Hierarchical and Partially Observable Goal-driven Policy Learning with Goals Relational Graph (Supplementary Material)

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

A. Experiments in the Grid-world Domain

A.1. Network Architectures

The detailed network architectures of all the baseline methods and our method are described in the following.

- DQN. The vanilla DQN implementation that takes the observation of shape $7 \times 7 \times 2$ as the input in which the first channel represents the obstacles and the second channel denotes the designated goal position. The input is fed into three convolutional layers with 1, 50 and 100 filters of kernel size 1×1, 3×3 and 3×3 respectively where a 2×2 max-pooling is attached after the second and the third layer. It is further followed by two fully-connected layers with 100 hidden units and 4 outputs respectively. Each output corresponds to a Q-value $Q(s, g, a)$ of an action.

  Since the goal may not be observable all the time, i.e. the second channel of the input may not have the goal information, we also explore the implementation DQN_ONEHOT where we specify the goal with the 16 dimensional one-hot vector and we concatenate it with the input vector of the first fully-connected layer in the method DQN, and the implementation DQN_FULL similar to the DQN_ONEHOT while we take the full observation of shape $7 \times 7 \times 17$ as the input. The first channel of the input is the obstacle map and each of the remaining 16 channels denotes corresponding goal position. We find both DQN_ONEHOT and DQN_FULL perform worse than DQN as Table 1 shows. Therefore, we compare our method with DQN.

- H-DQN. The hierarchical method where the high-level network takes the full observation of shape $7 \times 7 \times 17$ as the input and consists of three convolutional layers (1 filter of kernel size 1×1, 50 and 100 filters of kernel size 3×3 with 2×2 max-pooling after the second and the third layers), and two fully-connected layers with 100 hidden units and 17 outputs respectively. The goal that in the form of the 16 dimensional one-hot vector is concatenated with the hidden vector before inputting to the first fully-connected layer, and the 17 outputs are the Q-values $Q_h^e(s, g, sg)$ for all possible sub-goals including the back-up "random" sub-goal. The low-level network is exactly same as the method DQN in which the second channel of the input represents the position of the proposed sub-goal.

- OURS. The high-level network of our method HRL-GRG is similar to the method DQN while the second channel of the input denotes the position of a candidate sub-goal (a goal that is observable) and only a single output is generated to represent the Q-value $Q_h^e(s, g, sg)$ for the input sub-goal. Our low-level network is exactly same as the method DQN and the low-level network of H-DQN.

A.2. Training Protocols and Hyperparameters

For all the networks, we adopt the Double DQN [3] technique and we train all the methods on the 100 training grid-word 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 2. We evaluate all the methods on the 20 testing grid-world maps over 5 different random seeds, namely 1, 5, 13, 45 and 99.

Table 1: The performance of DQN vs DQN_ONEHOT and DQN_FULL on the unseen gird-word maps.

Method	Seen Goals			Unseen Goals			Overall		
	SR↑	AS / MS↓	SPL↑	SR↑	AS / MS↓	SPL↑	SR↑	AS / MS↓	SPL↑
DQN	0.20	20.28 / 5.47	0.13	0.20	11.90 / 4.10	0.15	0.32	16.23 / 5.71	0.23
DQN_ONEHOT	0.03	50.85 / 6.47	0.00	0.05	26.86 / 4.10	0.02	0.03	36.66 / 3.13	0.01
DQN_FULL	0.01	24.85 / 3.05	0.00	0.05	31.53 / 5.75	0.03	0.05	23.55 / 3.15	0.03

Table 2: Hyperparameters of all the methods for the grid-world domain.

Hyperparameter	Description	Value
γ	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^l_{max}	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
$\boldsymbol{\alpha_{ij}}$	The hyperparameter of our GRG $(\alpha_{ij,1}, ..., \alpha_{ij,10}, \alpha_{i,j,11})$	$(0, ..., 0, 1)$

A.3. Qualitative Results

We show some qualitative results performed by our method on the unseen grid-world maps to achieve both seen goals and unseen goals in Figure 1. The results shall be better viewed in the supplementary videos.

B. Experiments of the Robotic Object Search

B.1. Experiments on AI2-THOR [2]

B.1.1 Network Architecture

We take the first-person view semantic segmentation and the depth map of window size 30×30 as the agent's pre-processed observation. We concatenate 4 history observations to input to our model. Specifically, the high-level network of our method HRL-GRG takes a sub-goal specified semantic segmentation of size $4 \times 30 \times 30 \times 1$ and the depth map of the same size as the inputs. The semantic segmentation and the depth map are first concatenated to size $4 \times 30 \times 30 \times 2$ and then fed into two convolutional layers with 50 and 100 filters respectively of kernel size 3×3 where a 2×2 max-pooling is attached after each layer. The results further pass through two fully-connected layers with 256 hidden units and 1 output respectively. The output is the predicted Q-value of the input sub-goal.

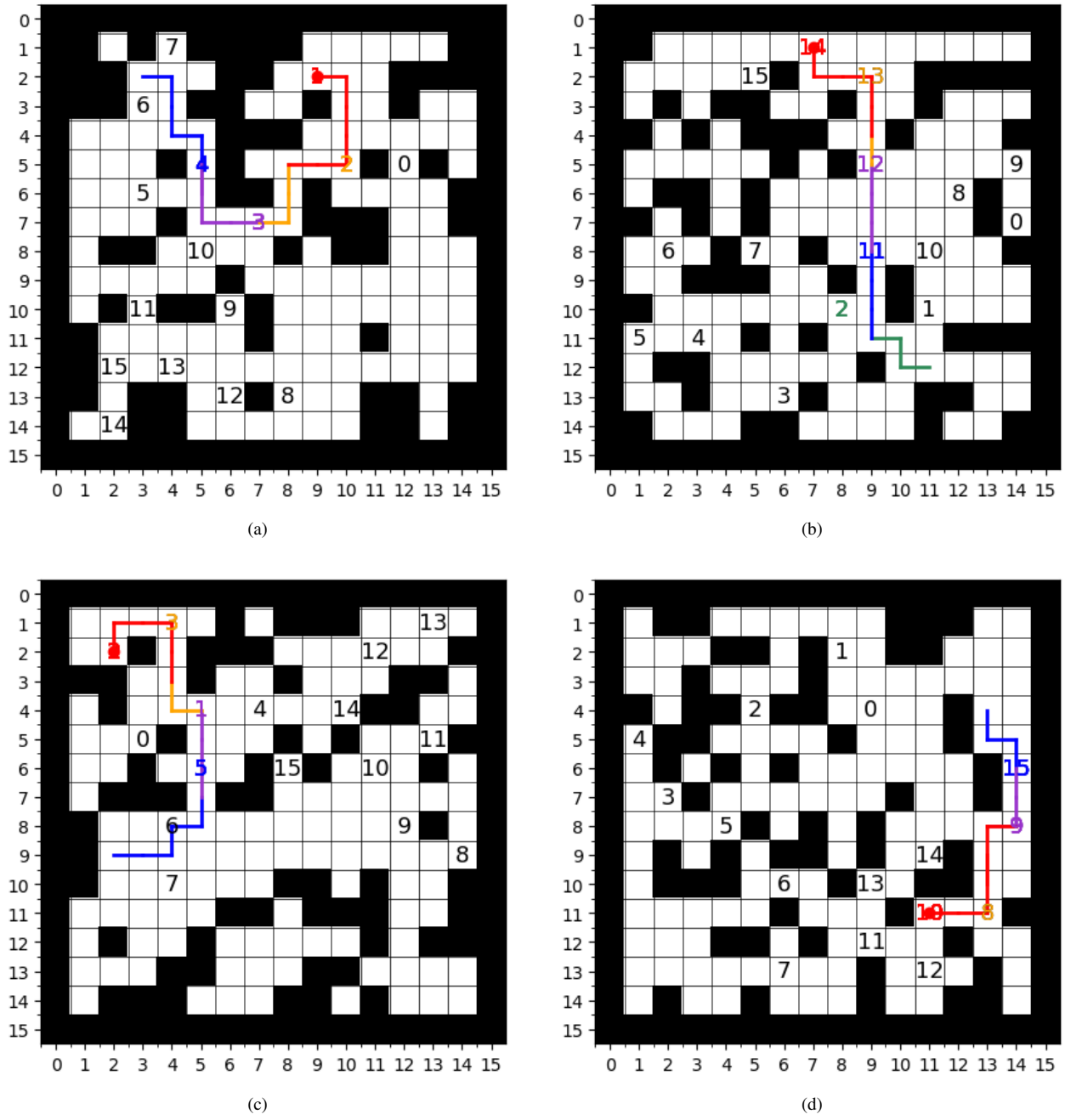
The low-level network of our method HRL-GRG takes the same inputs as the high-level network. For each input stream, our low-level network first flattens it and project it to a 256 dimensional vector. The two 256 dimensional vectors are then concatenated and projected to a joint 256 dimensional vector before passing through two branches. Each branch has two fully-connected layers with 20 hidden units in the first layer. The first branch outputs a 4 dimensional vector which is further converted to a probability distribution over the 4 actions using the softmax function, and the second branch outputs a single state value. All the hidden fully-connected layers are activated by the ReLU function.

B.1.2 Training Protocols and Hyperparameters

We follow the same experimental setting in [5] (without "stop" action). In addition, we adopt the Double DQN [3] 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 A.2. All hyperparameters are summarized in Table 5 and the detailed results are reported in Table 3.

B.1.3 Qualitative Results

Figure 2 shows some examples of how our method searches for unseen objects in unseen AI2-THOR [2] scenes. The results shall be better viewed in the supplementary videos.



(a) (b) (c) (d)

Figure 1: 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.

B.2. Experiments on House3D [4]

B.2.1 Data Pre-processing

In House3D simulation, the environments and goals we adopt for both the single environment setting and the multiple environments setting are shown in Table 4. For each environment, we consider discrete actions for the agent to navigate. Specifically, the agent moves forward / backward / left / right 0.2 meters, or rotates 90 degrees at each time step, which discretizes the environment into a certain num-

Table 3: The performance of SCENE PRIORS [5] (without stop action) and our HRL-GRG in performing robotic object search task on AI2-THOR [2]. (A: performance improvement; B and C: performance of the method and the RANDOM method.)

+A (B - C)		Seen Goals		Unseen Goals	
		SR↑	SPL↑	SR↑	SPL↑
Seen Env.	SCENE PRIORS	+0.25 (0.62 - 0.37)	+0.16 (0.26 - 0.10)	+0.08 (0.48 - 0.40)	+0.07 (0.18 - 0.11)
	HRL-GRG	**+0.37** (0.74 - 0.37)	**+0.24** (0.34 - 0.10)	**+0.33** (0.73 - 0.40)	**+0.23** (0.34 - 0.11)
Unseen Env.	SCENE PRIORS	+0.18 (0.56 - 0.38)	+0.11 (0.21 - 0.10)	+0.12 (0.49 - 0.37)	+0.06 (0.16 - 0.10)
	HRL-GRG	**+0.33** (0.71 - 0.38)	**+0.21** (0.31 - 0.10)	**+0.38** (0.75 - 0.37)	**+0.23** (0.33 - 0.10)

ber of reachable locations. We collect the first-person view RGB images as well as the corresponding the ground-truth semantic segmentations and depth maps at every locations of 100 preserved environments (other than those shown in Table 4) as the training data to train the encoder-decoder model [1] to predict both the semantic segmentations and the depth maps from the first-person RGB images. For our robotic object search task, we resize the both predictions to the size 10×10 and take them as the agent's observation. We consider an object that is unique in an environment as a valid target object for the agent to search, and we define the goal locations as the 5 locations that yields the observations with the largest target object area.

B.2.2 Network Architectures

For all the methods, we concatenate 4 history observations for the agent to make a decision. We describe the architectures of all methods in details as follows.

- DQN. The vanilla DQN implementation that takes the semantic segmentation of size $4 \times 10 \times 10 \times 78$ and the depth map of size $4 \times 10 \times 10 \times 1$ as the inputs. The segmentation input is first passed through a convolutional layer with 1 filter of kernel size 1×1 to reduce its channel size to 1. Then for both the segmentation input and the depth input, we flatten each of which to a 400 dimensional vector and project it down to a 256 dimensional vector through a fully-connected layer respectively. The two 256 dimensional vectors, as well as the 78 dimensional one-hot vector representing the target object are further concatenated and fed into two fully-connected layers with 256 hidden units and 6 outputs as the Q-values of 6 actions. Each hidden fully-connected layer is activated by the ReLU function.

- A3C. The vanilla A3C implementation takes the target object specified channel of the semantic segmentation that has size $4 \times 10 \times 10 \times 1$ and the depth map of size $4 \times 10 \times 10 \times 1$ as the inputs. For each input stream, we flatten it and project it to a 256 dimensional vector. The two 256 dimensional vectors are further concatenated and passed through two branches. Each branch has two fully-connected layers with 20 hidden units in the first layer. The first branch further projects the 20 dimensional vector to 6 outputs which are converted to probabilities of 6 actions using the softmax function, and the second branch outputs a single state value. All the hidden fully-connected layers are activated by the ReLU function.

- HRL. The high-level network of HRL is similar to the method DQN while it outputs 78 values representing the Q-values of the 78 sub-goals. The low-level network is exactly same as the method A3C.

- OURS. The high-level network of our method HRL-GRG is similar to the method A3C while the first branch is removed and only the second branch is in place to output a Q-value of the input sub-goal. The low-level network is exactly same as the method A3C.

B.2.3 Training Protocols and Hyperparameters

Similar to that in the grid-world domain, we also adopt the Double DQN [3] 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 A.2 and we summarize the hyperparameters in Table 5 .

B.2.4 Qualitative Results

Figure 3 shows some qualitative results of our method for the robotic object search task on House3D [4]. The agent can only access the first-person view RGB images while the top-down 2D maps are placed for better visualization. The results shall be better viewed in the supplementary videos.

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In

Table 4: The environments and goals we adopt for the robotic object search task on House3D [4] .

Single Environment	Seen Env	5cf0e1e9493994e483e985c436b9d3bc	Seen Goals	music, television, heater, stand, dressing table, table
			Unseen Goals	bed, mirror, ottoman, sofa, desk, picture frame
Multiple Environments	Seen Envs	5cf0e1e9493994e483e985c436b9d3bc 0c9a666391cc08db7d6ca1a926183a76 0c90efff2ab302c6f31add26cd698bea 00d9be7210856e638fa3b1addf2237d6	Goals	sofa, bed, television, tv stand, toilet, bathtub
	Unseen Envs	07d1d46444ca33d50fbcb5dc12d7c103 026c1bca121239a15581f32eb27f2078 0147a1cce83b6089e395038bb57673e3 0880799c157b4dff08f90db221d7f884	Goals	sofa, bed, dressing table, mirror, ottoman, music

Table 5: Hyperparameters of all the methods for the robotic object search task.

Hyperparameter	Description	Value
γ	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
β	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^l_{max}	The maximum steps that low-level network can take in AI2-THOR / House3D	10 / 50
N_{max}	The maximum steps that each method can take in AI2-THOR / House3D	[5] / 1000
optimizer	Optimizer for all the networks	RMSProp
$\boldsymbol{\alpha_{ij}}$	The hyperparameter of our GRG $(\alpha_{ij,1}, ..., \alpha_{ij,10}, \alpha_{i,j,11})$	$(0, ..., 0, 1)$

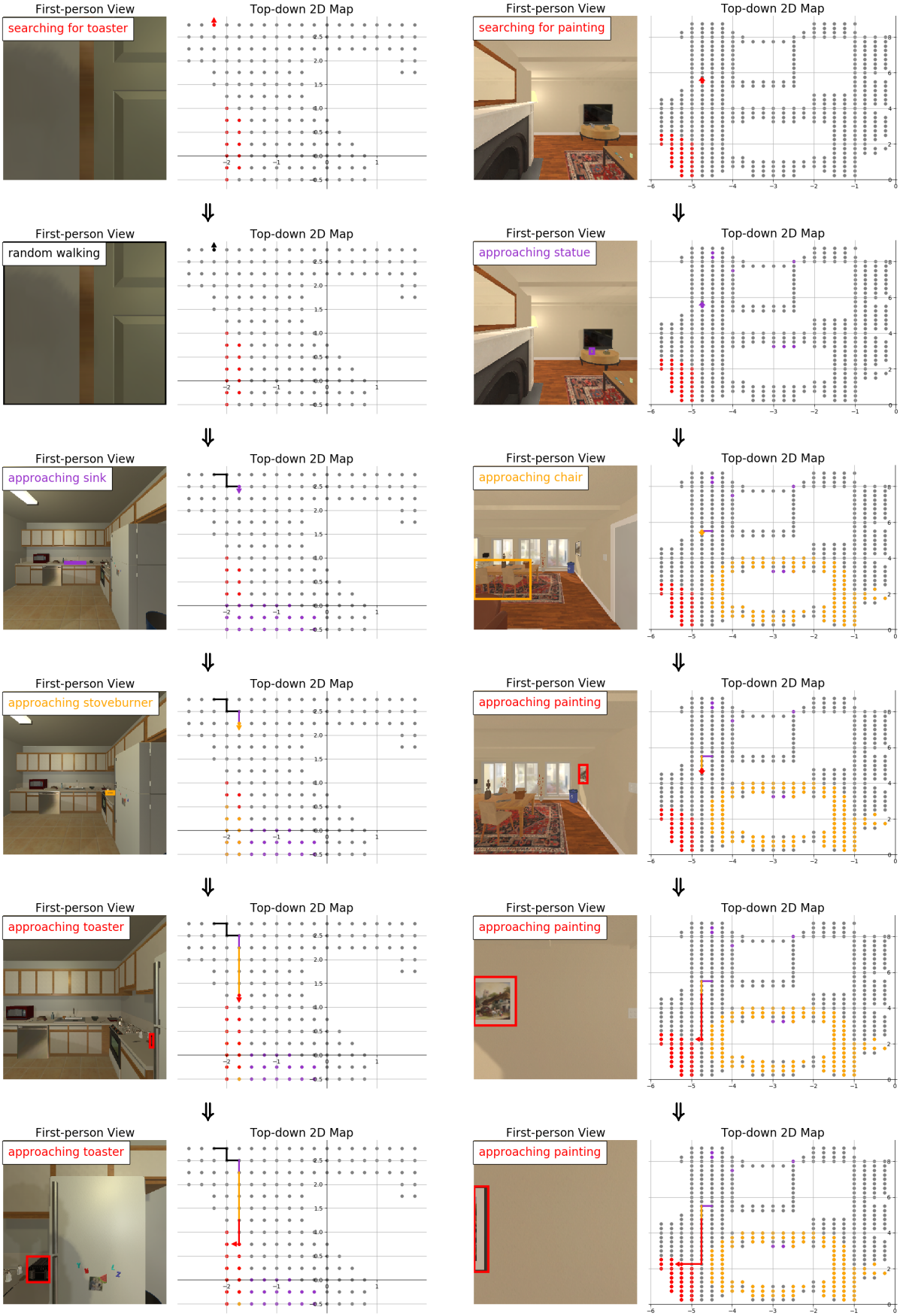
Proceedings of the European conference on computer vision (ECCV), pages 801–818, 2018. 4

[2] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 2, 4, 7

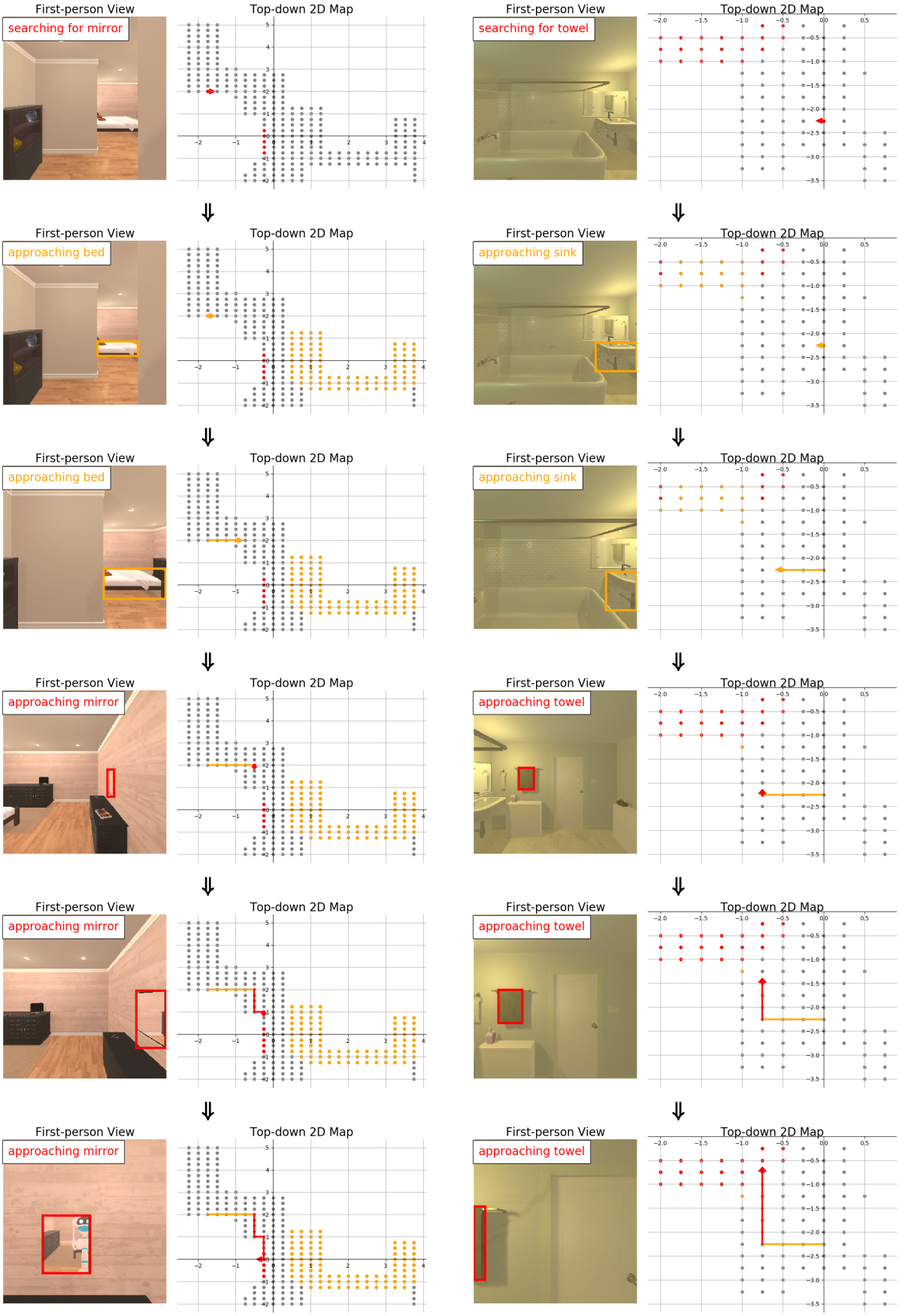
[3] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016. 1, 2, 4

[4] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018. 3, 4, 5, 9

[5] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018. 2, 4, 5



(a) kitchen (toaster) (b) living room (painting)

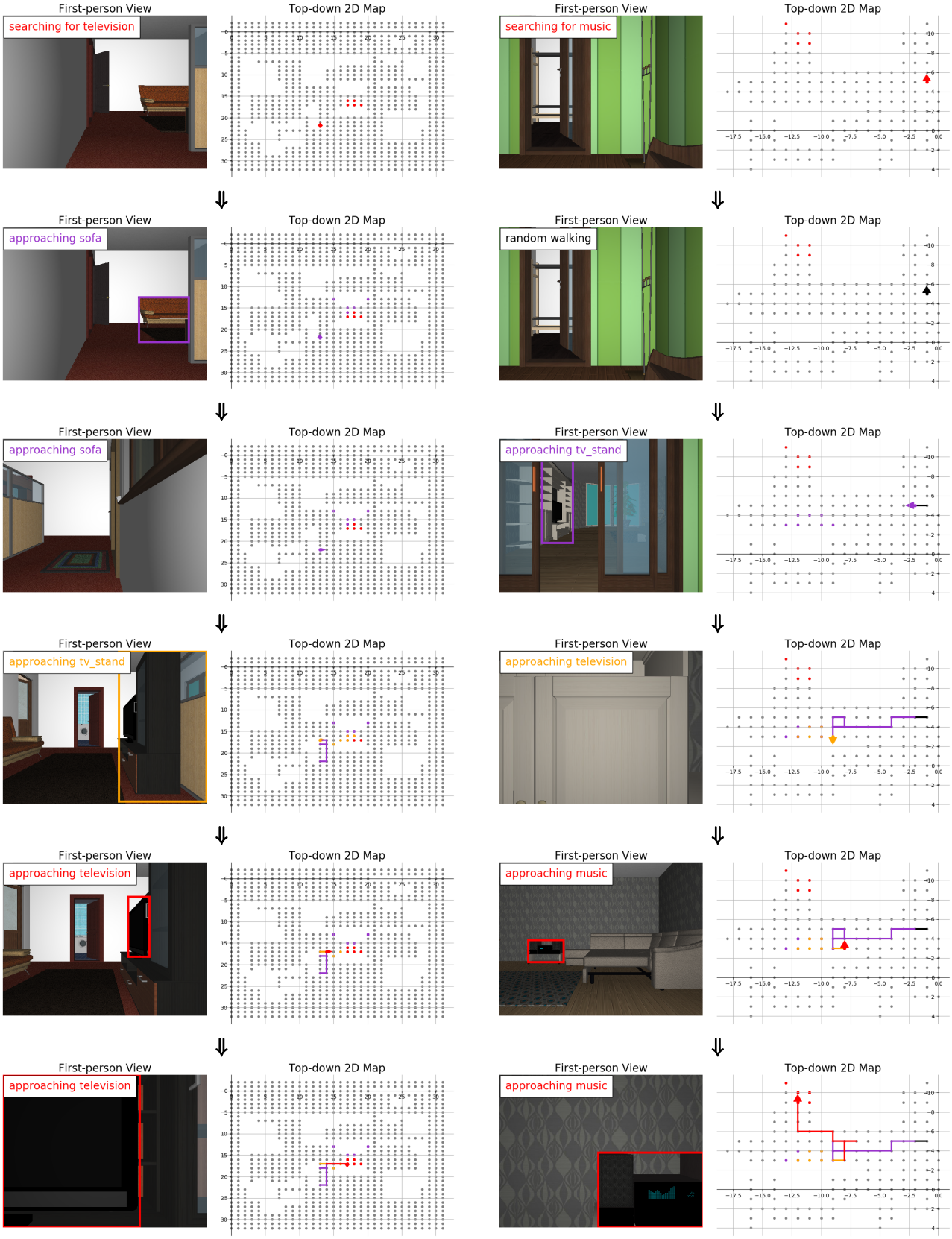


(c) bedroom (mirror) (d) bathroom (towel)

Figure 2: Trajectories generated by our method for the robotic object search task on AI2-THOR [2].

(a) seen environment seen goal

(b) seen environment unseen goal



(c) unseen environment seen goal

(d) unseen environment unseen goal

Figure 3: Trajectories generated by our method for the robotic object search task on House3D [4].