Generative Meta-Adversarial Network for Unseen Object Navigation ——Supplementary Materials——

Sixian Zhang[1,2], Weijie Li[1,2], Xinhang Song[1,2], Yubing Bai[1,2], and Shuqiang Jiang[1,2]

[1] Key Lab of Intelligent Information Processing Laboratory of the Chinese Academy of Sciences (CAS), Institute of Computing Technology (ICT), Beijing
[2] University of Chinese Academy of Sciences (UCAS), Beijing

1 The Pre-training of Feature Generator

In recent generalized zero-shot learning (GZSL) methods [13,14,8], training a generator with WGAN [5] has been proved to be an effective way to synthesize discriminative features for unseen objects. Inspired by those works, we also utilize WGAN to pre-train our feature generator (FG). Then the pre-trained FG can generate features of target objects to guide the navigation agent.

The pre-training of the FG aims at obtaining $G\left(SE\left(y\right),z\right)$ that synthesizes the object features $x=G\left(SE\left(y\right),z\right)$ of the target category y based on its semantic embedding $SE\left(y\right)$ and a random Gaussian noise z. The $SE\left(\cdot\right)$ is an embedding module that converts the object category into its class-specific semantic vector. The dataset for pre-training is $D_{train}=\{(x^s,SE\left(y^s\right))\,|x^s\in\mathcal{X}^s,y^s\in\mathcal{Y}^s\}$, where the $\mathcal{X}^s$ is the set of seen object features (extracted by ResNet18) and the $\mathcal{Y}^s$ is the set of seen object labels. The collection process of D_{train} has been introduced in the main text. Although the generator G is pre-trained with seen objects to generate seen object features conditioned on the semantic embedding, the well pre-trained generator can transfer the ability of synthesizing features to unseen objects (i.e. generating unseen objects features via their semantic embeddings).

The generator is pre-trained together with a discriminator D^{pre} via adversarial learning. Note that in the conventional works [5], the discriminator is written as D by default. In order to differentiate the pre-trained discriminator from our environmental meta-discriminator (EMD) in the main text, the discriminator in the pre-training is written as D^{pre}. In adversarial learning, the discriminator D^{pre} tries to distinguish the real features $x^s\in\mathcal{X}^s$ and the generated features x conditioned on the semantic embedding $SE\left(y\right)$, while the generator G tries to fool the discriminator by generating more realistic features. Such a pair of adversarial learners G and D^{pre} are learned by optimizing $\underset{G}{min}\underset{D^{pre}}{max}\mathcal{L}_{WGAN}$, where the optimization objective $\mathcal{L}_{WGAN}$ is given by

$$\begin{aligned}\mathcal{L}_{WGAN}=&\mathbb{E}[D^{pre}(x^s,SE(y))]-\mathbb{E}[D^{pre}(x,SE(y))]\\&-\lambda\mathbb{E}[(\|\nabla_{\hat{x}}D^{pre}(\hat{x},SE(y))\|_2-1)^2]\end{aligned}\tag{1}$$

Table 1. Distribution of the target objects in different scene types in AI2THOR [6].

Scene Type	Seen Objects	Unseen Objects
Kitchen	Microwave, Fridge, Bowl, Cabinet, Garbage Can, Pan, Toaster	Coffee Machine, Drawer, Kettle, Stove Burner, Pot, Faucet, Plate
Living Room	Garbage Can, Box, Pillow, Laptop, Shelf, Television, Side Table, Sofa, Floor Lamp	Plate, Drawer, Arm Chair, Desk Lamp
Bedroom	Book, Chair, Mug, Alarm Clock	Desk Lamp, Arm Chair, Drawer
Bathroom	Bathtub, Sink, Light Switch, Shower Curtain	Drawer, Faucet, Shower Door, Toilet Paper, Toilet

Table 2. Distribution of the target objects in RoboTHOR [3].

Seen Objects	Unseen Objects
Book, Bowl, Chair, Mug, Side Table, Laptop, Box, Shelf, Television, Sofa	Arm Chair, Plate, Pot, Drawer

where $x = G\left(SE\left(y\right), z\right)$ is the generated features, $\hat{x} = \varepsilon x^{s} + \left(1 - \varepsilon\right) x$ with $\varepsilon \in U(0, 1)$ and λ is the penalty coefficient.

The first thing should be noticed that although the motivation of pre-training the FG is similar to some GZSL works [13,14,8], there are some differences in the way of training the generator. To ensure that the generated features are more suitable for the classification task, the conventional optimization objectives of GZSL methods usually introduce an additional prediction likelihood [13] or a reconstruction constraint [8,14] in the Eq. 1. While our FG pre-training only adopts the original $\mathcal{L}_{WGAN}$, the main improvements of our method (making the FG more suitable for navigation) are essentially achieved by the later FG adaptation with our EMD.

Another thing should be noticed that although the FG pre-training and the FG adaptation during navigation both utilize the adversarial learning, there are three differences between these two adversarial learning processes: 1) The generator's initial parameters are different. The FG pre-training begins with the random initialized parameters, while the FG adaptation starts with the trained parameters in FG pre-training; 2) The structure of the discriminator is different. The D^{pre} is a conditional discriminator (i.e. conditioned on the semantic embedding $SE(y)$ of the input object category), while EMD adopts a non-conditional discriminator. Besides, the number of hidden units is also different; 3) The goal of the adversarial learning is different. In FG pre-training, the goal of the adversarial learning is to let the generator learn the characteristics of objects conditioned on their semantic embeddings. While the adversarial learning in FG adaptation

(with EMD) aims at adapting the generator to learn the environmental features, so that the generated object features could obtain more environment information. Then the adapted generator could synthesize more precise features of the target for the agent, especially helpful in the unseen object navigation.

2 Navigation Target Objects

Tab. 1 and Tab. 2 demonstrate the split of seen and unseen target objects in AI2THOR and RoboTHOR, respectively. Considering that each room in AI2THOR belongs to a specific scene and some objects only appear in certain scenes (e.g. the toilet only appears in the bathroom), we set different target objects for different scene types. While in RoboTHOR, each apartment has consistent layout so that all apartments share the same split of seen and unseen target objects. We use the seen objects for training, and test our model with both seen and unseen objects. Moreover, we adjust the initial distribution of objects in the simulator to avoid dense objects areas, and remove the unseen objects during the model training.

3 Semantic Embedding

Since the feature generation is based on the semantic embedding of the object category, semantic embedding plays a vital role in transferring the knowledge of feature generation from seen to unseen objects. Currently, two related domains (i.e. object navigation and generalized zero-shot learning) adopt different semantic vectors to embed the object categories. Most object navigation researches typically utilize the Glove vector [10], while most generalized zero-shot learning (GZSL) researches employ the attribute vector. We investigate such two widely used semantic embeddings as follows.

3.1 The Collection of Attribute Vector

The GZSL works generally synthesize the visual features based on the annotated attribute vector, which converts the object category into a class specific semantic vector. The attribute annotations are easily acquired in some GZSL datasets (e.g. CUB [11], FLO [9]). However, attribute annotations for object categories are not directly provided in the existing navigation datasets (e.g. AI2THOR [6], RoboTHOR [3]). Therefore, we collect the

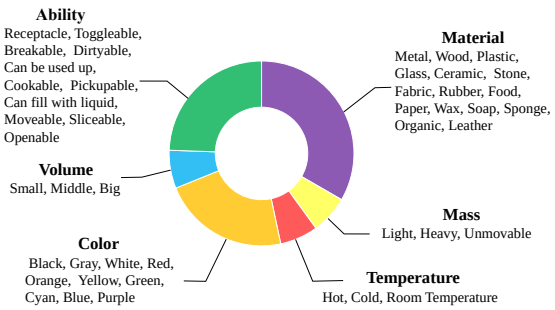

Fig. 1. The object attributes in the AI2THOR and RoboTHOR simulator.

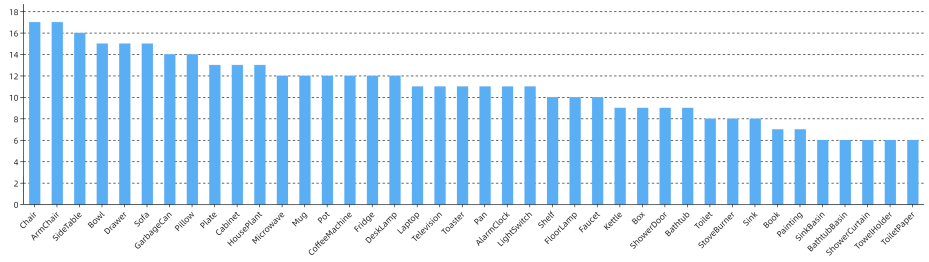

Fig. 2. The number of attributes for each object class. We only illustrate the object categories involved in our navigation task.

attributes for all object categories in AI2THOR and RoboTHOR simulator, and annotate the attribute vector as the semantic embedding.

We consider 45 common attributes as shown in Fig. 1. Through interacting with all objects in the simulator, we could obtain a 45-dimensional bag-of-words vector for each object instance to represent its owned attributes. Then, from instance-level to class-level, for each object class, we can get an average attribute vector by averaging all related (belonging to such object class) instances' attribute vector. In this way, we statistically collect the attribute vectors that reflect the general characteristic of each object category, i.e. each object category corresponds to a 45-dimensional attribute vector, where each dimension represents the probability of containing such attribute.

The number of attributes for each object class is illustrated in Fig. 2. These attributes come from different aspects (ability, material, volume, mass, color, temperature), which are the basic components to describe an object. According to the statistics, an object category covers 11.36 attributes in the average case, and could still cover 6 attributes even in the lowest case (e.g. shower curtain, toilet paper). Such attribute distribution guarantees adequate information to represent each object category. Besides, each dimension of the attribute vector is a constant probability value from 0 to 1. Therefore, although the considered attribute types (i.e. 45) are limited, the attribute vectors can theoretically represent and distinguish infinite object categories. Each object involved in our experiments has a unique attribute vector as semantic embedding.

3.2 Comparisons between Glove Vector and Attribute Vector

As shown in Tab. 3, We compare the navigation performance of our GMAN with different semantic embeddings (Glove vector and attribute vector). For navigating to the unseen objects, our GMAN with the Glove vector has a performance similar to the baseline (i.e. A3C model), while the GAMN with the attribute vector outperforms the baseline by a large margin. For navigating to the seen objects, our GMAN with the Glove vector brings some improvements than the baseline. However, such improvements are still less significant than those achieved by using the attribute vector.

Table 3. The comparisons of the Glove vector and attribute vector. In order to more intuitively observe the differences between such two semantic embeddings, we additionally list the A3C model here as a comparison.

Models	Unseen Objects				Seen Objects			
	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)
A3C baseline	21.50 1.23	9.36 0.66	8.56 0.15	1.09 0.01	31.37 1.00	14.19 0.30	9.36 0.18	1.06 0.01
GMAN-Glove [10]	21.60 1.84	8.55 0.64	8.78 0.71	1.08 0.03	33.83 0.60	14.37 0.36	10.51 0.01	1.03 0.01
GMAN-Attribute	**28.03** 0.91	**13.02** 0.14	8.71 0.52	**1.18** 0.02	**39.30** 0.87	**15.61** 0.14	10.46 0.17	**0.98** 0.02

The experimental results indicate that the visual features generated via Glove vectors only have a little effect on the seen object categories that are pre-trained in advance, while transferring the generation knowledge from seen to unseen with the Glove vector is poor. On the contrary, using the attribute vector as semantic embedding, our GMAN could generate more informative visual features for both seen and unseen objects, thus achieving better performance on both cases. We compare the two semantic embeddings and conclude that attributes vectors could more accurately reflect the visual characteristics of the object (e.g. ability, material, volume, mass, color, temperature), thus more helpful for transferring the knowledge (visual features generation based on semantic embeddings) from seen to unseen objects. As a result, we employ the attribute vector as the semantic embedding for the target object categories.

4 More Evaluations

4.1 Datasets

We employ two editable simulators AI2THOR [6] and RoboTHOR [3] for evaluations. AI2THOR provides 120 rooms in 4 types: kitchen, living room, bedroom and bathroom. Each type scene consists of 30 rooms, and we choose 20 for training, 5 for validation and 5 for testing. Different from AI2THOR where each environment contains only one scene category, RoboTHOR contains a variety of scene categories in each environment. We define each environment in AI2THOR as *room* and RoboTHOR as *apartment*. RoboTHOR consists of 75 apartments for training and validation, while the testing data is not in public. Thus, we choose 60 apartments for training, 5 for validation and 10 for testing.

4.2 Evaluation Metrics

We evaluate the models using Success Rate (SR), Success weighted by Path Length (SPL), Exploration Area (EPA) and Distance to Success (DTS). The detailed definitions of these evaluation metrics are as follows:

SR. The SR is used to evaluate the success rate of the agent in finding the target object (i.e. the agent finally gets close to the target object within a threshold of distance (i.e. $1m$) and the target is visible in agent's egocentric

Table 4. Comparisons with the image-goal navigation and our object-goal navigation.

Settings	Unseen Objects				Seen Objects			
	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)
Image-goal	**29.17** 0.32	**13.69** 0.36	8.51 0.46	**1.15** 0.04	30.63 0.31	12.94 0.47	11.13 0.50	0.99 0.01
Object-goal (ours)	28.03 0.91	13.02 0.14	8.71 0.52	1.18 0.02	**39.30** 0.87	**15.61** 0.14	10.46 0.17	**0.98** 0.02

view), which is defined as $SR = \frac{1}{N}\sum_{i=1}^{N}\mathcal{S}_i$, where N is the number of total episodes and $\mathcal{S}_i$ is a binary indicator to represent whether the i-th episode is successful.

SPL. The SPL is an improved evaluation metric that considers both the success rate and the path length. It is formulated as $SPL = \frac{1}{N}\sum_{i=1}^{N}\mathcal{S}_i\frac{L_i^*}{max(L_i,L_i^*)}$, where L_i represents the actual path length and L_i^* is the shortest path length provided by the simulator.

EPA. The EPA is defined as the percentage of explored area, which is computed by the ratio of explored points to all reachable points.

DTS. The DTS records the average distance of the agent towards the target at the end of episode. It is formulated as $DTS = \frac{1}{N}\sum_{i=1}^{N} max\left(\|l_i - l_o\|_2 - \xi, 0\right)$, where l_i is the end location of i-th episode, l_o is the target object's location and $\xi = 1m$ is the success threshold. The optimal value of DTS is 0.

4.3 Comparisons with Image-goal Navigation

As to the unseen object navigation, our task setting follows the typical Object-Goal navigation which provides the object category as the goal. Then the agent learns to transfer the navigation ability from seen to unseen objects based on the semantic embedding of the object categories. However, another setting following ImageGoal navigation can also be applied to unseen object navigation, which provides an egocentric image [16] or a panoramic image [2] around the unseen object as the goal. As shown in Tab. 4, we compare these two settings in unseen object navigation. For a fair comparison, we representatively choose the image-goal navigation method [16] since both of us utilize the egocentric image as the visual input. Besides, all methods are trained with equal amounts of episodes.

For the unseen objects, since our method takes the generated visual features as a clue for navigation, our performance is reasonably worse than that of the image-goal method, which directly provides the ground truth image as the target. Nevertheless, the performance gaps are not significant. However, for the seen objects, our method significantly outperforms the image-goal method [16]. We analyze the experimental results and infer that, for image-goal setting, the visual features of the provided image usually contain both the target object and the surrounding background (where the proportion of the background is usually relatively larger). For our object-goal setting, the generated features focus more on the target itself (based on the semantic embedding) even though the generator would be optimized (only within a few iterations) by EMD to introduce some

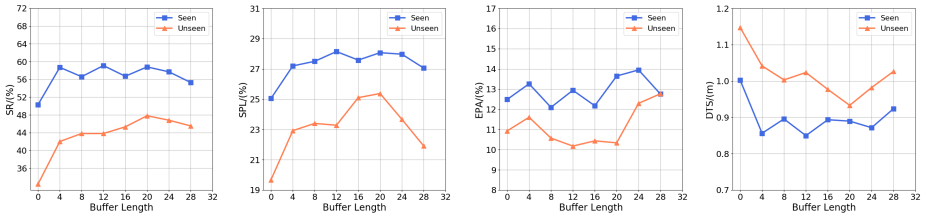

Fig. 3. The ablations on the buffer length k. We evaluate the impact of different buffer length k on the navigation metrics SR, SPL, EPA and DTS.

environmental features. Thus when navigating to unseen objects, the effects of our features ("imaginal" but more focusing) and the image features ("true" but more distracting) are similar. However, when navigation to seen objects, our features ("well-trained" and focusing) are more effective than the image features ("true" but distracting). As a result, our method outperforms the image-goal method by a large margin in seen objects and achieves comparable performance in unseen objects.

In summary, the advantages of our object-goal settings compared to the image-goal navigation are as follows: **1) More user-friendly target settings.** In our settings, the user only needs to provide an object category no matter whether it belongs to seen or unseen object categories. However, in image-goal setting, the user needs to obtain an image of the target beforehand. Although the images are easily acquired in the simulators with the oracle vision, it is impractical and inconvenient in real-world applications because the user must first locate the object and take a photo before demanding the agent to find it. **2) Better performances on averaging both seen and unseen objects.** Under equal amounts of training episodes, our method achieves better performance on seen objects and comparable performance on unseen objects. Thus our setting is an averagely better choice considering both seen and unseen objects.

4.4 The Ablations on the Buffer Length

In order to explore the optimal hyper-parameters for the best performance, we choose the A3C† baseline to evaluate the impact of different buffer length k, as illustrated in Fig. 3. Note that these ablations are conducted in the validation set. When the buffer length is 0, the GMAN† degenerates into the substitute method without the environmental meta-discriminator (EMD) module (i.e. line 5 of Tab. 2 in main text). The impacts of the buffer length on SR, SPL and DTS are basically same. For seen objects, the buffer length generally has a weak effect on SR, SPL and DTS metrics since the blue lines change steadily. However, for unseen objects, the impact of the buffer length is more obvious with the orange lines fluctuate. The navigation performances on unseen objects in SR, SPL and DTS metrics all indicate the same trend (getting better gradually and then getting worse with the buffer length k growing). We infer that as the buffer

Table 5. Comparisons with the related works for navigation in **seen** environments on AI2THOR simulator.

Method	Unseen Objects				Seen Objects			
	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)
Random	6.70 0.17	3.58 0.42	4.01 0.03	1.57 0.02	6.37 0.25	3.63 0.30	3.98 0.06	1.56 0.04
A3C	20.47 0.93	10.20 0.62	6.60 0.10	1.32 0.01	71.17 0.42	43.65 0.56	7.77 0.16	0.67 0.01
SP [15]	21.13 0.25	10.53 0.05	6.93 0.53	1.25 0.11	72.50 0.92	**44.32** 0.76	8.10 0.42	0.66 0.01
SAVN [12]	15.17 1.10	5.35 0.51	7.84 0.82	1.38 0.04	80.17 0.38	41.83 0.05	9.76 0.04	0.48 0.01
EOTP [7]	14.83 0.95	5.25 0.55	7.71 0.68	1.39 0.05	**80.83** 0.97	42.17 0.56	9.93 0.26	0.47 0.01
GMAN (ours)	**29.63** 0.88	**16.13** 0.41	7.70 0.36	**1.23** 0.02	80.20 0.70	42.25 0.11	9.77 0.16	**0.47** 0.01
A3C†	34.37 1.37	19.69 0.74	10.02 0.46	1.23 0.05	74.53 0.21	48.24 0.13	7.42 0.05	0.68 0.01
SP†	37.70 1.01	20.36 0.41	10.69 0.70	1.16 0.03	77.83 0.61	48.57 0.45	7.75 0.56	0.69 0.01
SAVN†	41.87 0.38	21.17 0.82	15.46 0.18	1.08 0.02	81.20 1.73	43.16 0.76	12.85 0.11	0.53 0.02
EOTP†	38.17 0.40	19.71 0.32	11.80 0.64	1.13 0.02	83.00 0.40	51.44 0.23	8.86 0.06	**0.48** 0.01
GMAN† (ours)	**48.87** 0.38	**25.08** 0.81	14.31 0.19	**0.96** 0.02	**83.57** 0.21	**54.26** 0.90	9.18 0.36	0.51 0.01

length increases, the batch size for optimizing the discriminator in EMD will increase, which causes the discriminator to capture more general environment information without overfitting to a small amount of observations. However, a large buffer length will make it difficult to accumulate enough observations to activate the inner-loop. This will make the inner-loop unable to optimize the feature generator (FG) and the policy.

The EPA reflects the efficiency of navigation. Under the same SR, the lower EPA indicates the higher navigation efficiency. For seen objects, the change trend of EPA is consistent with that of SR, which indicates that the buffer length has little effect on the efficiency of seen object navigation. However, for unseen objects, when the buffer length is set to 4-20, as the buffer length increases, the performance increases in the SR while decreases in EPA, which indicates that the navigation efficiency is increasing during this period. We infer that as the buffer length increases, a wiser EMD (as described above) will be obtained. A wiser EMD will adapt the FG to synthesize more accurate features of the target object, thereby improving the efficiency of unseen object navigation. Besides, when the buffer length is larger than 20, the performance shows a downward trend.

Therefore, selecting a suitable buffer length for SR, SPL, DTS and EPA is actually a trade-off between a wiser EMD and more optimization for FG and policy. Based on the above experimental results and analysis in the validation set, we set the buffer length as $k = 20$.

4.5 Comparisons on Seen Environments

In the main paper, we report the experimental results on two cases: 1) test on unseen objects in unseen environments; 2) test on seen objects in unseen environments. As shown in Tab. 5 and Tab. 6, there are more experimental results on other cases: 3) test on unseen objects in seen environments; 4) test

Table 6. Comparisons with the related works for navigation in **seen** environments on RoboTHOR simulator.

Method	Unseen Objects				Seen Objects			
	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)
Random	2.27 0.35	1.07 0.14	6.43 0.17	2.56 0.04	2.47 0.32	1.11 0.12	6.52 0.13	2.44 0.02
A3C	13.30 1.91	8.18 1.86	7.04 0.51	2.53 0.05	32.33 0.67	20.05 2.31	8.17 0.08	2.16 0.01
SP [15]	12.03 0.76	6.15 0.30	7.30 0.12	2.53 0.02	33.37 0.50	20.18 0.41	8.08 0.02	2.02 0.04
SAVN [12]	12.43 0.45	7.54 0.40	8.40 0.42	2.43 0.06	50.37 0.25	29.57 0.47	9.35 0.05	1.64 0.05
EOTP [7]	12.10 0.26	7.44 0.26	8.26 0.20	2.49 0.10	50.83 0.67	29.77 0.25	9.61 0.24	1.60 0.01
GMAN (ours)	**13.87** 0.23	**7.70** 0.26	8.67 0.47	**2.30** 0.05	**50.27** 0.25	**29.43** 0.54	9.47 0.15	1.63 0.02
A3C†	15.73 0.40	10.62 0.32	7.64 0.05	2.48 0.01	32.73 0.72	22.77 0.11	8.32 0.10	2.18 0.01
SP†	17.07 0.75	11.95 0.87	7.91 0.26	2.25 0.20	35.07 0.72	24.10 0.52	9.15 0.17	1.96 0.06
SAVN†	29.80 1.25	15.83 0.81	10.24 0.08	2.04 0.06	**54.13** 0.64	31.60 0.84	12.22 0.16	**1.50** 0.01
EOTP†	28.83 0.75	15.24 0.45	10.15 0.13	2.08 0.06	53.50 1.17	31.02 0.51	12.07 0.47	1.53 0.04
GMAN† (ours)	**32.10** 0.30	**18.21** 0.09	10.32 0.01	**1.93** 0.02	53.93 0.25	**36.78** 0.28	8.67 0.12	1.61 0.05

on seen objects in seen environments. It should be noticed that the 4th case reflects the performance on the training data, which has little correlation with the model's generalization performance since there may be over-fitting problems. Therefore, we just list the results here and do not compare with this case.

For the 3rd case navigating to unseen objects in seen environments, both GMAN and GMAN† achieve better performance compared with the related work. Notably, the GMAN† outperforms the state-of-the-art by 7.00% in SR, 3.91% in SPL and -0.12m in DTS within the AI2THOR simulator and 2.30% in SR, 2.38% in SPL and -0.11m within the RoboTHOR simulator. The experimental results indicate that our method is still better than the related works for navigating to unseen objects in seen environments.

Additionally, we notice that the GMAN performs better in the 3rd case (unseen objects in seen environments) than the 1st case (unseen objects in unseen environments). This difference demonstrates that the familiar (seen) environments can bring more improvements, which also supports our motivation that the environment background is important for navigating to unseen objects.

4.6 Comparisons with Other Navigation Methods

There are several other related works [4,1] for object navigation. These works achieve satisfactory performance on seen object navigation by applying pre-trained visual modules such as object detection or instance segmentation to construct object relation graphs or semantic maps. These methods are inapplicable to unseen objects since unseen object detection or unseen instance segmentation is not supported . For a fair comparison with these methods, we first consider modifying them by adding attribute detection or attribute segmentation. However, unlike the object detection or instance segmentation where a region or a pixel in an image normally has a unique object or instance label, the region or pixel may have a variety of attribute labels because an object or instance has

Table 7. Comparisons with the relation-based method (e.g. ORG) and the map-based method (e.g. SemExp) in AI2THOR.

Settings	Unseen Objects				Seen Objects			
	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)	SR↑ (%)	SPL↑ (%)	EPA (%)	DTS↓ (m)
ORG [4]	10.60 0.43	4.04 0.22	17.29 0.58	1.14 0.02	65.50 0.71	43.53 0.88	8.06 0.40	0.62 0.01
SemExp [1]	9.60 0.52	3.62 0.48	15.18 0.47	1.16 0.02	**68.70** 0.47	**48.67** 0.42	7.80 0.40	**0.61** 0.03
GMAN† (ours)	**48.83** 0.60	**25.09** 0.37	10.04 0.09	**0.93** 0.01	57.80 0.78	28.41 0.66	14.12 0.15	0.91 0.03

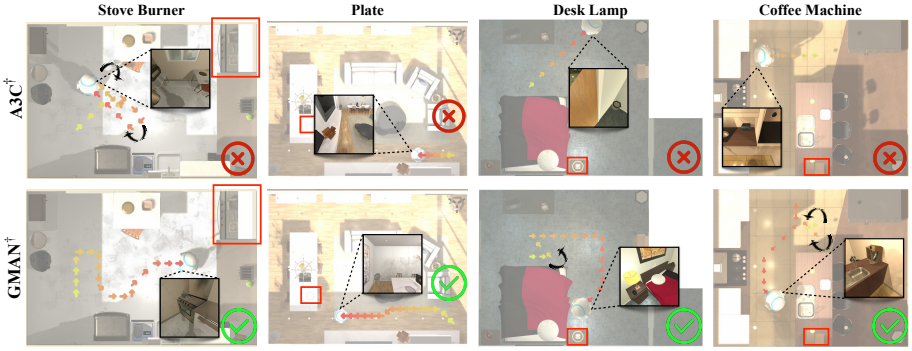

Fig. 4. Visualization of unseen object navigation in AI2THOR. We illustrate the trajectory of the agent, where the black arrows represent rotation, the yellow arrows represent the navigation start and the red arrows represent the navigation end.

various attributes. As a result, it is impossible to annotate the ground truth of attribute detection or segmentation in an image. Thus, it is hard to realize the attribute detection or attribute segmentation. Therefore, we choose to modify these methods by adding the PS module, which contains the information of the semantic similarity between the target object.

Tab. 7 compares our GMAN† with the relation-based and the map-based methods under the A3C† framework (i.e. all models are equipped with the PS module). These related methods benefit from robust object detection or segmentation modules and achieve outstanding performance on seen objects (note that our GMAN† only utilizes egocentric images without detection or segmentation inputs). However, their performances on unseen objects are really poor. We infer that these related methods depend excessively on those visual backbones (e.g. detection or segmentation modules) which offer explicit information about whether the target appears, while ignore the semantic similarity information provided by the PS module. As a result, these methods perform poorly on unseen objects which the detection and segmentation module cannot observe. Conversely, our method generalizes well from seen objects to unseen objects and significantly outperforms these methods in unseen object navigation.

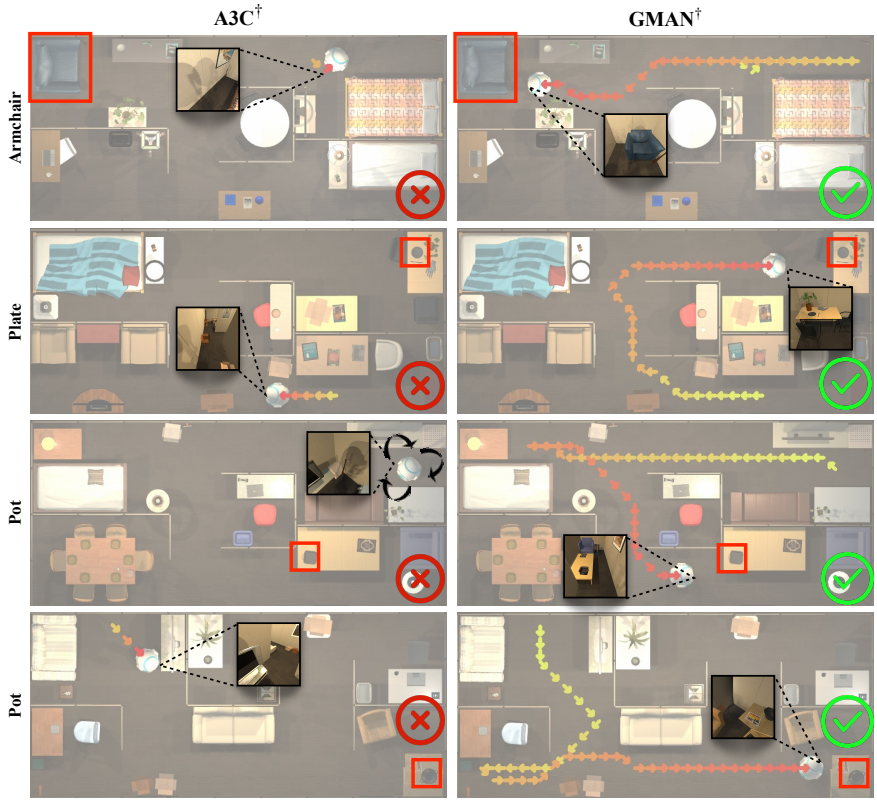

Fig. 5. Visualization of unseen object navigation in RoboTHOR.

5 Case Studies

5.1 Comparisons with the Baseline

As shown in the Fig. 4 and Fig. 5, we visualize the trajectory of the agent for unseen object navigation in AI2THOR and RoboTHOR. We consider two models A3C† and GMAN† for comparisons. Both agents are initially placed at the same position and given the same target object. The baseline A3C† model combines the A3C policy with the PS module. Based on A3C†, our GMAN† is additionally equipped with the feature generator (FG) and the environmental meta-learner (EMD). It can be observed that without generated features of unseen objects, the baseline agent executes straightforward or meaningless actions such as simply moving ahead, frequently spinning around and backing. However, our agent equipped with the FG and the EMD utilizes the generated feature as a more clear goal and performs efficient actions to reach the target object.

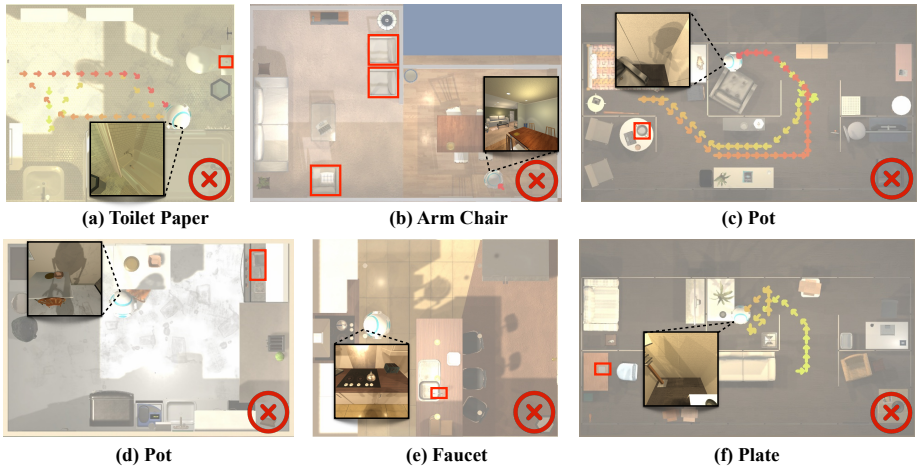

Fig. 6. Visualization of error cases. We illustrate six error cases for unseen object navigation in both AI2THOR and RoboTHOR.

5.2 Error Analysis

As shown in Fig. 6, we also visualize several error cases (for unseen objects) and analyze the failure reasons to discover that the following situations may confuse the agent:

1) Small objects. In the unseen environments, the features of small objects are less obvious, making them difficult to be observed, e.g. the toilet paper in the bathroom (Fig. 6(a)), the faucet in the kitchen (Fig. 6(e)) or the pot in the apartment (Fig. 6(c)).

2) The semantic embeddings of unseen objects are similar to those of seen objects. Our agent is only trained with seen objects and tries to utilize the learned "knowledge" on the unseen objects. Therefore, if an unseen object has some features highly similar to a seen object, the agent may intend to find this "familiar" seen object. In Fig. 6(b), given the unseen target arm chair, the agent wrongly finds the chair due to their similar semantic embeddings.

3) Similar functional objects. Since our method relies on semantic embeddings (e.g. attribute vectors) to adaptively generate the object features, it is challenging for our agent to distinguish those similar functional (e.g. similar attribute vector) objects. For example, the cooking tools usually have similar attributes like pans, bowls, plates and pots, which may distract the agent. Both Fig. 6(d) and Fig. 6(f) show that our agent navigates to the bowl while the targets are respectively the pot and plate.

In the above error cases, the informative semantic embeddings are essential to navigate to the unseen objects. Therefore, in future work, we will annotate more types of attributes to provide more discriminative and informative semantic embeddings for generating the object features.

References

1. Chaplot, D.S., Gandhi, D., Gupta, A., Salakhutdinov, R.R.: Object goal navigation using goal-oriented semantic exploration. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020)
2. Chaplot, D.S., Salakhutdinov, R., Gupta, A., Gupta, S.: Neural topological SLAM for visual navigation. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020. pp. 12872–12881 (2020)
3. Deitke, M., Han, W., Herrasti, A., Kembhavi, A., Kolve, E., Mottaghi, R., Salvador, J., Schwenk, D., VanderBilt, E., Wallingford, M., Weihs, L., Yatskar, M., Farhadi, A.: Robothor: An open simulation-to-real embodied AI platform. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020. pp. 3161–3171 (2020)
4. Du, H., Yu, X., Zheng, L.: Learning object relation graph and tentative policy for visual navigation. In: Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VII. pp. 19–34 (2020)
5. Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.): Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA (2017)
6. Kolve, E., Mottaghi, R., Gordon, D., Zhu, Y., Gupta, A., Farhadi, A.: AI2-THOR: an interactive 3d environment for visual AI. CoRR **abs/1712.05474** (2017)
7. Mayo, B., Hazan, T., Tal, A.: Visual navigation with spatial attention. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021. pp. 16898–16907 (2021)
8. Narayan, S., Gupta, A., Khan, F.S., Snoek, C.G.M., Shao, L.: Latent embedding feedback and discriminative features for zero-shot classification. In: Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII. pp. 479–495 (2020)
9. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing. pp. 722–729. IEEE (2008)
10. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. pp. 1532–1543 (2014)
11. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-ucsd birds 200 (2010)
12. Wortsman, M., Ehsani, K., Rastegari, M., Farhadi, A., Mottaghi, R.: Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 6750–6759 (2019)
13. Xian, Y., Lorenz, T., Schiele, B., Akata, Z.: Feature generating networks for zero-shot learning. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 5542–5551 (2018)
14. Xian, Y., Sharma, S., Schiele, B., Akata, Z.: F-VAEGAN-D2: A feature generating framework for any-shot learning. In: IEEE Conference on Computer Vision and

Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 10275–10284 (2019)
15. Yang, W., Wang, X., Farhadi, A., Gupta, A., Mottaghi, R.: Visual semantic navigation using scene priors. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019 (2019)
16. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017. pp. 3357–3364 (2017)