the most “beneficial” action can be drawn with a high probability), i.e. to let  $\pi(s) = a$ . This most “beneficial” action  $a$ , unlike the ground truth label in a general supervised learning problem, is acquired by the intelligent agent’s trial and error interactions with the environment.

For the object approaching task, the most “beneficial” action  $a$  essentially depends on the local map between the current location and the goal location. In order to let

$\pi(s) = a$ , if the input  $s$  doesn't provide any map information directly (such as the setting from Ye *et al.* (2018)), then the model  $\pi$  has to capture such information from the input  $s$  through learning. To avoid the over-fitting problem, it is necessary to train the model  $\pi$  on a large enough and diverse enough training data where the underlying distribution of the relations between the state  $s$  and the map information can be captured. Though it is straightforward, the well-known sample-inefficient issue lingering in the paradigm of deep reinforcement learning makes it fairly impracticable.

In this work, we first adopt a feature representation model  $f$  to learn the semantic segmentation and depth map from the input state  $s$ , then we take the semantic segmentation and the depth map as the inputs to the policy model  $\pi$ . In other words, we aim to let  $\pi(f(s)) = a$ . Here, we hypothesize that the depth map as an input to the policy network  $\pi$  encodes the local map well already. In such a way, the policy model  $\pi$  is not the only source for capturing the local map information well. At the same time, the feature representation model  $f$  directly learns the depth map from the state  $s$  in a supervised manner, which is much more sample efficient.

For further validation, we examine the relationship between the distance in physical space and the one in the feature space. For each pair of the locations in an environment, we calculate their Manhattan distance in terms of steps as the physical distance. We adopt  $L_1$  distance between the normalized feature maps of the images taken at the two locations with the same orientation as the feature distance. Fig. 4.8 shows the relations between the physical distance and the distance in both depth feature space and ResNet-50 feature space.

From Fig. 4.8, it shows that within a small range of physical distances (1 to 9 steps), the distance in the depth feature space increases notably along with the increment of the physical distance. While the physical distance is out of this range (over 9 steps), the feature distance shows minor changes. This observation suggests

that the depth feature captures the differences between different locations within a small region, which aligns well with our assumption. On the other hand, the distance in ResNet-50 feature space grows almost independently w.r.t. the growing of the physical distance. We speculate that this observation provides the actual reason why methods (such as Ye *et al.* (2018)) fails to generalize well.

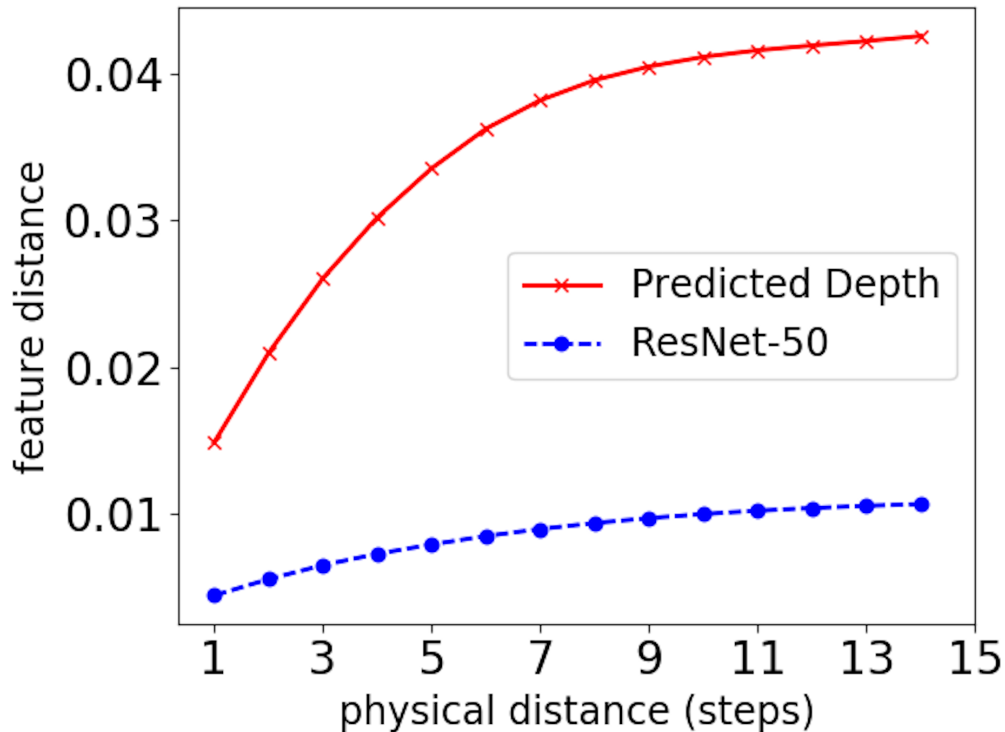


Figure 4.8: Pair-wise feature distances w.r.t. physical distances.

## 4.5 Conclusion

This chapter presents a generalizable policy learning for robotic object approach task which is a fully observable task, through explicit depth estimation and semantic segmentation. Empirical studies on the House3D platform and a real physical experiment on a mobile robot validate that the new framework can yield a significantly

higher generalization capability towards new target objects and novel environments, indicating a promising pathway for future research on achieving generalizable policy learning for partially observable task, such as robotic object search on mobile robots.



GENERALIZABLE POLICY LEARNING FOR PARTIALLY OBSERVABLE  
TASK**5.1 Introduction**

Classic RL methodology optimizes an agent’s decision-making action policy in a given environment (Sutton and Barto, 2018). To make RL towards real-world applicable, equipped with deep neural networks, Deep Reinforcement Learning (DRL) (Mnih *et al.*, 2015) 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 (Mnih *et al.*, 2015, 2016; Lillicrap *et al.*, 2015; Schulman *et al.*, 2017; Haarnoja *et al.*, 2018).

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 (Mnih *et al.*, 2015, 2016; Haarnoja *et al.*, 2018; Nasiriany *et al.*, 2019) or a partially observable task (Igl *et al.*, 2018; Lee *et al.*, 2019; Han *et al.*, 2019). 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

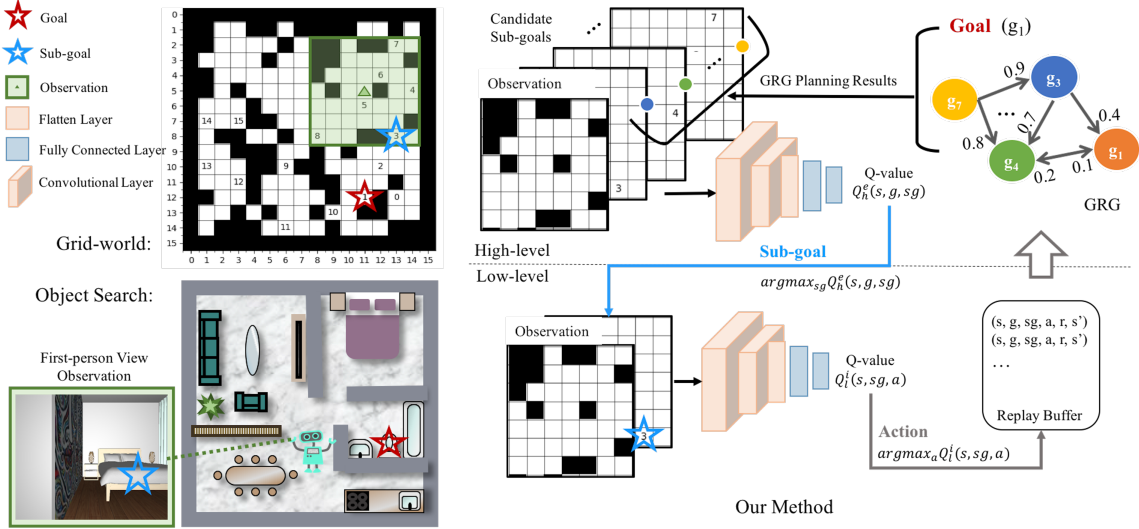


Figure 5.1: Illustrations of the grid-world domain and the robotic object search task (left), and an overview of our method (right).

observations  $\Omega$  and the goal descriptions  $G_d$ . This is challenging due to 1) the large augmented partially observable state space, 2) the different modalities that the observations  $\Omega$  and the goal descriptions  $G_d$  could have. For example, while RGB images are usually taken as the observations, semantic labels are more efficient in describing task goals (Batra *et al.*, 2020).

To address the challenges, we present a novel Hierarchical Reinforcement Learning approach with a Goals Relational Graph (HRL-GRG). 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 (Tu, 2014) 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 (Mnih *et al.*, 2015, 2016; Haarnoja *et al.*, 2018).

Many prior DRL methods tackling partially observable tasks (Igl *et al.*, 2018; Lee *et al.*, 2019; Han *et al.*, 2019) 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 (Zhu *et al.*, 2017), robotic object search (Ye *et al.*, 2018; Yang *et al.*, 2018; Batra *et al.*, 2020) and room navigation (Ye *et al.*, 2018). 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 5.2). Here, we argue and show our novel GRG modeling fills the gap.

Formally, we define GRG as a complete weighted directed graph  $\langle V, E, W \rangle$  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  $\tau^*$  from the sub-goal to the goal over the GRG; 2) terminating the low-level layer referring to the optimal plan  $\tau^*$  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 (Kolve *et al.*, 2017b) and the House3D (Wu *et al.*, 2018) benchmark environments. We show HRL-GRG model exhibits superior performance in both experiments over other baseline approaches, with extensive ablation analysis.

## 5.2 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 Zhu *et al.* (2017), 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 Wu *et al.* (2019a). Savinov *et al.* (2018) 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 Gupta *et al.* (2017) 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 (Anderson *et al.*, 2018b; Fried *et al.*, 2018; Wang *et al.*, 2019). 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 (Mousavian *et al.*, 2019; Yang *et al.*, 2018; Nguyen *et al.*, 2019; Qiu *et al.*, 2020; Batra *et al.*, 2020; Chaplot *et al.*, 2020; Ye and Yang, 2021a). 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 Yang *et al.* (2018), Nguyen *et al.* (2019) and Qiu *et al.* (2020), the authors extract object relations from the Visual Genome (Krishna *et al.*, 2017) corpus and incorporate this prior into their models through Graph Convolutional Networks (Kipf and Welling, 2016). 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 Wu *et al.* (2019b) 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 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 policies 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.

### 5.3 Hierarchical RL with GRG

#### 5.3.1 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 only on its partial observations. Without loss of generality, we represent the observations  $\Omega$  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 with no access to the global environment information (see Figure 5.1 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 5.1 (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  $VG = \{vg \mid vg \in G_d \text{ and } \Phi(o_t, vg) > 0\}$ . 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  $N_{max}^l$ . 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  $N_{max}$ .

### 5.3.2 Goals Relational Graph (GRG)

**GRG representation.** We formulate GRG as a complete weighted directed graph  $\langle V, E, W \rangle$  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  $\Phi$  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  $N_{max}^l$  steps to achieve the goal  $g_i$ . It introduces the following  $N_{max}^l + 1$  events that  $X_{ij}$  may take:

- Event  $k$  ( $1 \leq k \leq N_{max}^l$ ): 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  $N_{max}^l + 1$ : the goal  $g_j$  doesn't appear. We quantify this event as  $x_{ij,N_{max}^l+1} = 0$ .

It is fair to assume that  $X_{ij} \sim Cat(\boldsymbol{\theta}_{ij})$  in which the parameter  $\boldsymbol{\theta}_{ij} = (\theta_{ij,1}, \theta_{ij,2}, \dots, \theta_{ij,N_{max}^l+1}) \sim Dir(\boldsymbol{\alpha}_{ij})$  is a learnable Dirichlet prior.  $\boldsymbol{\alpha}_{ij} = (\alpha_{ij,1}, \alpha_{ij,2}, \dots, \alpha_{ij,N_{max}^l+1})$  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  $\mathbf{c}_{ij} = (c_{ij,1}, c_{ij,2}, \dots, c_{ij,N_{max}^l+1})$ . Since the Dirichlet distribution is the conjugate prior distribution of the categorical distribution, the posterior distribution of the parameter  $\boldsymbol{\theta}_{ij}$ , namely  $\boldsymbol{\theta}_{ij}|\mathcal{D} \sim Dir(\boldsymbol{\alpha}_{ij} + \mathbf{c}_{ij}) = Dir(\alpha_{ij,1} + c_{ij,1}, \alpha_{ij,2} + c_{ij,2}, \dots, \alpha_{ij,N_{max}^l+1} + c_{ij,N_{max}^l+1})$ . As a result, the posterior prediction distribution of a new observation  $P(X_{ij} = x_{ij,k}|\mathcal{D})$  can be estimated by Equation 5.1, and the weight  $w_{ij} = E(X_{ij}|\mathcal{D}) = \sum_k x_{ij,k}P(X_{ij} = x_{ij,k}|\mathcal{D})$ . Here, the time complexity for updating our GRG is linear to the number of the samples collected by our low-level network. In addition, since the samples are also used for learning the high-level network and/or the low-level network, our GRG doesn't introduce any extra space complexity comparing to the



RL counterparts.

$$P(X_{ij} = x_{ij,k} | \mathcal{D}) = \mathbb{E}[\theta_{ij,k} | \mathcal{D}] = \frac{\alpha_{ij,k} + c_{ij,k}}{\sum_k (\alpha_{ij,k} + c_{ij,k})}. \quad (5.1)$$

**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  $\tau_{i,j} = \{\tau_1, \tau_2, \dots, \tau_M\}$  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  $\tau_{i,j}^* = \operatorname{argmax}_{\tau_{i,j}} \prod_{m=1}^{M-1} w_{\tau_m \tau_{m+1}}$ . 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  $g_j$ .

### 5.3.3 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.  $r_t^e(s_t, g) = 1$  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  $Q_h^e(s, g, sg) = \mathbb{E}[\sum_t \gamma^t r_{t+1}^e | s_t = s, g = g, sg_t = sg]$ , 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  $Q_h^e(s, g, sg)$ , we adopt the Q-learning technique (Mnih *et al.*, 2015) to update the parameters of the high-level network  $\theta_h$  by Equation 5.2, where  $R_1^e = r^e(s, a, s', g) + \gamma \max_{sg'} Q_h^e(s', g, sg')$  is the 1-step extrinsic return. The sub-goal  $sg$  is given by  $\operatorname{argmax}_{sg} Q_h^e(s, g, sg)$  towards achieving the final goal  $g$ .

$$\theta_h \leftarrow \theta_h - \nabla_{\theta_h} (R_1^e - Q_{\theta_h}(s, g, sg))^2. \quad (5.2)$$

**Network architecture.** To approximate  $Q_h^e(s, g, sg)$ , 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  $o_t$ .<sup>1</sup> 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  $Q_h^e(s, g, sg)$  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 Section 5.3.2. 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  $Q_h^e(s, g, sg)$ . In such a way,  $Q_h^e(s, g, sg) = Q_h^e(s, sg \odot \mathbb{C}(\tau_{sg,g}^*))$  where the goal  $g$  is embedded with benefi-

---

<sup>1</sup>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.

cial 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.

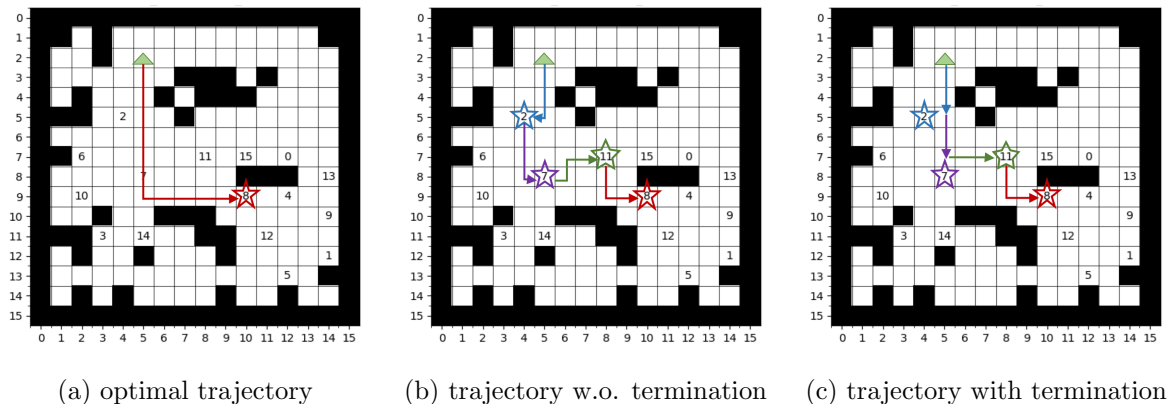


Figure 5.2: An illustration of how termination helps. The green triangle denotes the starting position. The stars and the arrows with different colors represent different sub-goals and the corresponding sub-goal-oriented trajectories. Termination helps to express an optimal trajectory with the limited sub-goal space.

### 5.3.4 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  $r_t^i(s_t, sg) = 1$  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  $\mathbb{E}[\sum_t \gamma^t r_{t+1}^i | s_t, sg_t, a_t]$ . 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 (Mnih *et al.*, 2015, 2016; Haarnoja *et al.*, 2018).

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 (Levy *et al.*, 2017a; Nachum *et al.*, 2018b). 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 5.2 (a) shows. Instead, following the set of available sub-goals proposed could yield a less optimal trajectory as Figure 5.2 (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 5.2 (c)). Some prior methods explore modeling a termination function in their formulations (Bacon *et al.*, 2017) or adding a special “stop” action in the action space for an optimal stop policy (Yang *et al.*, 2018). 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 5.3.2. 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. There-

fore, 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 (Mnih *et al.*, 2015) and A3C (Mnih *et al.*, 2016), without special requirements on the network architecture.

## 5.4 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 (Kolve *et al.*, 2017b) and House3D (Wu *et al.*, 2018) environments for the robotic object search task.

### 5.4.1 Grid-world Domain

**Grid-world generation.** We generate a total of 120 grid-world maps of size  $16 \times 16$  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  $7 \times 7$  centered at goal  $g_{i-1}$ . Figure 5.1 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.

Table 5.1: The performance of DQN vs DQN\_ONEHOT and DQN\_FULL on the unseen grid-world maps.

Method	Seen Goals		Unseen Goals		Overall	
	SR↑	SPL↑	SR↑	SPL↑	SR↑	SPL↑
DQN	0.20	0.13	0.20	0.15	0.32	0.23
DQN_ONEHOT	0.03	0.00	0.05	0.02	0.03	0.01
DQN_FULL	0.01	0.00	0.05	0.03	0.05	0.03

**Baseline methods.** We assume the agent can only observe the window of size  $7 \times 7$  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 (Mnih *et al.*, 2015) 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 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 (DQN\_ONEHOT) or full observations of all goal

positions (DQN\_FULL) (see Table 5.1).

- **H-DQN.** It is a widely adopted hierarchical method (Kulkarni *et al.*, 2016) 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).

**Training details.** For all the networks, we adopt the Double DQN (Van Hasselt *et al.*, 2016) technique and we train all the methods on the 100 training grid-world maps to achieve the 12 goals ( $g_0, g_1, g_3, g_4, g_6, g_7, g_8, g_9, g_{11}, g_{12}, g_{14}, g_{15}$ ). We first train the method DQN to achieve the goal from where the goal is observable and we take the model as the pre-trained model for the DQN and the low-level networks of both H-DQN and our method. For all the methods, we adopt the curriculum training paradigm. To be specific, at episode  $i < 10000$ , we start the agent at a position that is randomly selected from the top  $(i + 10)/100$  % positions closest to the goal position, and when  $i \geq 10000$ , we start the agent at a random position. All the hyperparameters are summarized in Table 5.2. 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 20 hours. We evaluate all the methods on the 20 testing grid-world maps over 5 different random seeds, namely 1, 5, 13, 45 and 99.

**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

Table 5.2: Hyperparameters of all the methods for the grid-world domain.

Hyperparameter	Description	Value
$\gamma$	Discount factor	0.99
lr	Learning rate for all networks (high-level/low-level)	0.0001
main_update	Interval of updating all the main networks of Double DQN	10
target_update	Interval of updating all the target networks of Double DQN	10000
batch_size	Batch size for training all the networks	64
epsilon	Initial exploration rate, anneal episodes, final exploration rate	1, 10000, 0.1
max_episodes	Maximum episodes to train each method	100000
$N_{max}^l$	The maximum steps that low-level network can take if applied	10
$N_{max}$	The maximum steps that each method can take	100
optimizer	Optimizer for all the networks	RMSProp
$\alpha_{ij}$	The hyperparameter of our GRG ( $\alpha_{ij,1}, \dots, \alpha_{ij,10}, \alpha_{ij,11}$ )	(0, ..., 0, 1)

Table 5.3: The performance of all methods on the unseen grid-world maps.

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

over all successful cases compared to the Minimal Steps over these cases (AS / MS), and the Success weighted by inverse Path Length (SPL) following Anderson *et al.* (2018a) 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 5.3 reports the results.



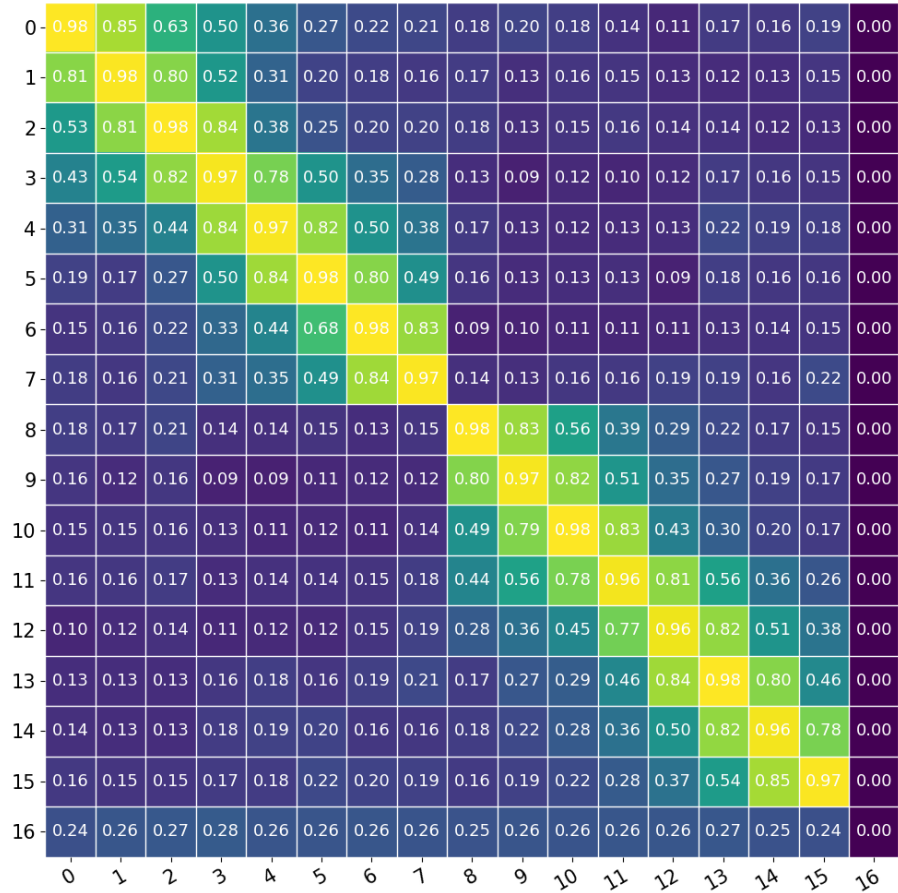


Figure 5.3: A visualization of a GRG learned on the grid-world domain ( $g_{16}$  is the back-up “random” goal).

As is shown in Table 5.3, 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 5.3 shows a visualization of

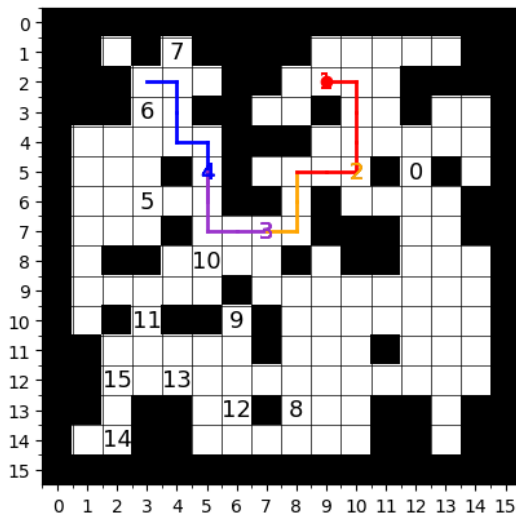
Table 5.4: The ablation studies of our method on the unseen gird-world maps.

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

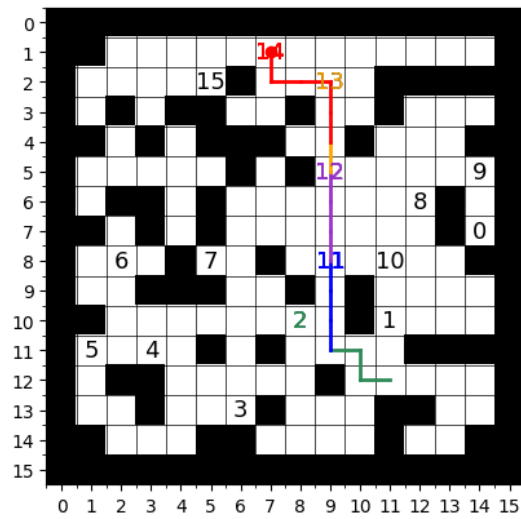
the learned GRG, which captures the goal relations well.

**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 5.4 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 5.4 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.

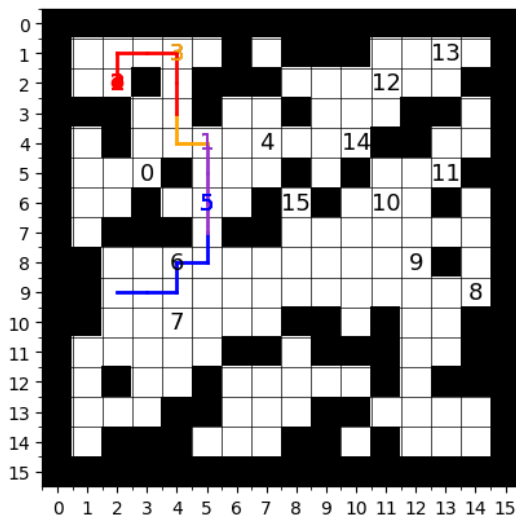
**Qualitative results.** We show some qualitative results performed by our method



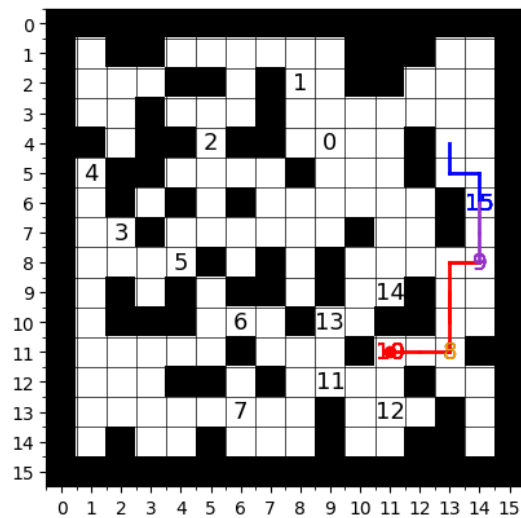
(a)



(b)



(c)



(d)

Figure 5.4: Trajectories generated by our method on the unseen grid-world maps for both the seen goals (a) (b) and the unseen goals (c) (d). The different colors represent different sub-goals and the corresponding sub-goal-oriented trajectories where the red one denotes the designated final goal.

on the unseen grid-world maps to achieve both seen goals and unseen goals in Figure 5.4.

#### 5.4.2 Robotic Object Search

Robotic object search is a challenging goal-driven visual navigation task (Yang *et al.*, 2018; Ye *et al.*, 2018; Mousavian *et al.*, 2019; Nguyen *et al.*, 2019; Ye *et al.*, 2019a; Qiu *et al.*, 2020; Batra *et al.*, 2020). 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 (Yang *et al.*, 2018) also incorporates object relations as scene priors to improve the robotic object search performance in the AI2-THOR (Kolve *et al.*, 2017b) environments. Unlike ours, it extracts the object relations from the Visual Genome (Krishna *et al.*, 2017) corpus and incorporates the relations through Graph Convolutional Networks (Kipf and Welling, 2016). 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.  $\Phi$ ). In addition, we adopt the A3C (Mnih *et al.*, 2016) 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 Yang *et al.* (2018) to implement both SCENE PRIORS (Yang *et al.*, 2018) (without “stop” action) and our HRL-GRG. Unlike SCENE PRIORS (Yang *et al.*, 2018), the first-person view semantic segmentation and the depth map as our agent’s pre-processed observations are of window size  $30 \times 30$  where we further concatenate 4 history of them as the inputs to our HRL-GRG. In addition, we

adopt the Double DQN (Van Hasselt *et al.*, 2016) technique for our high-level network and we pre-train our low-level network to approach a visible object. We train our method with the curriculum training paradigm we described in Section 5.4.1. All hyperparameters are summarized in Table 5.7. Both the methods are implemented with Tensorflow toolbox and experimented on Nvidia V100 GPUs and 16 Intel Xeon E5-2680 v4 CPU cores where each training takes around 30 hours in general. We report the results in Table 5.5 where we compare the two methods in terms of their performance improvement over the RANDOM method. Table 5.5 indicates an overfitting issue of the SCENE PRIORS method as reported in Yang *et al.* (2018) as well. At the same time, we observe a superior generalization ability of our method especially to the unseen goals.

Table 5.5: The performance improvement of SCENE PRIORS (Yang *et al.*, 2018) (top) and our HRL-GRG (bottom) over the RANDOM method in the AI2-THOR (Kolve *et al.*, 2017b) environment for the robotic object search task (without stop action).

		Seen Goals		Unseen Goals	
		SR↑	SPL↑	SR↑	SPL↑
Seen Env.	Yang <i>et al.</i> (2018)	+0.25	+0.16	+0.08	+0.07
	<b>Ours</b>	<b>+0.37</b>	<b>+0.24</b>	<b>+0.33</b>	<b>+0.23</b>
Unseen Env.	Yang <i>et al.</i> (2018)	+0.18	+0.11	+0.12	+0.06
	<b>Ours</b>	<b>+0.33</b>	<b>+0.21</b>	<b>+0.38</b>	<b>+0.23</b>

Figure 5.5 shows some examples of how our method searches for unseen objects in unseen AI2-THOR (Kolve *et al.*, 2017b) scenes.

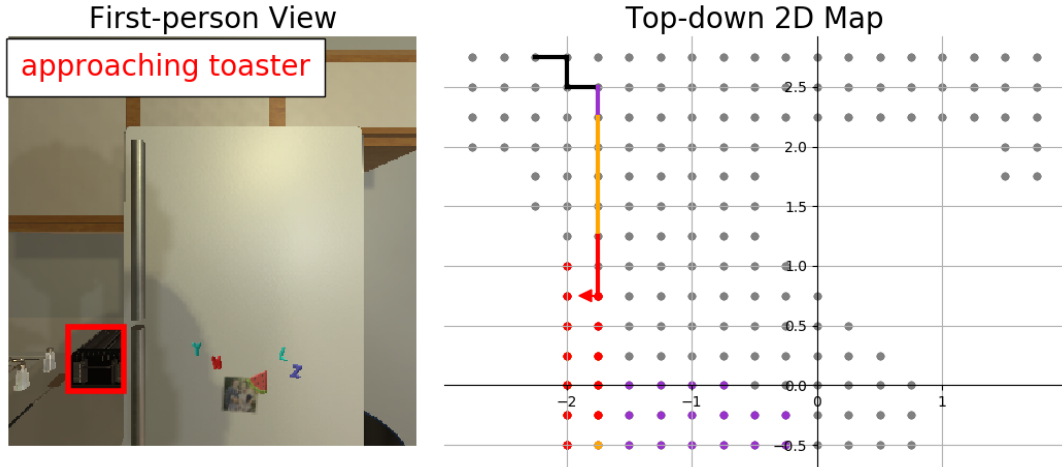
To further demonstrate the efficacy of our method in more complex environments, we conduct robotic object search on the House3D (Wu *et al.*, 2018) platform. Different

Table 5.6: The performance of all methods in the House3D (Wu *et al.*, 2018) environment for the robotic object search task.

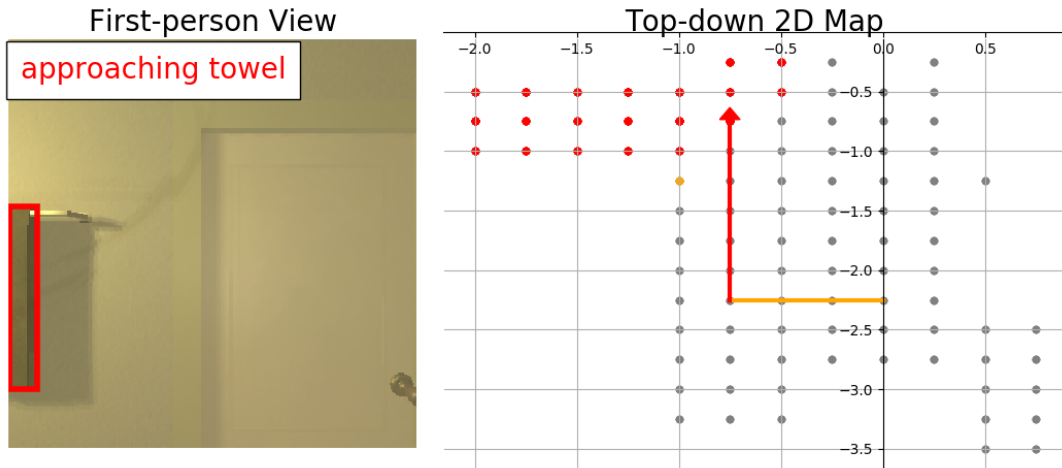
Method	Single Environment				Multiple Environments			
	Seen Goals		Unseen Goals		Seen Env.		Unseen Env.	
	SR $\uparrow$	SPL $\uparrow$	SR $\uparrow$	SPL $\uparrow$	SR $\uparrow$	SPL $\uparrow$	SR $\uparrow$	SPL $\uparrow$
RANDOM	0.20	0.05	0.23	0.04	0.39	0.03	0.60	0.05
DQN	0.58	0.27	0.18	0.05	0.42	0.06	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
<b>Ours</b>	<b>0.88</b>	<b>0.33</b>	<b>0.79</b>	<b>0.21</b>	<b>0.76</b>	<b>0.20</b>	<b>0.62</b>	<b>0.10</b>

Table 5.7: Hyperparameters of all the methods for the robotic object search task.

Hyperparameter	Description	Value
$\gamma$	Discount factor	0.99
lr	Learning rate for all networks (high-level/low-level)	0.0001
main_update	Interval of updating all the main networks of Double DQN	100
target_update	Interval of updating all the target networks of Double DQN	100000
A3C_update	Interval of updating all the A3C networks	10
$\beta$	The weight of the entropy regularization term in the A3C networks	0.01
batch_size	Batch size for training DQN networks	64
epsilon	Initial exploration rate, anneal episodes, final exploration rate	1, 10000, 0.1
max_episodes	Maximum episodes to train each method	100000
$N_{max}^l$	Maximum steps that low-level network can take in AI2-THOR / House3D	10 / 50
$N_{max}$	Maximum steps that each method can take in AI2-THOR / House3D	200 / 1000
optimizer	Optimizer for all the networks	RMSProp
$\alpha_{ij}$	The hyperparameter of our GRG ( $\alpha_{i,j,1}, \dots, \alpha_{i,j,10}, \alpha_{i,j,11}$ )	(0, ..., 0, 1)



(a) kitchen (toaster)



(b) bathroom (towel)

Figure 5.5: Trajectories generated by our method for the robotic object search task on AI2-THOR (Kolve *et al.*, 2017b).

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 forward / backward / left / right 0.2 meters, or rotates 90 degrees for each action step. We adopt the encoder-decoder model from Chen *et al.* (2018) to predict both the semantic segmentation and the depth map from the first-person view RGB image of window size  $600 \times 450$ . We then resize the both predictions to the size  $10 \times 10$  and we take them as the agent’s partial observation. Furthermore, we adopt the A3C (Mnih *et al.*, 2016) algorithm for the low-level network. We compare our method with the baseline methods introduced in Section 5.4.1 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.

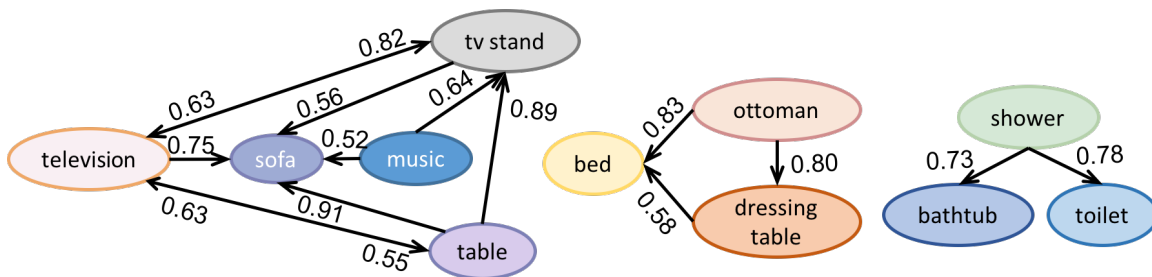


Figure 5.6: The object relations captured by our GRG in the House3D (Wu *et al.*, 2018) environment for the robotic object search task. Only a small number of objects as nodes and the edges with the weight  $\geq 0.5$  are shown.

We adopt the Double DQN (Van Hasselt *et al.*, 2016) technique for all the DQN networks. We pre-train the method A3C to approach an object when the object is observable and we take it as the pre-trained model for the low-level networks of both HRL and our method as well. For all the methods, we adopt the same curriculum training paradigm as we described in Section 5.4.1 and we summarize the hyperparameters in Table 5.7. We set the maximum steps for all the aforementioned

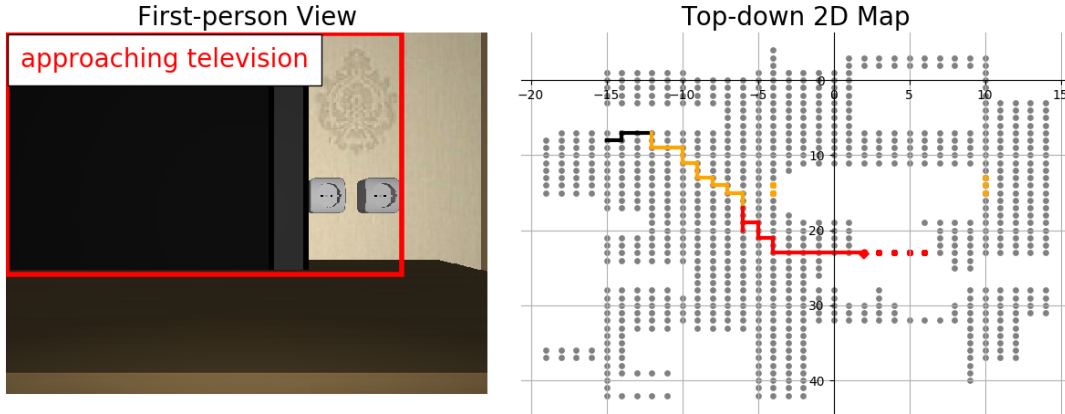


methods to solve the object search task in the House3D environment as 1000, and the maximum steps that the low-level networks of the hierarchical methods (HRL and ours) can take as 50. To better investigate each method’s properties, we first train and evaluate in a single environment. The training is conducted on Nvidia V100 GPUs and 16 Intel Xeon E5-2680 v4 CPU cores where each training takes around 40 hours. Table 5.6 (left part) shows the results. 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. Here, each training takes around 60 hours on the servers with the same configurations. The results presented in Table 5.6 (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 5.6).

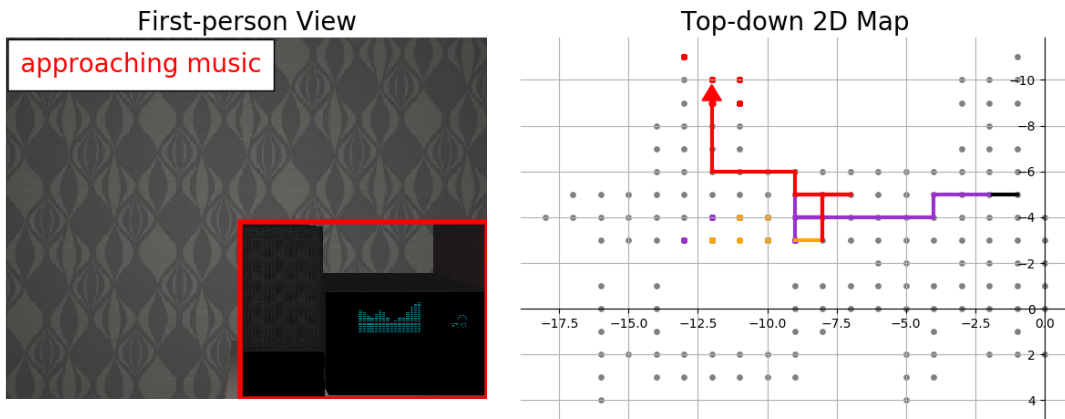
Figure 5.7 shows some qualitative results of our method for the robotic object search task on House3D (Wu *et al.*, 2018). The agent can only access the first-person view RGB images while the top-down 2D maps are placed for better visualization.

## 5.5 Conclusion

In this chapter, we present a novel hierarchical reinforcement learning approach equipped with a GRG formulation for the partially observable goal-driven task. Our



(a) seen environment seen goal



(d) unseen environment unseen goal

Figure 5.7: Trajectories generated by our method for the robotic object search task on House3D (Wu *et al.*, 2018).

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.

TOWARDS LEARNING FROM HUMAN-PROVIDED PRIORS FOR  
GENERALIZABILITY**6.1 Environment Dynamics from Natural Language Instructions**

When asking human beings to perform a task in a completely new environment, we may not succeed either if we are not familiar with the environment. It is because the new environment information (e.g. the environment dynamics) can hardly be inferred from our limited observations or past experiences, especially when the new environment is unusual. For example, while a TV remote control is always placed on the tea table in the living room, it could be left at a bedroom sometimes, requiring an exploration of the environment to find it. In such a case, to perform the task more efficiently, we typically ask other experienced people for instructions, such as “finding the TV remote control in the bedroom”. The instructions transfer the high-level environment dynamics that enable the policy to generalize towards the new environments. In this chapter, we study Vision-and-Language Navigation (VLN) task where we aim to infer the high-level environment dynamics from natural language instructions for generalizable policy learning.

The VLN task requires an agent to navigate in a real-world environment following natural language instructions (see Figure 6.1). The preliminary challenge of the VLN task is the cross-modal grounding of the visual observations and natural language instructions. With implicit supervisions, such as the desired navigation trajectories provided by the VLN benchmark dataset R2R (Anderson *et al.*, 2018b), many works make effort towards representation learning to better capture the visual-textual

correspondence. Wang *et al.* (2019) presented a novel cross-modal matching architecture to ground language instruction on both local visual observation and global visual trajectory. Hong *et al.* (2020) proposed a novel language and visual entity relationship graph modeling relations among scene, object and directional clues. Hong *et al.* (2021) and Qi *et al.* (2021) equipped their VLN models with a pre-trained V&L BERT (Vision and Language Bidirectional Encoder Representations from Transformers) (Devlin *et al.*, 2018). In Ma *et al.* (2019), Zhu *et al.* (2020) and Huang *et al.* (2019), the authors proposed self-supervised auxiliary tasks to accelerate the learning of the effective feature representations. Both Ma *et al.* (2019) and Zhu *et al.* (2020) estimated the navigation progress represented by either the distance towards the goal location or the percentage of steps. Zhu *et al.* (2020) and Huang *et al.* (2019) performed cross-modal alignment task, where Zhu *et al.* (2020) checked if the the language feature matches the vision-language feature while Huang *et al.* (2019) predicted if a given instruction-path fit each other. In addition, Zhu *et al.* (2020) also proposed the trajectory retelling task to reconstruct the instruction words and the angle prediction task to predict the ground-truth action angles considering the available actions incorporate vision noises. Some methods augmented the training data in order to acquire more robust feature representations so that they can achieve better generalization ability. For instance, the Speaker-Follower models introduced in Fried *et al.* (2018) augmented data by adopting its speaker model to create synthetic instructions on sampled new routes. Tan *et al.* (2019) came up with the “environment dropout” method to mimic unseen environments. Parvaneh *et al.* (2020) generated counterfactual environments to account for unseen scenarios.

Unlike the previous work, we propose a high-level network to capture the environment dynamics from the natural language instructions. Given the current visual observations, the high-level network determines a sub-task/sub-goal that should be

done for the next step. The sub-task/sub-goal is represented by an attentive instruction feature, highlighting either a visual concept, such as an object (e.g. “table”, “door”), or an action specification, like “go straight”, “turn left” in the instruction. Our low-level network learns an action policy from the visual observations to perform the proposed sub-task. Since the sub-task is provided as a simple concept, we decouple the policy learning from the language understanding. As a result, our two-layer hierarchical method is expected to be more robust.

Specifically, in VLN task, the goal description  $G_d$  is given by a natural language instruction with  $L$  words, i.e.  $\{w_1, w_2, \dots, w_L\}$ . The visual observation  $o_t$  at time step  $t$  is a panoramic view that is discretized into 36 single views  $\{o_{t,1}, o_{t,2}, \dots, o_{t,36}\}$  (12 horizontal by 3 vertical). Each view  $o_{t,i}$  ( $1 \leq i \leq 36$ ) is represented by an RGB image  $v_{t,i}$  and its orientation  $(\theta_{t,i}, \phi_{t,i})$ , where  $\theta_{t,i}$  is the heading angle and  $\phi_{t,i}$  is the elevation angle. At each step  $t$ , there are  $N_t$  predefined navigable viewpoints for the agent to select from, namely  $\{l_{t,1}, l_{t,2}, \dots, l_{t,N_t}\}$ . Similarly,  $l_{t,k}$  ( $1 \leq k \leq N_t$ ) is also represented by an RGB image  $v_{t,k}$  and the orientation relative to the current orientation  $(\Delta\theta_{t,k}, \Delta\phi_{t,k})$ . Here,  $v_{t,k} \in \{v_{t,1}, v_{t,2}, \dots, v_{t,36}\}$ . Following the common practice (Fried *et al.*, 2018; Tan *et al.*, 2019), we concatenate the ResNet (He *et al.*, 2016) feature of the RGB image and the orientation feature to represent both the observation  $o_{t,i}$  with  $f_{t,i}$  and the candidate navigable viewpoint  $l_{t,k}$  with  $a_{t,k}$  as Equation 6.1 shows.

$$\begin{aligned} f_{t,i} &= [\text{ResNet}(v_{t,i}); (\cos\theta_{t,i}, \sin\theta_{t,i}, \cos\phi_{t,i}, \cos\phi_{t,i})] \\ a_{t,k} &= [\text{ResNet}(v_{t,k}); (\cos\Delta\theta_{t,k}, \sin\Delta\theta_{t,k}, \cos\Delta\phi_{t,k}, \cos\Delta\phi_{t,k})] \end{aligned} \tag{6.1}$$

Figure 6.1 illustrates the baseline VLN model EnvDrop (Tan *et al.*, 2019). It is an encoder-decoder model. The encoder model is a bidirectional LSTM-RNN equipped with an embedding layer encoding the natural language instruction as formulated in Equation 6.2. The decoder is an attentive LSTM-RNN decoding the instruction at

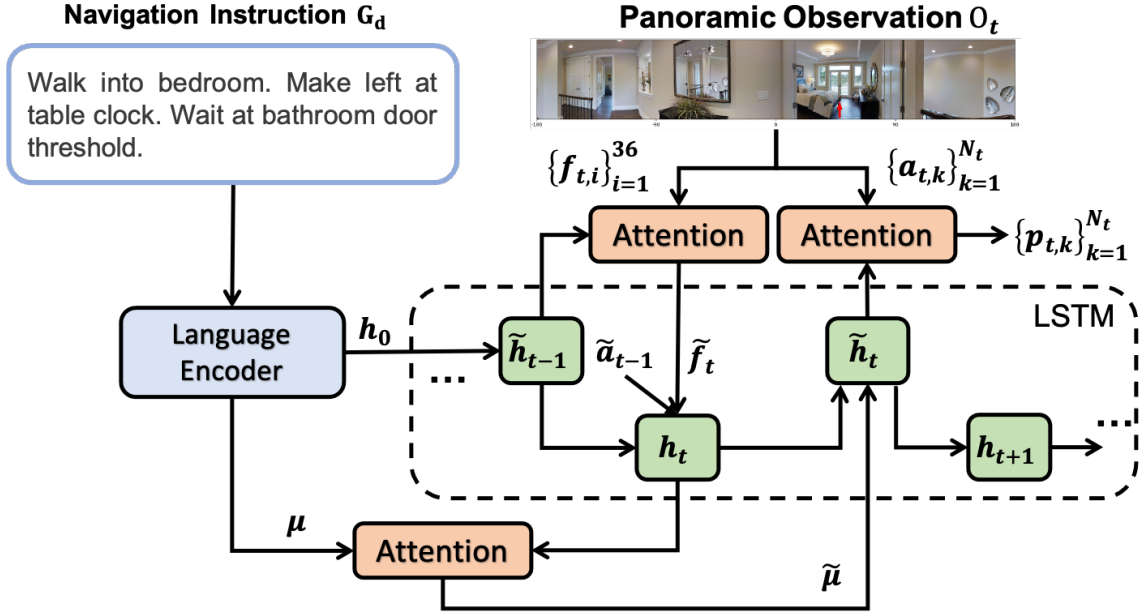


Figure 6.1: An illustration of the baseline VLN model EnvDrop (Tan *et al.*, 2019).

each step  $t$  given the visual feature  $f_t$  and the previous action embedding  $\tilde{a}_{t-1}$ . The formulation is shown in Equation 6.3 where  $W_F$ ,  $W$  and  $W_U$  are learnable parameters. Finally, the probability of moving to the navigable viewpoint  $l_{t,k}$  can be calculated by Equation 6.4 where  $W_A$  is another learnable parameter. EnvDrop is trained with both imitation learning and reinforcement learning methods.

$$u_1, u_2, \dots, u_L = \text{Bi-LSTM}(\text{embedding}(w_1), \dots, \text{embedding}(w_L)) \quad (6.2)$$

$$\begin{aligned}
 h_t &= \text{LSTM}([\tilde{f}_t; \tilde{a}_{t-1}], \tilde{h}_{t-1}) \\
 \tilde{f}_t &= \sum_i \alpha_{t,i} f_{t,i}, \quad \alpha_{t,i} = \text{softmax}_i(f_{t,i}^T W_F \tilde{h}_{t-1}) \\
 \tilde{h}_t &= \tanh(W[\tilde{u}_t; h_t]) \\
 \tilde{u}_t &= \sum_j \beta_{t,j} u_j, \quad \beta_{t,j} = \text{softmax}_j(u_j^T W_U h_t)
 \end{aligned} \quad (6.3)$$

$$p_{t,k} = \text{softmax}_k(a_{t,k}^T W_A \tilde{h}_t) \quad (6.4)$$

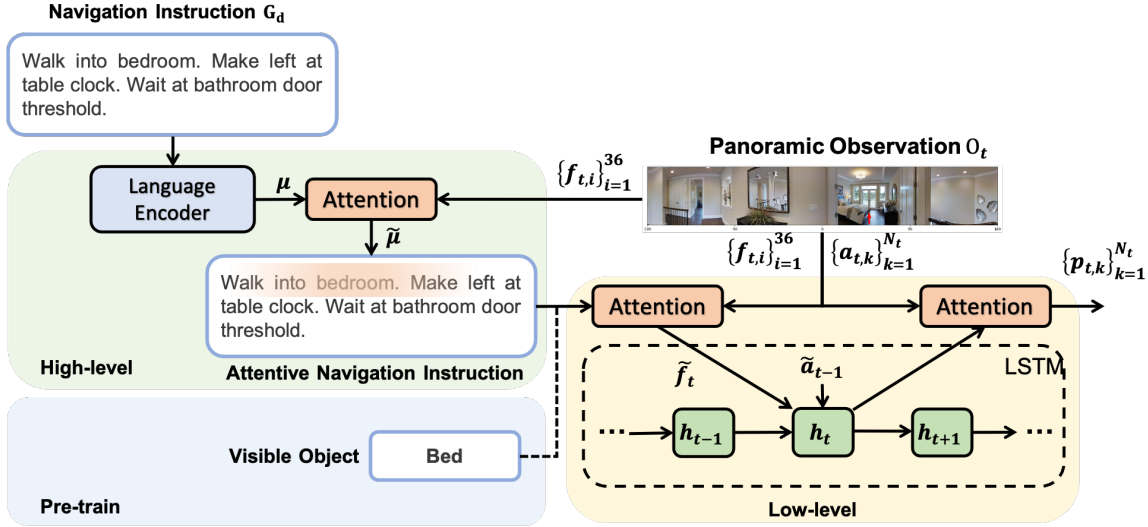


Figure 6.2: An illustration of our proposed VLN model.

Instead of matching vision and language to determine a viewpoint to navigate which is brittle as the instruction may not be informative for step-wise decision making, we here propose to decouple the policy learning from the language understanding with a two-layer hierarchical model as shown in Figure 6.2. Our high-level layer is an attention network that generates an attentive instruction as a sub-goal (see Equation 6.5). The low-level layer is an LSTM-RNN learning an action policy to achieve the proposed sub-goal. In order to be adaptive to various sub-goals, we learn the action policy from the attentive visual features that is sub-goal aware. Equation 6.6 describes the formulation.

$$\tilde{u}_t = \sum_j \beta_{t,j} u_j, \quad \beta_{t,j} = \text{softmax}_j(u_j^T W_U f_{t,i}) \quad (6.5)$$



$$\begin{aligned}
h_t &= \text{LSTM}([\tilde{f}_t; \tilde{a}_{t-1}], h_{t-1}) \\
\tilde{f}_t &= \sum_i \alpha_{t,i} f_{t,i}, \quad \alpha_{t,i} = \text{softmax}_i(f_{t,i}^T W_F \tilde{u}_t) \\
p_{t,k} &= \text{softmax}_k(a_{t,k}^T W_A h_t)
\end{aligned} \tag{6.6}$$

Our high-level and low-level layers can be trained jointly to perform the VLN task with imitation learning and reinforcement learning methods as the EnvDrop. Moreover, our low-level network can be trained with additional datasets. To be specific, we take the semantic labels of the objects that are visible in the observations as pseudo attentive instructions and we train our low-level network to approach the designated visible objects. We define the object locations as the viewpoints that yield largest observations of the objects. As a result, we can calculate the shortest paths towards the object locations for imitation learning and/or define reward functions upon their distances for reinforcement learning of the low-level policy. The additional dataset supplements the VLN dataset with sub-goals that are of high variety and could also be referred in natural language instructions, which in consequence help to learn a more robust low-level policy.




## 6.2 Environment Dynamics from Human Demonstrations

For more complex tasks, such as object manipulation, it has shown to be efficient to let a robot to learn from observing human demonstrators performing them (Argall *et al.*, 2009). Under this setting, the robot first visually observes the human performance through its perception sensors. Then, the visual signals are parsed into middle level representations. At the end, the representations are reversely interpreted into a sequence of motor signals for the robot physically executing the learned task.




However, the natural disparity in action space between robots and humans makes it difficult for a robot to imitate the human demonstrators step by step. To be specific,

the mechanical design of robots is not constrained by the phenomenon of evolution. Industrial robots may be equipped with multiple heterogeneous end effectors. Even for humanoid robots, the common phenomenon of handedness of human beings does not apply to robots. But, among the publicly available video corpora of human beings performing manipulation actions, such as MANIAC (Aksoy *et al.*, 2017) or 50 Salads dataset (Stein and McKenna, 2013), around 90% of the performers are right-handed. Therefore, simply mimicking the action sequences of human demonstrators inevitably constrains the robots with handedness.

Instead of learning an action policy from human demonstrations, we hereby propose to capture a middle level representation of human demonstrations as a high-level environment dynamic to facilitate the robot learning of the generalizable action policy. We take the object manipulation task as an example and we assume a human demonstration is provided as a video that depict human doing an activity step by step. To capture a high-level environment dynamic, we first need perception modules to convert the sensory signals into semantic labels. Here, we model a primitive action as a process which takes an object as an input and then updates the object’s state, including the state of its appearance and physical position, to achieve an action sub-goal. For instance, consider the salad making scenario, the primitive action “cut cucumber” takes “cucumber” as an input and then updates its state from “whole” to “chopped”. “Put on top” is another kind of primitive action that updates the position state of the input object from one location to another location (see Fig. 6.3 for examples). Following this way of modeling primitive manipulation action, we adopt three perception modules: an object recognition module (Lei *et al.*, 2012) recognizing the object under the action with an object label  $o$ ; an action detection module (Lea *et al.*, 2016) recognizing the action with an action label  $a$ , and an object state perception module (Yang *et al.*, 2013) recognizing the object state before and after the

	...		...	
Object	Cucumber	Cucumber	Cucumber	Cucumber
Object State	Whole	Chopped	Chopped	Chopped
Action	Cut	Cut		

	...		...	
Object	Cup	Cup	Cup	Cup
Object State	On table	On box	On box	On box
Action		Put on top	Put on top	

Action Representation:



Figure 6.3: Action representation. Top two rows are two examples of primitive actions. The upper one is from the 50 Salads (Stein and McKenna, 2013) dataset and the lower one is from the MANIAC dataset (Aksoy *et al.*, 2017). Bottom row is an abstract representation of one example primitive action.

action is performed with label  $s$  and  $s'$  respectively. We represent the object-centric action as a quad  $(o, s, a, s')$ . As a result, a demonstration video can be represented as a sequence of object-centric actions, i.e.  $(o_1, s_1, a_1, s'_1), (o_2, s_2, a_2, s'_2), \dots, (o_t, s_t, a_t, s'_t)$ .

Once an object-centric action  $(o_i, s_i, a_i, s'_i)$  is recognized by the three perception

modules, the object  $o_i$  and its state before the action happens  $s_i$  indicates the action's precondition, while the object state afterwards  $s'_i$  represents the action's consequence. Following this protocol, we create a node  $v = a_i$ . If the action's precondition  $(o_i, s_i)$  is the consequence of the previous recognized actions  $(o_k, s'_k)$ , we create a directed edge  $e$  from the previous action nodes to the current action node  $e = \langle a_k, a_i \rangle$ , to represent the precedence relations between these action nodes. We build the high-level environment dynamic as the directed acyclic graph  $(V, E)$  at the end, where  $V = \{v \mid \text{action node } v\}$  and  $E = \{e \mid \text{directed edge } e\}$ , that captures the precedence relations between all recognized actions. Figure 6.4 shows an example of the graph learned on 50 Salads dataset (Stein and McKenna, 2013).

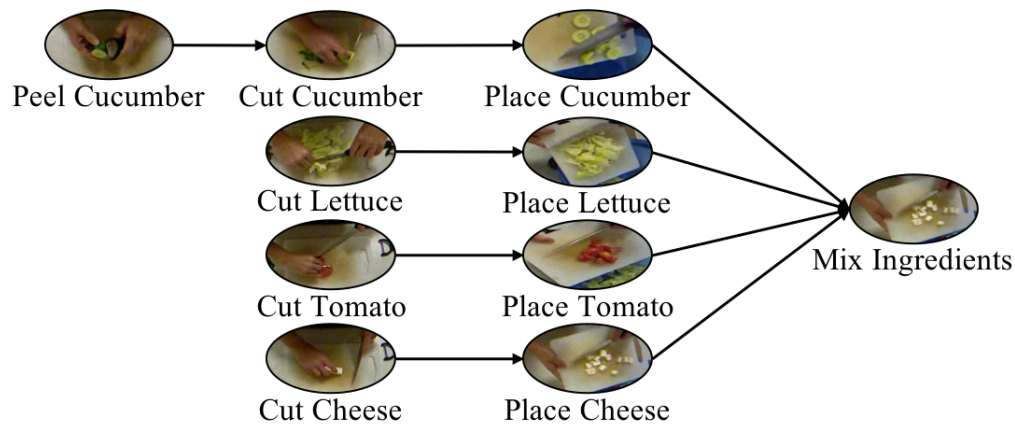


Figure 6.4: The GRG learned on 50 Salads dataset (Stein and McKenna, 2013).

With the learned precedence graph as the high-level environment dynamic, we can adopt a hierarchical reinforcement learning approach similar as Chapter 5 to learn a generalizable action policy performing the demonstrated task. Here, the high-level network predicts a primitive action as a sub-goal given the state estimated by the object state perception module. Equipped with the learned precedence graph, our high-level network is expected to generalize well to new configurations of the demonstrated task, such as making salads in a novel kitchen with all ingredients

placed in different positions. The objective of our low-level networks is to perform the proposed primitive action. The primitive actions with different action labels can be learned with different networks, and each network can be trained to perform the specified action over various objects in order to achieve generalization ability.

## CONCLUSION

Throughout the dissertation, we present several methods to address the challenges in learning an action policy from the agent’s visual observations for the agent to perform the task well. First, to enable the policy learning from the high dimensional visual observations, we propose to improve the sample efficiency by introducing a denser reward function in Chapter 2. The reward function is defined upon a vision module that performs a prediction task over agent’s visual observations. The experimental results demonstrate that our proposed reward function enables a better action policy for the robotic object search task on AI2-THOR (Kolve *et al.*, 2017b) simulation platform and a real indoor environment with a physical robot. To further improve the sample efficiency and address the common sparse reward issue, we propose a novel hierarchical policy learning paradigm in Chapter 3 for efficient exploration in high dimensional state space. 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. The empirical and extensive experiments together with the ablation studies on House3D (Wu *et al.*, 2018) platform where a sparse reward setting is deployed demonstrate the efficacy and efficiency of our presented framework.

Following the methods that enable policy learning from visual observations for a specific task, we explore the learning of generalizable action policies. In Chapter 4, we present a generalizable policy learning for the robotic object approach task which is a fully observable version of the robotic object search task, through task-relevant representations, namely depth estimation and semantic segmentation. Empirical studies

on the House3D (Wu *et al.*, 2018) platform and a real physical experiment on a mobile robot validate that our method can yield a significantly higher generalization capability towards new target objects and novel environments. To learn a generalizable policy for partially observable task which is more common, we present in Chapter 5 a hierarchical reinforcement learning approach equipped with a Goals Relational Graph (GRG) formulation. Our GRG captures the underlying relations of all goals in the goal space as a representation of the high-level environment dynamics that helps our high-level network generalize. At the same time, our low-level network also generalizes well with the method introduced in Chapter 4. We validate our approach on the grid-world domain, and both AI2-THOR (Kolve *et al.*, 2017b) and House3D (Wu *et al.*, 2018) simulation platform for the robotic object search task. The results show our approach is effective and is exceptional in generalizing to unseen environments and new goals.

Finally, we explore in Chapter 6 the direction of inferring the high-level environment dynamics from human-provided priors, such as natural language based instructions and human demonstrations, for generalizable policy learning.

For future work, we believe it is worth studying the vision-guided policy learning problem with the visual navigation task (Ye and Yang, 2020). The visual navigation task is a challenging AI task that requires an agent to develop comprehensive capabilities, including visual understanding, policy learning, language grounding, knowledge representation, common-sense reasoning. While the task has drawn increased research interest and been studied extensively, the performance still leaves much to be desired, indicating a significant space we can further explore. Based on the visual navigation task, we think it would be more promising to take a further look into the following questions, 1) the definitions of a policy’s optimality and generalizability; 2) the relations between the optimality and the generalizability of a policy. Current

definitions of optimality and generalizability are ambiguous. For example, the consensus of the optimal policy in the visual navigation task is the shortest path towards the navigation goal which might not always be reasonable. In addition, some work may require a generalizable policy to be optimal in any new environment while the others may allow a few updates to the learned policy. In such a situation, more clear definitions are urgently required. Moreover, while the optimality relies heavily on the knowledge of the environments, the generalizability requires the policy to be environment-independent. As a result, it is critical to achieving a balance between them and we should benefit a lot from developing the relations between them like the one built for the speed-accuracy trade-off. Finally, transfer learning would also be an interesting avenue for future research, with which we can learn from other tasks rather than taking each task as an independent task.



## REFERENCES

- Abbeel, P. and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning”, in “Proceedings of the twenty-first international conference on Machine learning”, p. 1 (2004).
- Aksoy, E. E., A. Orhan and F. Wörgötter, “Semantic decomposition and recognition of long and complex manipulation action sequences”, *International Journal of Computer Vision* **122**, 1, 84–115 (2017).
- Ammirato, P., P. Poirson, E. Park, J. Košecká and A. C. Berg, “A dataset for developing and benchmarking active vision”, in “2017 IEEE International Conference on Robotics and Automation (ICRA)”, pp. 1378–1385 (IEEE, 2017).
- Anderson, P., A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, “On evaluation of embodied navigation agents”, arXiv preprint arXiv:1807.06757 (2018a).
- Anderson, P., Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould and A. 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”, pp. 3674–3683 (2018b).
- Andreas, J., D. Klein and S. Levine, “Modular multitask reinforcement learning with policy sketches”, in “Proceedings of the 34th International Conference on Machine Learning-Volume 70”, pp. 166–175 (JMLR. org, 2017).
- Aneja, J., A. Deshpande and A. G. Schwing, “Convolutional image captioning”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 5561–5570 (2018).
- Antol, S., A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick and D. Parikh, “Vqa: Visual question answering”, in “Proceedings of the IEEE international conference on computer vision”, pp. 2425–2433 (2015).
- Argall, B. D., S. Chernova, M. Veloso and B. Browning, “A survey of robot learning from demonstration”, *Robotics and autonomous systems* **57**, 5, 469–483 (2009).
- Arora, S. and P. Doshi, “A survey of inverse reinforcement learning: Challenges, methods and progress”, arXiv preprint arXiv:1806.06877 (2018).
- Arulkumaran, K., M. P. Deisenroth, M. Brundage and A. A. Bharath, “A brief survey of deep reinforcement learning”, arXiv preprint arXiv:1708.05866 (2017).
- Bacon, P.-L., J. Harb and D. Precup, “The option-critic architecture”, in “Thirty-First AAAI Conference on Artificial Intelligence”, (2017).

- Batra, D., A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev and E. Wijmans, “Objectnav revisited: On evaluation of embodied agents navigating to objects”, arXiv preprint arXiv:2006.13171 (2020).
- Bütepage, J., S. Cruciani, M. Kokic, M. Welle and D. Kragic, “From visual understanding to complex object manipulation”, *Annual Review of Control, Robotics, and Autonomous Systems* **2**, 161–179 (2019).
- Chang, A., A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng and Y. Zhang, “Matterport3d: Learning from rgb-d data in indoor environments”, arXiv preprint arXiv:1709.06158 (2017).
- Chaplot, D. S., D. P. Gandhi, A. Gupta and R. R. Salakhutdinov, “Object goal navigation using goal-oriented semantic exploration”, *Advances in Neural Information Processing Systems* **33** (2020).
- Chen, L.-C., G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”, *IEEE transactions on pattern analysis and machine intelligence* **40**, 4, 834–848 (2017a).
- Chen, L.-C., Y. Zhu, G. Papandreou, F. Schroff and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation”, in “Proceedings of the European conference on computer vision (ECCV)”, pp. 801–818 (2018).
- Chen, S., P. B. Gibbons, M. Kozuch, V. Liaskovitis, A. Ailamaki, G. E. Blelloch, B. Falsafi, L. Fix, N. Hardavellas, T. C. Mowry *et al.*, “Scheduling threads for constructive cache sharing on cmps”, in “Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures”, pp. 105–115 (ACM, 2007).
- Chen, Y. F., M. Everett, M. Liu and J. P. How, “Socially aware motion planning with deep reinforcement learning”, in “2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)”, pp. 1343–1350 (IEEE, 2017b).
- Chollet, F., “Xception: Deep learning with depthwise separable convolutions”, arXiv preprint pp. 1610–02357 (2017).
- Coupric, C., C. Farabet, L. Najman and Y. LeCun, “Indoor semantic segmentation using depth information”, arXiv preprint arXiv:1301.3572 (2013).
- Das, A., 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”, pp. 2054–2063 (2018a).
- Das, A., G. Gkioxari, S. Lee, D. Parikh and D. Batra, “Neural modular control for embodied question answering”, arXiv preprint arXiv:1810.11181 (2018b).
- Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding”, arXiv preprint arXiv:1810.04805 (2018).

- Donahue, J., Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition”, in “International conference on machine learning”, pp. 647–655 (2014).
- Dosovitskiy, A. and V. Koltun, “Learning to act by predicting the future”, arXiv preprint arXiv:1611.01779 (2016).
- Druon, R., Y. Yoshiyasu, A. Kanazaki and A. Watt, “Visual object search by learning spatial context”, *IEEE Robotics and Automation Letters* **5**, 2, 1279–1286 (2020).
- Dwiel, Z., M. Candadai, M. J. Phielipp and A. K. Bansal, “Hierarchical policy learning is sensitive to goal space design”, arXiv preprint arXiv:1905.01537 (2019).
- Eigen, D. and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture”, in “Proceedings of the IEEE International Conference on Computer Vision”, pp. 2650–2658 (2015).
- Everett, M., Y. F. Chen and J. P. How, “Motion planning among dynamic, decision-making agents with deep reinforcement learning”, in “2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)”, pp. 3052–3059 (IEEE, 2018).
- Fried, D., R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein and T. Darrell, “Speaker-follower models for vision-and-language navigation”, *Advances in Neural Information Processing Systems* **31**, 3314–3325 (2018).
- Fu, H., M. Gong, C. Wang, K. Batmanghelich and D. Tao, “Deep ordinal regression network for monocular depth estimation”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 2002–2011 (2018).
- Fu, J., A. Korattikara, S. Levine and S. Guadarrama, “From language to goals: Inverse reinforcement learning for vision-based instruction following”, arXiv preprint arXiv:1902.07742 (2019).
- Ghosh, D., A. Singh, A. Rajeswaran, V. Kumar and S. Levine, “Divide-and-conquer reinforcement learning”, arXiv preprint arXiv:1711.09874 (2017).
- Gordon, D., A. Kembhavi, M. Rastegari, J. Redmon, D. Fox and A. Farhadi, “Iqa: Visual question answering in interactive environments”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 4089–4098 (2018).
- Goyal, Y., T. Khot, D. Summers-Stay, D. Batra and D. Parikh, “Making the v in vqa matter: Elevating the role of image understanding in visual question answering”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 6904–6913 (2017).
- Gu, S., E. Holly, T. Lillicrap and S. Levine, “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates”, in “2017 IEEE International Conference on Robotics and Automation (ICRA)”, pp. 3389–3396 (IEEE, 2017).

- Guo, Y., Y. Liu, A. Oerlemans, S. Lao, S. Wu and M. S. Lew, “Deep learning for visual understanding: A review”, *Neurocomputing* **187**, 27–48 (2016).
- Gupta, S., J. Davidson, S. Levine, R. Sukthankar and J. Malik, “Cognitive mapping and planning for visual navigation”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 2616–2625 (2017).
- Haarnoja, T., A. Zhou, P. Abbeel and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”, arXiv preprint arXiv:1801.01290 (2018).
- Han, D., K. Doya and J. Tani, “Variational recurrent models for solving partially observable control tasks”, arXiv preprint arXiv:1912.10703 (2019).
- He, K., X. Zhang, S. Ren and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”, in “Proceedings of the IEEE international conference on computer vision”, pp. 1026–1034 (2015).
- He, K., X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 770–778 (2016).
- Hong, Y., C. Rodriguez-Opazo, Y. Qi, Q. Wu and S. Gould, “Language and visual entity relationship graph for agent navigation”, in “Advances in Neural Information Processing Systems”, (2020).
- Hong, Y., Q. Wu, Y. Qi, C. Rodriguez-Opazo and S. Gould, “Vln bert: A recurrent vision-and-language bert for navigation”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 1643–1653 (2021).
- Hossain, M. Z., F. Sohel, M. F. Shiratuddin and H. Laga, “A comprehensive survey of deep learning for image captioning”, *ACM Computing Surveys (CSUR)* **51**, 6, 1–36 (2019).
- Huang, H., V. Jain, H. Mehta, A. Ku, G. Magalhaes, J. Baldrige and E. Ie, “Transferable representation learning in vision-and-language navigation”, in “Proceedings of the IEEE/CVF International Conference on Computer Vision”, (2019).
- Igl, M., L. Zintgraf, T. A. Le, F. Wood and S. Whiteson, “Deep variational reinforcement learning for pomdps”, arXiv preprint arXiv:1806.02426 (2018).
- Jaderberg, M., V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver and K. Kavukcuoglu, “Reinforcement learning with unsupervised auxiliary tasks”, arXiv preprint arXiv:1611.05397 (2016).
- Jafari, O. H., O. Groth, A. Kirillov, M. Y. Yang and C. Rother, “Analyzing modular cnn architectures for joint depth prediction and semantic segmentation”, in “Robotics and Automation (ICRA), 2017 IEEE International Conference on”, pp. 4620–4627 (IEEE, 2017).

- Kipf, T. N. and M. Welling, “Semi-supervised classification with graph convolutional networks”, arXiv preprint arXiv:1609.02907 (2016).
- Kolve, E., R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta and A. Farhadi, “AI2-THOR: An Interactive 3D Environment for Visual AI”, arXiv (2017a).
- Kolve, E., R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta and A. Farhadi, “Ai2-thor: An interactive 3d environment for visual ai”, arXiv preprint arXiv:1712.05474 (2017b).
- Krishna, R., Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. 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).
- Krizhevsky, A., I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, *Communications of the ACM* **60**, 6, 84–90 (2017).
- Kulhánek, J., E. Derner, T. de Bruin and R. Babuška, “Vision-based navigation using deep reinforcement learning”, in “2019 European Conference on Mobile Robots (ECMR)”, pp. 1–8 (IEEE, 2019).
- Kulkarni, T. D., K. Narasimhan, A. Saeedi and J. Tenenbaum, “Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation”, in “Advances in neural information processing systems”, pp. 3675–3683 (2016).
- Laina, I., C. Rupprecht, V. Belagiannis, F. Tombari and N. Navab, “Deeper depth prediction with fully convolutional residual networks”, in “3D Vision (3DV), 2016 Fourth International Conference on”, pp. 239–248 (IEEE, 2016).
- Le, H. M., N. Jiang, A. Agarwal, M. Dudík, Y. Yue and H. Daumé III, “Hierarchical imitation and reinforcement learning”, arXiv preprint arXiv:1803.00590 (2018).
- Lea, C., R. Vidal and G. D. Hager, “Learning convolutional action primitives for fine-grained action recognition”, in “Robotics and Automation (ICRA), 2016 IEEE International Conference on”, pp. 1642–1649 (IEEE, 2016).
- Lee, A. X., A. Nagabandi, P. Abbeel and S. Levine, “Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model”, arXiv preprint arXiv:1907.00953 (2019).
- Lei, J., X. Ren and D. Fox, “Fine-grained kitchen activity recognition using rgb-d”, in “Proceedings of the 2012 ACM Conference on Ubiquitous Computing”, pp. 208–211 (ACM, 2012).
- Levy, A., G. Konidaris, R. Platt and K. Saenko, “Learning multi-level hierarchies with hindsight”, arXiv preprint arXiv:1712.00948 (2017a).
- Levy, A., R. Platt and K. Saenko, “Hierarchical actor-critic”, arXiv preprint arXiv:1712.00948 (2017b).

- Levy, A., R. Platt and K. Saenko, “Hierarchical reinforcement learning with hindsight”, arXiv preprint arXiv:1805.08180 (2018).
- Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, “Continuous control with deep reinforcement learning”, arXiv preprint arXiv:1509.02971 (2015).
- Liu, C. *et al.*, *Beyond pixels: exploring new representations and applications for motion analysis*, Ph.D. thesis, MIT (2009).
- Liu, F., C. Shen and G. Lin, “Deep convolutional neural fields for depth estimation from a single image”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 5162–5170 (2015).
- Long, J., E. Shelhamer and T. Darrell, “Fully convolutional networks for semantic segmentation”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 3431–3440 (2015).
- Ma, C.-Y., J. Lu, Z. Wu, G. AlRegib, Z. Kira, R. Socher and C. Xiong, “Self-monitoring navigation agent via auxiliary progress estimation”, in “International Conference on Learning Representations (ICLR)”, (2019).
- Mirowski, P., R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu *et al.*, “Learning to navigate in complex environments”, arXiv preprint arXiv:1611.03673 (2016).
- Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning”, in “International conference on machine learning”, pp. 1928–1937 (2016).
- Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, “Playing atari with deep reinforcement learning”, arXiv preprint arXiv:1312.5602 (2013).
- Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning”, *Nature* **518**, 7540, 529 (2015).
- Mousavian, A., A. Toshev, M. Fišer, J. Košecká, A. Wahid and J. Davidson, “Visual representations for semantic target driven navigation”, in “2019 International Conference on Robotics and Automation (ICRA)”, pp. 8846–8852 (IEEE, 2019).
- Nachum, O., S. Gu, H. Lee and S. Levine, “Near-optimal representation learning for hierarchical reinforcement learning”, arXiv preprint arXiv:1810.01257 (2018a).
- Nachum, O., S. S. Gu, H. Lee and S. Levine, “Data-efficient hierarchical reinforcement learning”, in “Advances in Neural Information Processing Systems”, pp. 3307–3317 (2018b).

- Nachum, O., H. Tang, X. Lu, S. Gu, H. Lee and S. Levine, “Why does hierarchy (sometimes) work so well in reinforcement learning?”, arXiv preprint arXiv:1909.10618 (2019).
- Nasiriany, S., V. Pong, S. Lin and S. Levine, “Planning with goal-conditioned policies”, in “Advances in Neural Information Processing Systems”, pp. 14843–14854 (2019).
- Nguyen, T.-L., D.-V. Nguyen and T.-H. Le, “Reinforcement learning based navigation with semantic knowledge of indoor environments”, in “2019 11th International Conference on Knowledge and Systems Engineering (KSE)”, pp. 1–7 (IEEE, 2019).
- Osa, T., V. Tangkaratt and M. Sugiyama, “Hierarchical reinforcement learning via advantage-weighted information maximization”, arXiv preprint arXiv:1901.01365 (2019).
- Parvaneh, A., E. Abbasnejad, D. Teney, Q. Shi and A. van den Hengel, “Counterfactual vision-and-language navigation: Unravelling the unseen”, in “Advances in Neural Information Processing Systems”, (2020).
- Pathak, D., P. Agrawal, A. A. Efros and T. Darrell, “Curiosity-driven exploration by self-supervised prediction”, in “International Conference on Machine Learning (ICML)”, vol. 2017 (2017).
- Popov, I., N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez and M. Riedmiller, “Data-efficient deep reinforcement learning for dexterous manipulation”, arXiv preprint arXiv:1704.03073 (2017).
- Qi, Y., Z. Pan, Y. Hong, M.-H. Yang, A. v. d. Hengel and Q. Wu, “Know what and know where: An object-and-room informed sequential bert for indoor vision-language navigation”, arXiv preprint arXiv:2104.04167 (2021).
- Qi, Y., Q. Wu, P. Anderson, X. Wang, W. Y. Wang, C. Shen and A. v. d. Hengel, “Reverie: Remote embodied visual referring expression in real indoor environments”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 9982–9991 (2020).
- Qiu, Y., A. Pal and H. I. Christensen, “Target driven visual navigation exploiting object relationships”, arXiv preprint arXiv:2003.06749 (2020).
- Rajeswaran, A., V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov and S. Levine, “Learning complex dexterous manipulation with deep reinforcement learning and demonstrations”, arXiv preprint arXiv:1709.10087 (2017).
- Redmon, J., S. Divvala, R. Girshick and A. Farhadi, “You only look once: Unified, real-time object detection”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 779–788 (2016).
- Ren, S., K. He, R. Girshick and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks”, *IEEE transactions on pattern analysis and machine intelligence* **39**, 6, 1137–1149 (2016).

- Rennie, S. J., E. Marcheret, Y. Mroueh, J. Ross and V. Goel, “Self-critical sequence training for image captioning”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 7008–7024 (2017).
- Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge”, *International Journal of Computer Vision* **115**, 3, 211–252 (2015).
- Savinov, N., A. Dosovitskiy and V. Koltun, “Semi-parametric topological memory for navigation”, arXiv preprint arXiv:1803.00653 (2018).
- Schulman, J., F. Wolski, P. Dhariwal, A. Radford and O. Klimov, “Proximal policy optimization algorithms”, arXiv preprint arXiv:1707.06347 (2017).
- Simonyan, K. and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, arXiv preprint arXiv:1409.1556 (2014).
- Sohn, S., J. Oh and H. Lee, “Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies”, in “Advances in Neural Information Processing Systems”, pp. 7156–7166 (2018).
- Song, D., N. Kyriazis, I. Oikonomidis, C. Papazov, A. Argyros, D. Burschka and D. Kragic, “Predicting human intention in visual observations of hand/object interactions”, in “Robotics and Automation (ICRA), 2013 IEEE International Conference on”, pp. 1608–1615 (IEEE, 2013).
- Stein, S. and S. J. McKenna, “Combining embedded accelerometers with computer vision for recognizing food preparation activities”, in “Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013), Zurich, Switzerland”, (ACM, 2013).
- Sutton, R. S. and A. G. Barto, *Reinforcement learning: An introduction* (MIT press, 2018).
- Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, “Going deeper with convolutions”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 1–9 (2015).
- Tan, H., L. Yu and M. Bansal, “Learning to navigate unseen environments: Back translation with environmental dropout”, in “NAACL-HLT”, (2019).
- Tu, S., “The dirichlet-multinomial and dirichlet-categorical models for bayesian inference”, Computer Science Division, UC Berkeley (2014).
- Van Hasselt, H., A. Guez and D. Silver, “Deep reinforcement learning with double q-learning”, arXiv preprint arXiv:1509.06461 (2015).
- Van Hasselt, H., A. Guez and D. Silver, “Deep reinforcement learning with double q-learning”, in “Thirtieth AAAI conference on artificial intelligence”, (2016).



- Wang, P., X. Shen, Z. Lin, S. Cohen, B. Price and A. L. Yuille, “Towards unified depth and semantic prediction from a single image”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 2800–2809 (2015).
- Wang, X., Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang and L. 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”, pp. 6629–6638 (2019).
- Wang, X., 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 “Proceedings of the European Conference on Computer Vision (ECCV)”, pp. 37–53 (2018).
- Wang, Y., X. Ye, Y. Yang and W. Zhang, “Collision-free trajectory planning in human-robot interaction through hand movement prediction from vision”, in “2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)”, pp. 305–310 (IEEE, 2017).
- Wang, Z., V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu and N. de Freitas, “Sample efficient actor-critic with experience replay”, arXiv preprint arXiv:1611.01224 (2016).
- Wu, Y., Z. Rao, W. Zhang, S. Lu, W. Lu and Z.-J. Zha, “Exploring the task cooperation in multi-goal visual navigation”, in “Proceedings of the 28th International Joint Conference on Artificial Intelligence”, pp. 609–615 (AAAI Press, 2019a).
- Wu, Y., Y. Wu, G. Gkioxari and Y. Tian, “Building generalizable agents with a realistic and rich 3d environment”, arXiv preprint arXiv:1801.02209 (2018).
- Wu, Y., Y. Wu, A. Tamar, S. Russell, G. Gkioxari and Y. Tian, “Bayesian relational memory for semantic visual navigation”, in “Proceedings of the IEEE International Conference on Computer Vision”, pp. 2769–2779 (2019b).
- Yang, W., X. Wang, A. Farhadi, A. Gupta and R. Mottaghi, “Visual semantic navigation using scene priors”, arXiv preprint arXiv:1810.06543 (2018).
- Yang, Y., C. Fermuller and Y. Aloimonos, “Detection of manipulation action consequences (mac)”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 2563–2570 (2013).
- Yang, Y., Y. Li, C. Fermuller and Y. Aloimonos, “Robot learning manipulation action plans by” watching” unconstrained videos from the world wide web”, in “Twenty-ninth AAAI conference on artificial intelligence”, (Citeseer, 2015).
- Yang, Y., C. Teo, H. Daumé III and Y. Aloimonos, “Corpus-guided sentence generation of natural images”, in “Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing”, pp. 444–454 (2011).

- Ye, X., 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 Robotics and Automation Letters* **4**, 4, 4003–4010 (2019a).
- Ye, X., Z. Lin, H. Li, S. Zheng and Y. 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)”, pp. 6857–6863 (IEEE, 2018).
- Ye, X., Z. Lin and Y. Yang, “Robot learning of manipulation activities with overall planning through precedence graph”, *Robotics and Autonomous Systems* **116**, 126–135 (2019b).
- Ye, X. and Y. Yang, “From seeing to moving: A survey on learning for visual indoor navigation (vin)”, arXiv preprint arXiv:2002.11310 (2020).
- Ye, X. and Y. Yang, “Efficient robotic object search via hiem: Hierarchical policy learning with intrinsic-extrinsic modeling”, *IEEE Robotics and Automation Letters* **6**, 3, 4425–4432 (2021a).
- Ye, X. and Y. Yang, “Hierarchical and partially observable goal-driven policy learning with goals relational graph”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, (2021b).
- Yi, K., J. Wu, C. Gan, A. Torralba, P. Kohli and J. Tenenbaum, “Neural-symbolic vqa: Disentangling reasoning from vision and language understanding”, *Advances in neural information processing systems* **31**, 1031–1042 (2018).
- You, Q., H. Jin, Z. Wang, C. Fang and J. Luo, “Image captioning with semantic attention”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 4651–4659 (2016).
- Zhu, F., Y. Zhu, X. Chang and X. Liang, “Vision-language navigation with self-supervised auxiliary reasoning tasks”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, (2020).
- Zhu, Y., R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei and A. Farhadi, “Target-driven visual navigation in indoor scenes using deep reinforcement learning”, in “2017 IEEE international conference on robotics and automation (ICRA)”, pp. 3357–3364 (IEEE, 2017).