Layout-based Causal Inference for Object Navigation —Supplements—

Sixian Zhang[1,2], Xinhang Song[1,2], Weijie Li[1,2], Yubing Bai[1,2], Xinyao Yu[1,2], Shuqiang Jiang[1,2]
[1]Key Lab of Intelligent Information Processing Laboratory of the Chinese Academy of Sciences (CAS),
Institute of Computing Technology, Beijing [2]University of Chinese Academy of Sciences, Beijing
{sixian.zhang, xinhang.song, weijie.li, yubing.bai, xinyao.yu}@vipl.ict.ac.cn
sqjiang@ict.ac.cn

1. Calculation Details

In our paper, the object layout is defined as the set $D = \{\theta_i\}_{i=1}^{K}$, where K is the number of object categories. The object layout D is composed of K context distribution of each object category. Each context distribution $\theta_i = (\theta_{i,j})_{j=1}^{K}$ is assumed to follow the Dirichlet distribution $Dir(\alpha_i)$, i.e. $\theta_i \sim Dir(\alpha_i)$. Therefore, the probability density function of θ_i can be calculated as:

$$p(\theta_i|\alpha_i) = Dir(\alpha_i) = \frac{\Gamma\left(\sum_{j=1}^{K}\alpha_{i,j}\right)}{\prod_{j=1}^{K}\Gamma(\alpha_{i,j})}\prod_{j=1}^{K}\theta_{i,j}^{\alpha_{i,j}-1} \quad (1)$$

where $\sum_{j=1}^{K}\theta_{i,j} = 1, \theta_{i,j} \geq 0$, $\alpha_i = (\alpha_{i,j})_{j=1}^{K}$, $\alpha_{i,j} > 0$ is the hyperparameter, and the $\Gamma(s)$ is the gamma function, which is defined as:

$$\Gamma(s) = \int_0^{\infty} x^{s-1}e^{-x}dx, s > 0 \quad (2)$$

To be simplified, we denote that:

$$B(\alpha_i) = \frac{\prod_{j=1}^{K}\Gamma(\alpha_{i,j})}{\Gamma\left(\sum_{j=1}^{K}\alpha_{i,j}\right)} \quad (3)$$

Then the $p(\theta_i|\alpha_i)$ can be re-written as:

$$p(\theta_i|\alpha_i) = \frac{1}{B(\alpha_i)}\prod_{j=1}^{K}\theta_{i,j}^{\alpha_{i,j}-1} \quad (4)$$

The $B(\alpha_i)$ can be regarded as the standardization factor. Additionally, based on the property of probability density functions: $\int p(\theta_i|\alpha_i)\,d\theta_i = 1$, then we have:

$$\int \frac{1}{B(\alpha_i)}\prod_{j=1}^{K}\theta_{i,j}^{\alpha_{i,j}-1}d\theta_i = \frac{1}{B(\alpha_i)}\int\prod_{j=1}^{K}\theta_{i,j}^{\alpha_{i,j}-1}d\theta_i = 1 \quad (5)$$

According to Eq. 5, the $B(\alpha_i)$ can also be calculated as:

$$B(\alpha_i) = \int\prod_{j=1}^{K}\theta_{i,j}^{\alpha_{i,j}-1}d\theta_i \quad (6)$$

Based on the above descriptions, the calculation details about the posterior distribution of θ_i (i.e. Eq. 4 in the main manuscript) are deduced by:

$$\begin{aligned} p(\theta_i|s_i^t,\alpha_i^t) &= \frac{p(s_i^t|\theta_i)\,p(\theta_i|\alpha_i^t)}{p(s_i^t|\alpha_i^t)} \\ &= \frac{p(s_i^t|\theta_i)\,p(\theta_i|\alpha_i^t)}{\int p(s_i^t|\theta_i)\,p(\theta_i|\alpha_i^t)\,d\theta_i} \\ &= \frac{\prod_{j=1}^{K}\theta_{i,j}^{n_{-i,j}^t}\frac{1}{B(\alpha_i^t)}\theta_{i,j}^{\alpha_{i,j}^t-1}}{\int\prod_{j=1}^{K}\theta_{i,j}^{n_{-i,j}^t}\frac{1}{B(\alpha_i^t)}\theta_{i,j}^{\alpha_{i,j}^t-1}d\theta_i} \\ &= \frac{\prod_{j=1}^{K}\theta_{i,j}^{\alpha_{i,j}^t+n_{-i,j}^t-1}}{\int\prod_{j=1}^{K}\theta_{i,j}^{\alpha_{i,j}^t+n_{-i,j}^t-1}d\theta_i} \end{aligned} \quad (7)$$

Based on the Eq. 6, the Eq. 7 can be further deduced to:

$$\begin{aligned} p(\theta_i|s_i^t,\alpha_i^t) &= \frac{1}{B\left(\alpha_i^t+n_{-i,j}^t\right)}\prod_{j=1}^{K}\theta_{i,j}^{\alpha_{i,j}^t+n_{-i,j}^t-1} \\ &= Dir\left(\theta_i|\alpha_i^t+n_{-i}^t\right) \end{aligned} \quad (8)$$

The above derivations detail the calculation of Eq. 4 in the main manuscript.

2. Algorithms

The algorithms about the calculation process of the layout estimator, the training process and the inference process are shown in Alg. 1, Alg. 2 and Alg. 3, respectively.

Algorithm 1 Layout estimator (LE)

Input: The distribution parameter $(\alpha_1^t, \ldots, \alpha_i^t, \ldots, \alpha_K^t)$, where $\alpha_i^t = \left(\alpha_{i,j}^t\right)_{j=1}^K$, the observation s^t, the detection results $n^t = (n_i^t)_{i=1}^K$, and the episode memory Φ

Output: The expectation of the posterior distribution of the object layout w^t, the layout gap D_Δ^t, updated parameter $(\alpha_1^t, \ldots, \alpha_i^t, \ldots, \alpha_K^t)$

for all $i \leftarrow \{1, \ldots, K\}$ **do**
  if $n_i^t > 0$ **and** s^t is not in Φ **then**
    Get context information (s_i^t, n_{-i}^t) of object c_i
    Get prior distribution $p\left(\theta_i|\alpha_i^t\right) = Dir\left(\theta_i|\alpha_i^t\right)$
    Compute the posterior distribution
    $$p(\theta_i|s_i^t, \alpha_i^t) = Dir(\theta_i|\alpha_i^t + n_{-i}^t)$$
    Compute the layout gap of object c_i
    $$D_{\Delta,i}^t = KL\left(p\left(\theta_i|s_i^t, \alpha_i^t\right)||p\left(\theta_i|\alpha_i^t\right)\right)$$
    Update the parameters: $\alpha_i^{t+1} \leftarrow \alpha_i^t + n_{-i}^t$
  else
    $p(\theta_i|s_i^t, \alpha_i^t) = p\left(\theta_i|\alpha_i^t\right) = Dir\left(\theta_i|\alpha_i^t\right)$
    $D_{\Delta,i}^t = 0$
    $\alpha_i^{t+1} \leftarrow \alpha_i^t$
Compute the expectation w^t of $p(\theta_i|s_i^t, \alpha_i^t)$ based on the Eq. 6 in the main manuscript
Compute the object layout $D_\Delta^t = \sum_{i=1}^K D_{\Delta,i}^t$

Algorithm 2 Training

for all training episode $\tau \in \mathcal{T}_{train}$ **do**
  $t \leftarrow 0$, $\Phi \leftarrow \varnothing$
  $(\alpha_1^t, \ldots, \alpha_i^t, \ldots, \alpha_K^t) \leftarrow (\alpha_1^*, \ldots, \alpha_i^*, \ldots, \alpha_K^*)$
  while the termination action is not issued **do**
    Get the observation s^t, the target g^t, and the detection results $n^t = (n_i^t)_{i=1}^K$
    $w^t, D_\Delta^t, (\alpha_i^{t+1})_{i=1}^K \leftarrow \text{LE}(s^t, n^t, (\alpha_i^t)_{i=1}^K, \Phi)$
    Encode w^t into model π by $\hat{z}^t = z^t \odot [w^t \times g^t]$
    Take action from $a^t = \pi(A|S = s^t, G = g^t)$
    $\Phi \leftarrow \Phi \cup \{s^t\}$, $t \leftarrow t + 1$
  Optimize model π by navigation loss $\mathcal{L}_{nav}$

The training and inference stages are detailed in Sec. 4.3 of the main manuscript. The main difference between training and testing is the action prediction (marked in blue). In the training stage, the agent adopts the fact prediction of the navigation model π, while in inference, since the fact prediction could be biased by the effect of the experience Z, our agent will not directly use the trained navigation model. Alternatively, the agent first calculates the layout gap D_Δ^t and the counter-fact prediction $\bar{a}^t$. Then the proposed sTDE removes the counter-fact prediction $\bar{a}^t$ (i.e. effect of the experience) from the fact prediction a^t based on the layout gap D_Δ^t. Therefore, our agent adopts the de-biased prediction sTDE(a^t) for navigation.

Algorithm 3 Inference

Input: The hyperparameter threshold ε

for all testing episode $\tau \in \mathcal{T}_{test}$ **do**
  $t \leftarrow 0$, $\Phi \leftarrow \varnothing$
  $(\alpha_1^t, \ldots, \alpha_i^t, \ldots, \alpha_K^t) \leftarrow (\alpha_1^*, \ldots, \alpha_i^*, \ldots, \alpha_K^*)$
  while the termination action is not issued **do**
    Get the observation s^t, the target g^t, and the detection results $n^t = (n_i^t)_{i=1}^K$
    $w^t, D_\Delta^t, (\alpha_i^{t+1})_{i=1}^K \leftarrow \text{LE}(s^t, n^t, (\alpha_i^t)_{i=1}^K, \Phi)$
    Encode w^t into model π by $\hat{z}^t = z^t \odot [w^t \times g^t]$
    Compute the fact $a^t = \pi(A|S = s^t, G = g^t)$
    Compute the counter-fact (effect of experience) $\bar{a}^t = \pi(A|do\,(S = \bar{s}^t, G = \bar{g}^t), Z = Z_{s^t,g^t})$
    Take action from $sTDE(a^t)$
    $sTDE(a^t) = a^t - ReLU(D_\Delta^t - \varepsilon) \cdot \bar{a}^t$
    $\Phi \leftarrow \Phi \cup \{s^t\}$, $t \leftarrow t + 1$

The calculation of the layout estimator (LE) is shown in Alg. 1. Note that our method also adds a memory Φ to store observations. The memory is designed to prevent repetitive observations (e.g. when the agent gets stuck in a dead-end or spinning in place), which may result in biased estimates of the object layout.

3. Supplements of Experimental Details

3.1. Experimental Setup

Datasets. We utilize the AI2THOR [9], RoboTHOR [4] and the Gibson [14] in the Habitat simulator [12] for evaluation. Specifically, the AI2THOR simulator is widely used in the learning-based navigation methods [6, 7, 10, 13, 15–17], which contains 120 indoor 3D synthetic rooms in four types: kitchen, living room, bedroom and bathroom. For each type, we employ 20 rooms for training, 5 for validation and 5 for testing. Following the previous object goal navigation works [6, 7, 16], we choose the following goals for each type of room in AI2THOR [9]: 1) Kitchen: *Fridge, Light Switch, Pot, Toaster, Coffee Machine, Sink, Pan, Stove Burner, Kettle, Microwave, Garbage Can*. 2) Living Room: *Floor Lamp, Chair, Plate, Light Switch, Garbage Can, Laptop, Remote Control, Book, Television, Desk Lamp*. 3) Bedroom: *Book, Light Switch, Bowl, Desk Lamp, Laptop, Chair, Alarm Clock, Garbage Can*. 4) Bathroom: *Light Switch, Garbage Can, Sink, Cell Phone*.

The environment of RoboTHOR contains multi-rooms, thus, we denote each environment as an apartment. The RoboTHOR consists of 89 apartments, where 75 in train/val (60 for training and 15 for validation), 4 in test-dev (which are used for validation in the real world) and 10 in test-standard (blind physical test set). Since the test-dev and test-standard are not public, we choose the train/val for eval-

Table 1. The comparisons of different training strategies. The π denotes the fact navigation model $a^t = \pi(A|S = s^t, G = g^t)$, while the $\bar{\pi}$ denotes the counter-fact model $\bar{a}^t = \pi(A|do\left(S = \bar{s}^t, G = \bar{g}^t\right), Z = Z_{s^t,g^t})$, whose prediction is only affected by the experience Z.

ID	Training	$do\,(S,G)$	Navigating with a^t			Navigating with $\bar{a}^t$ (experience)			Navigating with $sTDE\left(a^t\right)$		
			SR↑ (%)	SPL↑ (%)	DTS↓ (m)	SR↑ (%)	SPL↑ (%)	DTS↓ (m)	SR↑ (%)	SPL↑ (%)	DTS↓ (m)
I	Only train π	zero	$71.10_{\pm 0.26}$	$39.13_{\pm 0.19}$	$0.47_{\pm 0.01}$	$25.46_{\pm 0.16}$	$13.08_{\pm 0.19}$	$1.15_{\pm 0.01}$	$72.03_{\pm 0.29}$	$39.97_{\pm 0.16}$	$0.48_{\pm 0.02}$
		random				$32.08_{\pm 0.16}$	$16.66_{\pm 0.13}$	$0.97_{\pm 0.01}$	$72.13_{\pm 0.36}$	$39.92_{\pm 0.15}$	$0.47_{\pm 0.02}$
II	Separately train π and $\bar{\pi}$	zero	$71.10_{\pm 0.26}$	$39.13_{\pm 0.19}$	$0.47_{\pm 0.01}$	$67.73_{\pm 0.11}$	$38.64_{\pm 0.13}$	$0.56_{\pm 0.01}$	$75.15_{\pm 0.14}$	$41.64_{\pm 0.21}$	$0.45_{\pm 0.02}$
		random				$68.43_{\pm 0.22}$	$38.75_{\pm 0.11}$	$0.53_{\pm 0.01}$	$75.25_{\pm 0.11}$	$41.69_{\pm 0.19}$	$0.46_{\pm 0.02}$
III	Alternatively train π and $\bar{\pi}$	zero	$71.40_{\pm 0.37}$	$39.11_{\pm 0.44}$	$0.49_{\pm 0.02}$	$68.43_{\pm 0.27}$	$36.42_{\pm 0.17}$	$0.55_{\pm 0.02}$	$74.95_{\pm 0.38}$	$41.68_{\pm 0.23}$	$0.47_{\pm 0.02}$
		random				$69.76_{\pm 0.33}$	$35.31_{\pm 0.19}$	$0.52_{\pm 0.02}$	$75.03_{\pm 0.69}$	$41.48_{\pm 0.44}$	$0.46_{\pm 0.03}$

uation, where 60 for training, 5 for validation and 10 for testing. Different from AI2THOR, the set of the target objects is basically consistent among different apartments in RoboTHOR. Therefore, we consider 12 object categories as the target, involving: *Book, Bowl, Chair, Plate, Television, Floor Lamp, Garbage Can, Alarm Clock, Desk Lamp, Laptop, Pot, CellPhone*.

The Gibson dataset consists of full buildings, which also contain multi-rooms. The Gibson is widely used for the evaluations of the map-based methods [3, 11]. Our experimental settings on Gibson are the same as [3]. We adopt 25 train/ 5 test scenes from the Gibson tiny split whose semantic annotations are available from [2], and consider 6 goal categories: *Chair, Couch, Potted Plant, Bed, Toilet, Television*.

Evaluation metrics. We evaluate ObjectNav performance with SR, SPL and DTS metrics.

SR (Success Rate): SR evaluates the success rate of the agent in finding the target object, which is formulated 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 an indicator representing whether the i-th episode is successful.

SPL (Success weighted by Path Length): SPL further considers both the success rate and the path length. It is defined as $SPL = \frac{1}{N}\sum_{i=1}^{N}\mathcal{S}_i\frac{l_i^*}{max\left(l_i,l_i^*\right)}$, where l_i^* refers to the shortest path calculated by the simulator and l_i is the path length in the i-th episode.

DTS (Distance to Goal): DTS measures the distance $L_{i,g}$ of the agent towards the goal at the end of the episode. It is defined as $DTS = \frac{1}{N}\sum_{i=1}^{N} max\left(L_{i,g} - \xi, 0\right)$, where $\xi = 1m$ is the success threshold (the successful episode is defined in the Sec. 3 of the main manuscript). For a successful episode, the DTS is 0 [1].

3.2. More Evaluations

Comparisons of different training strategies. Initially, we train the model π during the training stage (see the setting I in Tab. 1) as most previous methods do [6, 7, 15, 16]. In the inference, we construct the counterfactual prediction $\bar{a}^t$ (the effect of experience) by specifying a certain value for the S, G and Z, then remove the biased prediction $\bar{a}^t$ from the original prediction (the fact a^t) based on the proposed sTDE. However, the results indicate that the improvements of sTDE are limited with only 1.03%, 0.79% gains on SR and SPL metrics. For comparison, as shown in the setting II in Tab. 1, we alternatively construct the counterfactual prediction by training the counterfactual model $\pi(A|do\left(S = \bar{s}^t, G = \bar{g}^t\right), Z = Z_{s^t,g^t})$ with the same navigation loss as the fact model $\pi(A|S = s^t, G = g^t)$. Based on the trained counterfactual model $\bar{\pi}$, the sTDE achieves significant improvements by 4.15%, 2.56% and -0.01m on SR, SPL and DTS.

To investigate the reason for such difference, we conduct the ablation studies (see the column 'Navigating with $\bar{a}^t$' in Tab. 1), and the results demonstrate that the counterfactual model (setting I) obtained by directly assigning the value has worse performance that the trained model (setting II). We analyze that the original model π is trained to predict actions based on the information from both experience Z and other inputs (i.e. S and G), while it is not well-adapted to predict actions relying on information from only one side. Therefore, the unseen pattern (directly assigning values in the original model) is unfamiliar to the trained model, thus the counterfactual model obtained by assigning values cannot precisely reflect the situation that the agent navigates to the target only with experience.

The above experimental results seem to suggest that the counterfactual model needs to be additionally trained, separate from the original training process. However, such a setting would cause additional computational costs. Motivated by the dropout operation, we randomly mask the value of S and G as if the π and $\bar{\pi}$ are alternatively trained (see the setting III in Tab. 1). Note that the trained π and $\bar{\pi}$ have different parameters in setting II, while they share same parameters in setting III. This modified setting (setting III) has the same computational costs as the original

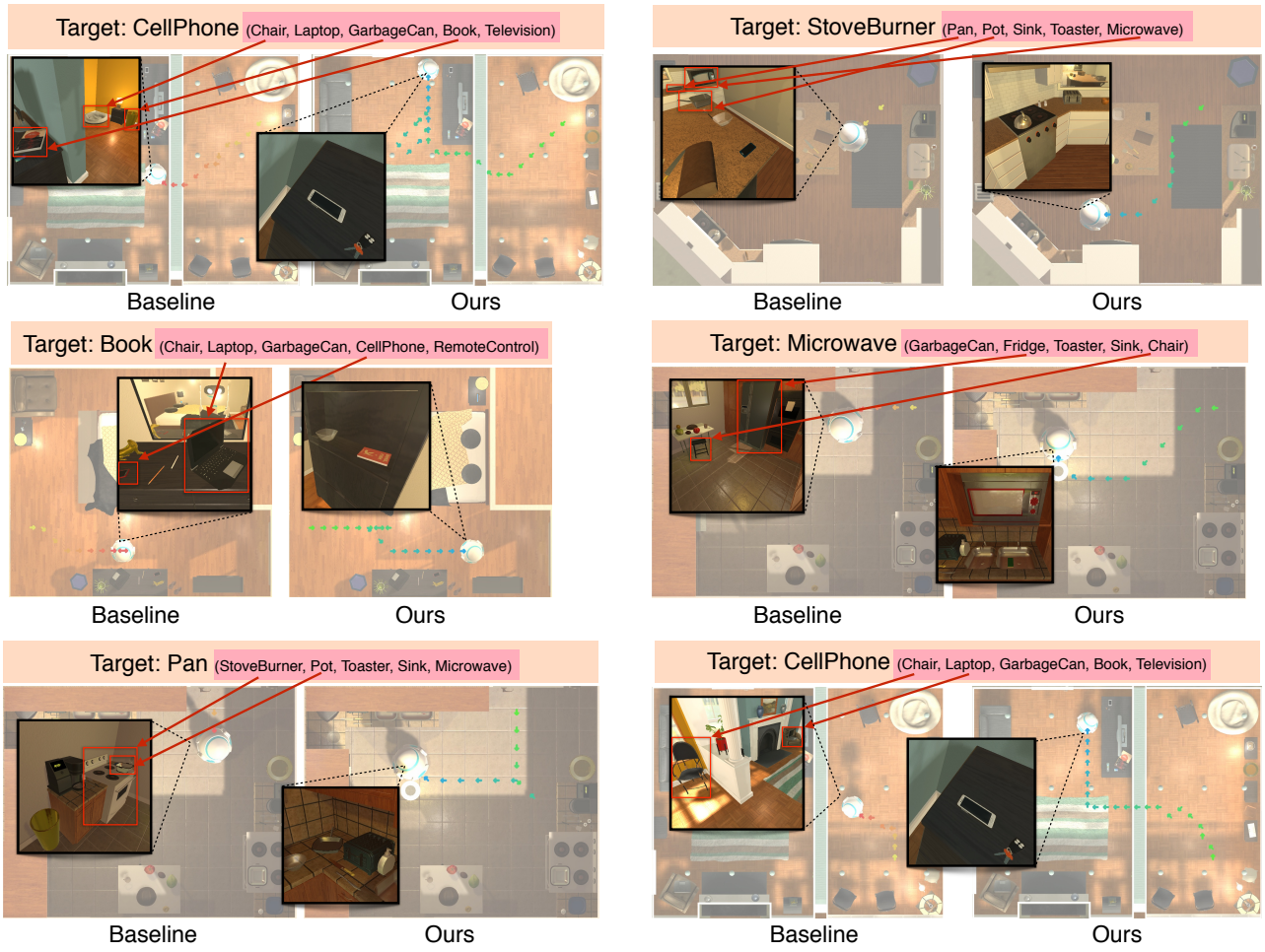

Figure 1. Visualization of trajectories in AI2THOR [9]. The trajectories of successful episodes are illustrated with green and blue colors, where green refers to the beginning, and blue represents the end. In contrast, the trajectories of failed episodes are shown with yellow and red colors, where yellow denotes the beginning, and red represents the end. The target object and the top 5 objects with the highest co-occurrence frequency (arranged from high to low) with the target object in the training environments are also annotated in the figure.

setting (setting I), while achieving similar performances as the additional training setting (setting II) in both navigating with $\bar{a}^t$ and navigating with sTDE. Therefore, we adopt the training strategy (setting III) that randomly masks the S and G to train our model.

Additionally, the results indicate that under our training strategy, the counterfactual prediction $\bar{a}^t$ obtained by the intervention and counterfactual operations has practical meanings, which represent the situation of navigating only with experience, and achieves reasonable performance (comparing setting III and setting II in both 'Navigating with $\bar{a}^t$' and 'Navigating with $sTDE\,(a^t)$').

Comparison with other baselines. As shown in Tab. 2 (I, II), we conducted more comparisons of adding our sTDE into EmbCLIP [8] in two settings: 1) only training with RoboTHOR (60 scenes) and 2) pre-training on large-scale datasets ProcTHOR [5] (10K scenes). The results show that under limited data (case I), adding our sTDE also gains notable improvement, while, when large training data is avail-

Table 2. More comparisons with other baselines: (I) baseline with CLIP backbone. (II) baseline with pre-training in ProcTHOR.

ID		Method	SR(%)	SPL(%)
		Results in RobTHOR (val)		
I	1	EmbCLIP	52.22	25.99
	2	EmbCLIP + sTDE	57.37(5.15↑)	26.81(0.82↑)
		Results in RobTHOR (val) with pre-trained in ProcTHOR		
II	3	EmbCLIP (pre-train w/o fine-tune)	51.33	22.20
	4	EmbCLIP (pre-train w/ fine-tune)	66.39	27.44
	5	EmbCLIP (pre-train w/ fine-tune)+sTDE	67.73(1.34↑)	27.77(0.33↑)

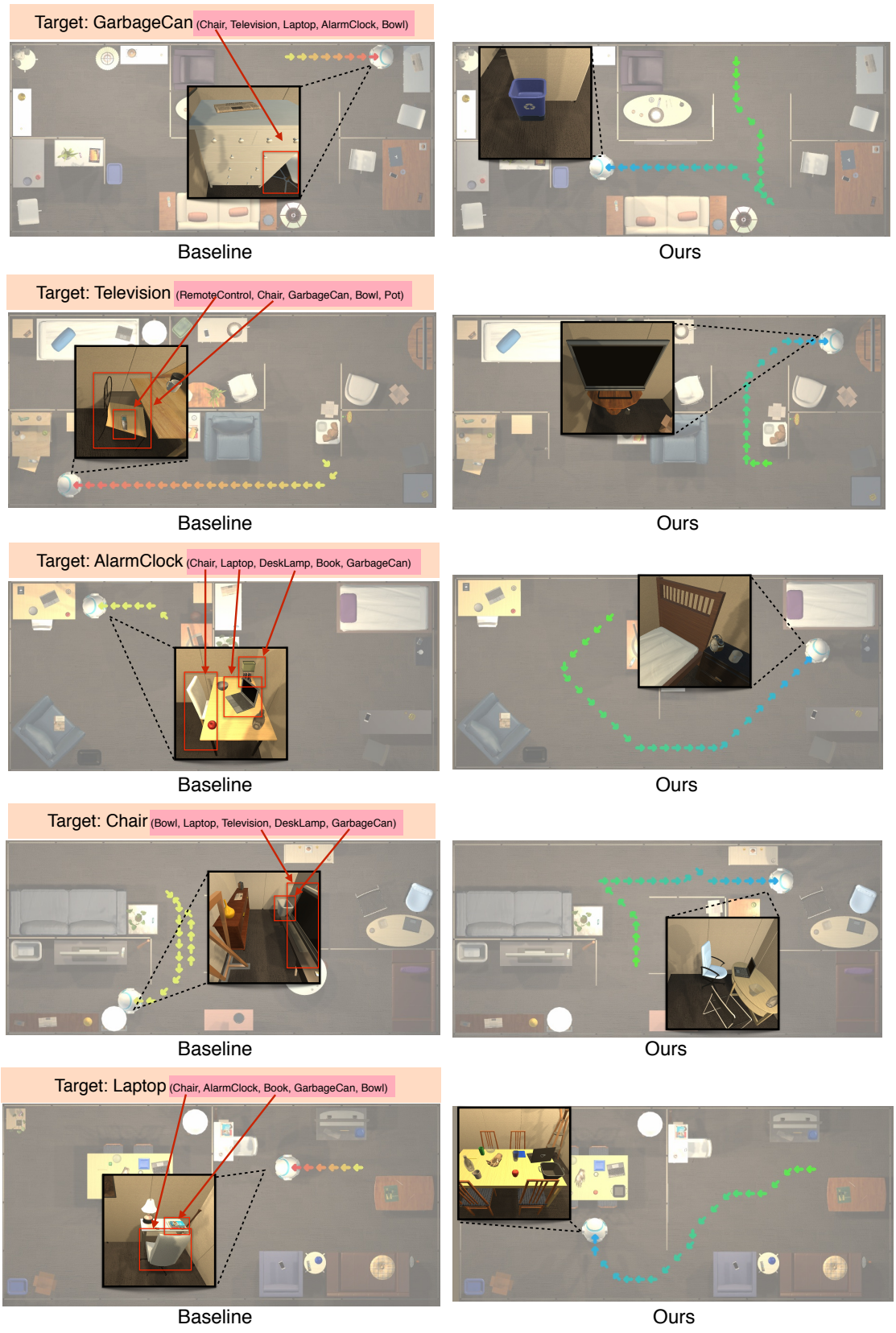

Figure 2. Visualization of trajectories in RoboTHOR [4]. The annotations are consistent with that of Fig. 1.

able (case II), our method is still effective but the gains are limited. We analyze that large-scale datasets provide more diverse layouts, which make the learned experience more general, and thus obtain less benefit from our de-biased method.

Our sTDE attempt to improve the model generalization from the algorithm, which is less concise and efficient than directly scaling up the training data e.g. ProcTHOR. However, compared to the huge computation cost for pre-training, the advantages of our sTDE are that it is parameter-free and easy to plug in many methods, even including those pre-trained models. Besides, our sTDE is orthogonal to the data-based method and can further improve the performance of data-based models.

Case study. We further visualize the trajectories of the agent in both AI2THOR [9] and RoboTHOR [4] simulators. The baseline model is chosen as the ORG [6]. Besides, we annotate the top 5 object categories with the highest frequency of co-occurrence with the target object in the training environments, and these objects are arranged from high to low based on their co-occurrence frequency. These co-occurrence objects reflect the context layout of the target object in the training environments. As shown in Fig. 1 and 2, the co-occurring objects are often observed at the location where the agent stops navigating in the failed episodes (see the red arrows). The visualization results indicate that the baseline agent is prone to get stuck in the locations where the target object usually appears in the training environments. Furthermore, the results demonstrate that the agent is easily influenced by the experience learned in training, which results in the failure of navigation. However, our method appropriately eliminates the effect of experience ($Z \rightarrow A$) and encourages the exploration-based effect ($S \rightarrow A$ and $G \rightarrow A$) as shown in the causal graph in the main manuscript. Therefore, the agent, driven by our method, utilizes the learned experience more wisely, and tends to be more exploratory rather than suffering from the biased experience. Consequently, our agent achieves better performance and navigates with more efficient trajectories.

4. Limitation and Societal Impacts

4.1. Limitation Analysis

We will discuss the limitation of our method from the following two aspects:

(1) Based on the proposed causal graph, we argue that the bias of action prediction comes from the negative effect of experience. The proposed causal graph focuses on emphasizing the effects of experience while omitting other factors (e.g. hidden state h_{t-1} of LSTM), which may also affect the action prediction. Therefore, in future work, we will introduce more factors into our causal graph and analyze their potential influences.

(2) The proposed object layout estimator calculates the posterior context distribution based on the continuous observations. To collect high-quality observations, the agent needs to navigate efficiently without getting stuck or backtracking. In this paper, we construct a memory (see the Φ in the Alg.1) to avoid the influence of invalid observations. However, such memory can only eliminate invalid observations, but cannot increase valid observations. Therefore, in future work, we will utilize additional measures (e.g. adding extra exploration rewards) to encourage the agent to obtain more valid observations.

4.2. Potential Negative Societal Impacts

Our method is a general method for improving the navigation abilities of object-oriented navigation (ObjectNav) task. The prevailing methods place the agent in 3D virtual environments and train the agent with reinforcement learning. Some virtual environments (e.g. AI2THOR) are constructed manually with the game engines (e.g. Unity), while other virtual environments (e.g. Matterport3D) are 3D reconstructions of real-world environments, which have the potential negative impacts of exposing the privacy on room layout. Additionally, the technologies of ObjectNav task are gradually intended to be applied to some service robots which can make human life more convenient and comfortable. However, the activities of these robots may also produce some hazards, which will cause damage to persons or societal property. Therefore, further researches need to ensure the positive impact of this ObjectNav technology on the society.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3674–3683. IEEE Computer Society, 2018. 3

[2] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5664–5673, 2019. 3

[3] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Russ R. Salakhutdinov. 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. 3

[4] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. 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*, pages 3161–3171, 2020. 2, 5, 6

[5] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied AI using procedural generation. 2022. 4

[6] Heming Du, Xin Yu, and Liang Zheng. 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*, pages 19–34, 2020. 2, 3, 6

[7] Heming Du, Xin Yu, and Liang Zheng. Vtnet: Visual transformer network for object goal navigation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. 2, 3

[8] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: CLIP embeddings for embodied AI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 14809–14818. IEEE, 2022. 4

[9] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, abs/1712.05474, 2017. 2, 4, 6

[10] Bar Mayo, Tamir Hazan, and Ayellet Tal. Visual navigation with spatial attention. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 16898–16907, 2021. 2

[11] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. PONI: potential functions for objectgoal navigation with interaction-free learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 18868–18878. IEEE, 2022. 3

[12] Manolis Savva, Jitendra Malik, Devi Parikh, Dhruv Batra, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, and Vladlen Koltun. Habitat: A platform for embodied AI research. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9338–9346. IEEE, 2019. 2

[13] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 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*, pages 6750–6759, 2019. 2

[14] Fei Xia, Amir Roshan Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9068–9079. IEEE Computer Society, 2018. 2

[15] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. 2, 3

[16] Sixian Zhang, Xinhang Song, Yubing Bai, Weijie Li, Yakui Chu, and Shuqiang Jiang. Hierarchical object-to-zone graph for object navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15130–15140, October 2021. 2, 3

[17] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 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*, pages 3357–3364, 2017. 2