Unbiased Directed Object Attention Graph for Object Navigation

Ronghao Dang
dangronghao@tongji.edu.cn
Tongji University

Zhuofan Shi
zhuofanshi@student.ethz.ch
ETH Zürich

Liuyi Wang
wly@tongji.edu.cn
Tongji University

Zongtao He
xingchen327@tongji.edu.cn
Tongji University

Chengju Liu
liuchengju@tongji.edu.cn
Tongji University

Qijun Chen
qjchen@tongji.edu.cn
Tongji University

ABSTRACT

Object navigation tasks require agents to locate specific objects in unknown environments based on visual information. Previously, graph convolutions were used to implicitly explore the relationships between objects. However, due to differences in visibility among objects, it is easy to generate biases in object attention. Thus, in this paper, we propose a directed object attention (DOA) graph to guide the agent in explicitly learning the attention relationships between objects, thereby reducing the object attention bias. In particular, we use the DOA graph to perform unbiased adaptive object attention (UAOA) on the object features and unbiased adaptive image attention (UAIA) on the raw images, respectively. To distinguish features in different branches, a concise adaptive branch energy distribution (ABED) method is proposed. We assess our methods on the AI2-Thor dataset. Compared with the state-of-the-art (SOTA) method, our method reports 7.4%, 8.1% and 17.6% increase in success rate (SR), success weighted by path length (SPL) and success weighted by action efficiency (SAE), respectively.

CCS CONCEPTS
- Computing methodologies → Vision for robotics.

KEYWORDS
Object Navigation, Object Attention Bias, Object Attention Graph

ACM Reference Format:
Ronghao Dang, Zhuofan Shi, Liuyi Wang, Zongtao He, Chengju Liu, and Qijun Chen. 2022. Unbiased Directed Object Attention Graph for Object Navigation. In Proceedings of the 30th ACM International Conference on Multimedia (MM ’22), October 10–14, 2022, Lisboa, Portugal. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3503161.3547852

1 INTRODUCTION

The object navigation [15, 23, 24, 34] requires an agent to navigate in unseen environments and find a specified target by executing a sequence of actions. The agent can only use visual observation and target information as inputs and predict actions by deep reinforcement learning in each step.

Most prior works [19, 21, 22] have directly used global image features to recursively train agents based on egocentric observations. Nevertheless, if the target is invisible, it is difficult to efficiently navigate to the object position with these methods. Therefore, some recent works [6, 33] have focused on establishing specific object prior knowledge to better understand complete scenes. Yang et al. [33] and Du et al. [6] used the graph convolutional networks (GCNs) to learn graph representations of object features.

However, the agent may not treat all items equally, preferring to focus on more conspicuous objects. Therefore, we propose the object attention bias problem, which refers to the situations in which agents focus on objects with high visibility during navigation. For example, when looking for a cell phone, an agent may focus on a closer, clearer TV while ignoring a distant, blurrier laptop that has a higher correlation with the cell phone (Figure 1). In principle, this phenomenon occurs mainly because neural networks prefer features with higher information entropy. Therefore, the direct use of graph convolutions to implicitly learn relationships between objects can lead to considerable object attention bias.

To address the above problem, object attention should be decoupled from the object features, so we let the agent explicitly learn the attention weights of objects according to the different targets. Concretely, we propose a learnable two-layer directed object attention (DOA) graph that represents the relationships between objects in view and the target to address the object attention bias problem. The first graph layer is an intrinsic attention graph (IG) that establishes the basic object attention relationships. The second graph layer is a view adaptive attention graph (VAG) that changes based on the observations of the agent. The DOA graph, which is produced by the weighted average of the two graph layers, can guide an agent to reasonably assign attention to each object in view.

Based on the DOA graph, we further design two novel cross-attention modules: the unbiased adaptive image attention (UAIA) module and the unbiased adaptive object attention (UAOA) module. As illustrated in Figure 1, the target is the arrival point, and the objects in view are the starting points for the directed edges in the DOA graph. The weight of a directed edge from an object node to a target node is the object’s attention while searching for this target. The UAOA module uses object attention in the DOA graph to directly distribute weights to different objects. The UAIA module uses a multihead attention mechanism to determine the areas in the global image that need attention. We follow the operation described in [6] to concatenate the image branch, object branch and previous action branch into a vector. However, just as the temporal sequence
Figure 1: In the previous biased method [6], since the nearby TV is clearer than the distant laptop, the agent focuses more on the TV, resulting in an incorrect decision. Our proposed unbiased DOA graph learns that the cell phone is more likely to be near the laptop according to the target and the current view, resulting in a correct decision.

in the transformer needs positional encodings to acquire position information [28, 31, 37], different branches need tokens to represent their identities. In consequence, we propose an adaptive branch energy distribution (ABED) method, which allows the network to distinguish different branches with the addition of few parameters. Then, in accordance with [36], we input the concatenated features of the multimodel information into a long short-term memory (LSTM) network for learning and optimize the model with the A3C reinforcement learning strategy.

Extensive experiments on the AI2-Thor [13] dataset show that our method not only eliminates the object attention bias problem but also increases the state-of-the-art (SOTA) method by 7.4%, 8.1%, 17.6% in the success rate (SR), success weighted by path length (SPL) and success weighted by action efficiency (SAE) [36]. Our method performs well as much as the agent has a more comprehensive understanding of object relationships and an unbiased attention distribution. In summary, our contributions are as follows:

- We identify the prevalent object attention bias problem in object navigation, which occurs due to differences in object visibility.
- We propose the directed object attention (DOA) graph, which addresses the problem of object attention bias and provides the agent with a better understanding of the internal relationships between objects.
- The unbiased adaptive object attention (UAOA) and unbiased adaptive image attention (UAIA) modules use the DOA graph to allocate more reasonable attention resources to object features and global image features.
- We propose the parameter-free adaptive branch energy distribution (ABED) method to optimize branch feature aggregation.

2 RELATED WORKS
2.1 Object Navigation
In an object navigation task, an agent is given goal-oriented instruction to search for a specified object. The primitive object navigation models make decisions by directly processing input images with convolutional neural networks (CNNs). Recently, researchers [3, 8, 33] have found that using only CNN features in raw images cannot achieve the desired results. An increasing number of researchers have begun to use methods such as object detection to extract high-level semantic features to better guide the agent’s movement. Yang et al. [33] initially use graph convolutional networks (GCNs) to learn the object prior knowledge. Gao et al. [8] utilize cross-modality knowledge reasoning (CKR) to apply an external knowledge graph in the agent’s navigation. Zhang et al. [36] propose the hierarchical object-to-zone (HOZ) graph to guide an agent in a coarse-to-fine manner. In our work, we conduct the online-learning directed object attention (DOA) graph to serve as prior knowledge, which provides more unbiased object attention.
We cannot criticize biased training because our visual world is inherently biased; people simply prefer to trust objects that are easy to identify. For example, when looking for a plate, we often use the frequency each object is present in the dataset. This paper mainly starts from the endogenous cause without changing the number of objects in the dataset. Our proposed DOA graph corrects the agent’s neglect of small and ambiguous objects (bad bias) while maintaining the agent’s trust in high-confidence objects (good bias).

4 PROPOSED METHOD

Our goal is to propose an attention allocation strategy without object attention bias and a reasonable cross-branch aggregation method for a target-driven visual navigation system. To achieve this goal, our navigation system contains four major components, as illustrated in Figure 3: (i) directed object attention (DOA) graph generator; (ii) unbiased adaptive object attention (UAOA) module; (iii) unbiased adaptive image attention (UAIA) module; (iv) adaptive branch energy distribution (ABED) method. (ii) and (iii) are based on the object attention in the DOA graph.

4.1 Task Definition

Initially, the agent is given a random target from a group of $N$ objects $P = \{\text{Pan}, \cdots, \text{Cellphone}\}$, and starts from a random state $s = (x, y, \theta_r, \theta_h)$ in a random house. Here, $(x, y)$ and $(\theta_r, \theta_h)$ represent the coordinates and angles of the agent. After the state and target are initialized, the agent begins to navigate based on its own observations. At each timestamp $t$, the agent only receives the RGB image $o_t$ from a single perspective and the target $p \in P$. According to $o_t$ and $p$, the agent learns a navigation strategy $\pi(a_t|o_t, p)$, where $a_t \in A = \{\text{MoveAhead}; \text{RotateLeft}; \text{RotateRight}; \text{LookDown}; \text{LookUp}; \text{Done}\}$ and $\text{Done}$ is the output if the agent believes it has navigated to the target location. The successful episode is defined as: an agent selects the termination action $\text{Done}$ when the distance between the agent and the target is less than 1.5 meters and the target is in the field of view.

4.2 Directed Object Attention (DOA) Graph

DOA graph is a graphical representation of the correlation degree between identifiable objects and the target. According to the analysis presented in Section 3.1, previous GCN-based methods cannot learn unbiased relationships between objects in object navigation tasks. In contrast, our proposed DOA graph provides an explicit yet flexible representation of the relationships between objects. The DOA graph is obtained through a weighted summation of the intrinsic object graph and the view adaptive graph.

4.2.1 Intrinsic Object Graph. The intrinsic object graph represents the ubiquitous intrinsic relationships between objects. For example, a mouse and a laptop have a strong inherent relationship. Here, we define a learnable matrix $G_n \in \mathbb{R}^{N \times N}$ to represent the intrinsic object graph, where $N$ is the number of objects. As the agent uses reinforcement learning to continuously explore different environments, $G_n$ gradually tends to be reasonable and stable. $G_n$ is fixed during testing. Each edge between objects in $G_n$ is bidirectional. The end node of an edge represents the target object $p$, while the start node of the edge represents the object that needs to be assigned attention. Therefore, the weight of each directed edge represents the intrinsic correlation between an object $q \in P$ and the target object $p \in P$. Each row of $G_n$ is normalized using SoftMax to ensure that the sum of all edge values connected to a target node is 1.
4.2.2 View Adaptive Graph. The view adaptive graph represents the real-time adaptive relationships between objects. The agent generates the global image features \( I_t \in \mathbb{R}^{512 \times 512} \) (from ResNet18 [10]) and object features \( S_t \in \mathbb{R}^{N \times 518} \) (from Faster-RCNN [27]) after observing the current scene. Here, \( M \) is the pixel number of the image feature map. \( S_t \) is concatenated by the object visual features \( S_i^{\text{visual}} \), object position features \( S_i^{\text{pos}} \), confidence \( S_i^{\text{conf}} \) and target indicator \( S_t^{\text{target}} \). Since low-confidence object detection results often contain excessive noise, we filter \( S_t \) with the confidence criterion \( S_i^{\text{conf}} > \text{threshold} \) to obtain \( S_t \).

For introducing target information to the image features \( I_t \), we encode the object index using the one-hot method and two fully connected layers to obtain \( OI \). The input image query \( \tilde{I}_t \in \mathbb{R}^{1 \times 512} \) can be formulated as:

\[
\tilde{I}_t = \text{Concat}(\sum_{i=1}^{M} I_{i}, OI)
\]  \hspace{1cm} (1)

where \( OI \) refers to the \( p \)-th object (target) semantics. \( \frac{1}{M} \sum_{i=1}^{M} I_{i} \) is squeezing global spatial information into a channel descriptor using global average pooling. The agent grounds the current overall environmental characteristics \( \tilde{I}_t \) to the object features \( S_t \) via multi-head score calculation [31] to produce the view adaptive graph \( G_o \in \mathbb{R}^{N \times 1} \):

\[
\tilde{Q}_i = \tilde{I}_t \tilde{W}_i^Q, \quad \tilde{K}_i = \tilde{S}_t \tilde{W}_i^K \quad i = 1, 2, \cdots, NH
\]  \hspace{1cm} (2)

\[
\text{head}_i = \text{softmax}(\tilde{Q}_i \tilde{K}_i^T / \sqrt{\text{HD}})
\]  \hspace{1cm} (3)

\[
G_o = \text{Concat} (\text{head}_1, \cdots, \text{head}_{NH}) \tilde{W}_i^O
\]  \hspace{1cm} (4)

where \( HD \) and \( NH \) denote the hidden dimensionality and number of heads in the multi-head score calculation. \( \tilde{W}_i^Q \in \mathbb{R}^{512 \times HD} \) and \( \tilde{W}_i^K \in \mathbb{R}^{518 \times HD} \) map \( \tilde{I}_t \) and \( \tilde{S}_t \) to the same dimension \( HD \). \( \tilde{W}_i^O \in \mathbb{R}^{NH \times 1} \) aggregate the various subgraphs calculated by the scaled dot-product of the multiple heads to generate the graph \( G_o \).

4.2.3 Object Attention. According to the searched target \( p \), we take the edge weight \( G^p_n \in \mathbb{R}^{N \times 1} \) from the intrinsic object graph \( G_n \) with the \( p \)-th node as the end point. With the weighted summation of the intrinsic weight and the view adaptive weight, we can obtain the attention:

\[
G_t = G^p_n w_n + G_o w_o
\]  \hspace{1cm} (5)

that each object requires. The learnable \( w_n \) and \( w_o \) are the weights of the two graphs.

4.3 Unbiased Adaptive Attention

4.3.1 Unbiased Adaptive Object Attention (UAOA). The purpose of unbiased adaptive object attention (UAOA) is to use the object attention \( G_t \) obtained in Section 4.2 to assign attention to different object features. To balance the information integrity and computational complexity, we apply two fully connected layers around the ReLU [25] to map the object features \( S_t \) to a lower-dimensional representation \( S_t' \in \mathbb{R}^{N \times 64} \). Finally, the attention weight of each object \( q \) is multiplied by the object features \( S_t'q \):

\[
\tilde{S}_t^q = \text{scale}(S_t', G_t^q) = S_t'^q G_t^q \quad q = 1, 2, \cdots, N
\]  \hspace{1cm} (6)

where \( S_t'q = \{S_t'^1, S_t'^2, \cdots, S_t'^N\} \), \( S_t'^q \) is the low dimensional feature of the \( q \)-th object. \( G_t^q \) is the weight of the \( q \)-th object in DOA graph at time \( t \).
Table 1: We obtain a strong and concise baseline by comparing the roles of different modules in previous methods [6, 26, 36] (%). These modules include the object GCN (GCN), the raw image branch (Image), the zone branch (Zone), the room branch (Room), the previous action branch (Action) and the object branch (Object).

| GCN | Image | Zone | Room | Action | Object | All SR | SPL | SAE | L=5 | Episode Time |
|-----|-------|------|------|--------|--------|-------|-----|-----|-----|--------------|
| ✓   | ✓     | ✓    | ✓    | ✓      | ✓      | 68.53±1.32 | 37.50±1.01 | 25.98±0.96 | 60.27±1.67 | 27.68±1.31 | 0.239 |
| ✓   | ✓     | ✓    | ✓    | ✓      | ✓      | 68.14±1.61 | 37.80±0.87 | 26.07±1.25 | 58.15±1.19 | 27.55±0.88 | 0.177 |
| ✓   | ✓     | ✓    | ✓    | ✓      | ✓      | 63.95±1.37 | 32.04±1.69 | 26.72±0.87 | 54.38±0.56 | 26.52±1.21 | 0.145 |
| ✓   | ✓     | ✓    | ✓    | ✓      | ✓      | 67.51±0.97 | 37.81±1.13 | 25.75±1.04 | 58.31±1.22 | 27.83±1.36 | 0.223 |
| ✓   | ✓     | ✓    | ✓    | ✓      | ✓      | 67.61±1.16 | 37.60±1.32 | 26.11±1.88 | 59.21±1.45 | 28.35±1.17 | 0.235 |
| ✓   | ✓     | ✓    | ✓    | ✓      | ✓      | 64.46±1.74 | 34.60±1.14 | 25.49±1.09 | 54.68±0.67 | 27.81±1.12 | 0.243 |
| ✓   | ✓     | ✓    | ✓    | ✓      | ✓      | 48.26±1.11 | 19.88±1.31 | 17.58±1.57 | 35.80±1.12 | 17.23±1.65 | 0.160 |
| ✓   | ✓     | ✓    | ✓    | ✓      | ✓      | 69.14±0.67 | 37.87±0.88 | 27.77±0.95 | 60.42±1.11 | 27.28±0.66 | 0.174 |

4.3.2 Unbiased Adaptive Image Attention (UAIA). We use the DOA graph to focus more attention on the areas around important objects in the image. We use the encoded object index OI rather than word embeddings trained by other networks to represent the object semantic information (what the object is). The object index embeddings learned by our network are more coupled to our dataset than word embeddings trained on other tasks. We use the object attention $G_t$ to generate the attention-aware object semantics $D$:

$$ D = \sum_{q=1}^{N} I(S^{conf}_t > \text{threshold}) G_t^q \times OI $$  \hspace{1cm} (7)

where $I(\cdot)$ is the indicator function. The confidence filter $S^{conf}_t > \text{threshold}$ indicates that only the semantics of objects whose detection confidence exceeds threshold are used. Unlike the convolution operation in CNNs, the multi-head attention operation is permutation-invariant [5], which cannot leverage the order of the tokens in an input sequence. To mitigate this issue, our work adds absolute positional encodings to each pixel in the input sequence, such as in [5, 31], enabling order awareness. We use standard learned 1D pixel index embedding $PI \in \mathbb{R}^{M \times 64}$ since we have not observed any significant performance gains when using more complicated 2D position embeddings. The resulting sequence of embedded vectors serves as input to the encoder.

$$ I_t' = \delta(\delta(I_t W_t) W_2) + PI $$ \hspace{1cm} (8)

where $\delta$ refers to the ReLU function, $W_t \in \mathbb{R}^{512 \times 128}$ and $W_2 \in \mathbb{R}^{128 \times 64}$ reduce the dimension of the global image features $I_t$ and the pixel index embedding $PI$ is introduced to generate position-aware image features $I_t' \in \mathbb{R}^{M \times 64}$. We use the attention-aware object semantics $D$ as the query and the position-aware image features $I_t'$ as the key and value in the multi-head image attention $(HD = 64, NH = 4)$ to generate the final image embedding $\hat{I}_t$.

$$ Q_i = DW_t^Q \hspace{1cm} K_i = I_t' W_t^K \hspace{1cm} V_i = I_t' W_t^V \hspace{1cm} i = 1, 2, \ldots, NH $$ \hspace{1cm} (9)

$$ head_i = \text{softmax} \left( \frac{Q_i K_i^T}{\sqrt{HD}} \right) W_i $$ \hspace{1cm} (10)

$$ \hat{I}_t = \text{Concat}(head_1, \ldots, head_{NH}) W_O $$ \hspace{1cm} (11)

4.4 Adaptive Branch Energy Distribution (ABED)

Previous works [6, 36] have input the directly concatenated embedded features of the three branches (object, image and previous action branches) into LSTM networks to establish time series models. However, there are two issues with this simple feature stitching: (i) It is difficult for the network to distinguish between the different information types of the three branches during training; (ii) It is difficult to guarantee that the data distribution of the concatenated vector is rational. Therefore, we propose the adaptive branch energy distribution (ABED) method to provide additional tokens to each branch without introducing extra parameters, and optimize the data distribution of the concatenated vector. We establish a learnable vector $\mathbf{r} = \{r_1, r_2, r_3\}$ with only three parameters. The final output vector $H_t$ can be expressed as:

$$ H_t = F_{pw} (\text{Concat}(r_1 \hat{I}_t, r_2 \hat{I}_t, r_3 \mathbf{P}_A)) $$ \hspace{1cm} (12)

where $PA$ is the previous action embedding and $F_{pw}$ refers to the pointwise convolution. The operation, which is similar to the energy distribution of the input signal, uses the significant differences between the feature distributions of the three branches to explicitly distinguish the branches and learn a more reasonable distribution of the concatenated vector. Although our method is quite simple, experiments have proven that it can significantly improve the navigation ability of the agent. When compared to the process of directly adding branch semantic (BS) embeddings to each branch, this method is unique in that it can provide the model with a strong scene understanding without destroying the other modules in complex models.

4.5 Policy Learning

Previous works [32, 33] have used direct observations to learn a strategy $\pi(a_t | o_t, p)$. Our work uses unbiased object graph relationships to learn an LSTM action policy $\pi(a_t | H_t, p)$, where $H_t$ is the joint representation of the global image embedding $r_1 \hat{I}_t$, object embedding $r_2 \hat{S}_t$ and previous action embedding $r_3 \mathbf{P}_A$. Based on previous works [7, 19], we treat this task as a reinforcement learning problem and utilize the asynchronous advantage actor-critic (A3C) algorithm [21], which applies policy gradients to assist the agent in choosing an appropriate action $a_t$ in the high-dimensional action.
space $A$. In accordance with the done reminder operation presented in [36], we use the target detection confidence to explicitly enhance the probability of the Done action.

## 5 EXPERIMENTS

### 5.1 Experimental Setup

**5.1.1 Dataset.** We choose the AI2-Thor [13] dataset and its corresponding simulator as the experimental platform. The AI2-Thor dataset includes 30 different floorplans for each of 4 room layouts: kitchen, living room, bedroom, and bathroom. For each scene type, we use 20 rooms for training, 5 rooms for validation, and 5 rooms for testing. There are 22 kinds of objects ($N = 22$) that agents can recognize, and we ensure that there are at least 4 kinds of objects in each room [32].

**5.1.2 Evaluation Metrics.** We use the success rate (SR), success weighted by path length (SPL) [1], and success weighted by action efficiency (SAE) [36] metrics to evaluate our method. SR indicates the success rate of the agent in completing the task, which is formulated as $SR = \frac{1}{E} \sum_{i=1}^{E} S_{uci}$, where $E$ is the number of episodes and $S_{uci}$ indicates whether the $i$-th episode succeeds.

SPL considers the path length more comprehensively and is defined as $SPL = \frac{1}{E} \sum_{i=1}^{E} S_{uci} L^{\star}_{\min}$, where $L_{\star}$ is the path length taken by the agent and $L^{\star}_{\min}$ is the theoretical shortest path. SAE considers the effects of unnecessary rotations and is defined as $SAE = \frac{1}{E} \sum_{i=1}^{E} S_{uci} \frac{T_i}{\max(L_i, L^{\star})}$, where $L_i$ is the path length taken by the agent and $L^{\star}$ is the theoretical shortest path. SAE considers the effects of unnecessary rotations and is defined as $SAE = \frac{1}{E} \sum_{i=1}^{E} S_{uci} \frac{T_i}{\max(L_i, L^{\star})}$, where $L_i$ is the path length taken by the agent and $L^{\star}$ is the theoretical shortest path.

**5.1.3 Implementation Details.** We train our model with 18 workers on 2 RTX 2080Ti Nvidia GPUs. The Adam optimizer [12] is used to update the network parameters with a learning rate of $10^{-4}$. We introduce a dropout of 0.3 to the multi-head attention mechanism and global image embedding. The confidence threshold for filtering objects is set to 0.6. Faster-RCNN [27] is fine-tuned on 50% [36] of the training data from the AI2-Thor dataset. For evaluation, our results take the average value of the test for 3 times. We report the results for all targets (ALL) and for a subset of targets (L=$\geq 5$) with optimal trajectory lengths greater than 5.

### 5.2 Strong and Concise Baseline

The methods presented in [6, 26, 36] include five kinds of branches (image, zone, room, previous action and object) that are concatenated into a vector before being input into the LSTM network. To evaluate the influence of the five branches separately on the object navigation task, we eliminate each branch in turn, as shown in Table 1. The object branch has the greatest impact on the experimental results, and the removal of the object branch drops 20.27/24.27, 17.62/19.38, 8.40/11.02 in SR, SPL and SAE (ALL/L=$\geq 5$, %). This confirms the importance of object features in object navigation task. In this perspective, the adequate exploration of object relationships is necessary. Moreover, the image and previous action branches also have significant impacts on the agent’s navigation ability. Whereas, the room branch and the zone branch have little effect on the SR and SPL. Accordingly, our baseline retains only the image, object and previous action branches.

The last row in Table 1 shows our simplified baseline. The removal of the object GCN, room branch and zone branch reduces the agent’s exploration time by 27.2% while leaving the SR, SPL and
SAE essentially unchanged. This more concise baseline allows us to observe the advantages of the added modules more clearly.

5.3 Ablation Experiments

We verify the effectiveness of each proposed module with extensive experiments. Table 2, Table 3, Table 4 and Table 5 show the results of ablation experiments on the UAIA, DOA graph, ABED and overall model. More ablation experiments are provided in the Supplementary Material.

5.3.1 UAIA. As shown in Table 2, the UAIA module has two main components: the confidence filter (CF), which is used to eliminate outlier objects; the pixel index (PI) embedding, which is used to increase the order-awareness of the global image features. The UAIA module with CF (confidence > 0.6) outperforms the UAIA module without CF by 1.64/1.11 in SR (ALL/L >= 5, %). This result shows that reducing the influence of irrelevant objects in the UAIA module can effectively improve the navigation ability. The UAIA module with PI embedding outperforms the UAIA module without PI embedding by 1.78/2.25 in SPL (ALL/L >= 5, %), demonstrating that adding positional encoding to each pixel in the image can optimize the agent’s path. The two components complement each other to improve the effect of the UAIA module.

5.3.2 DOA Graph. As shown in Table 3, we explore the role of intrinsic graph (IG) and view adaptive graph (VAG) in DOA graph. The DOA graph with only IG outperforms the DOA graph without IG by 2.90/1.96, 1.47/1.19 and 2.25/3.24 in SR, SPL and SAE (ALL/L >= 5, %). However, the DOA graph with only VAG brings a blow to the agent’s navigation ability. This result implies that it is difficult to directly perform fully adaptive learning in object relationships, requiring the extensive prior knowledge to narrow the learning domain. During the calculation of VAG, the muti-head attention (MA) allows for more reasonable object graph relationships, while the use of target semantics (TS) allows the agent to be more specific about the target, thereby improving the navigation efficiency.

5.3.3 ABED. In Table 4, we compare two methods of providing identities for the branches: branch semantics (BS), which use the embedding of one-hot vectors; energy distribution (ED), which uses only three energy distribution coefficients. Without cross-attention (UAIA and UAOA), adding BS and ED to the original model improves the agent’s navigation ability well. Notably, with cross-attention (UAIA and UAOA), adding BS to the complex model causes the model to crash. In contrast, the simple ED method improves the complex model with cross-attention by 0.88/0.87, 0.72/1.02 and 0.67/2.01 in SR, SPL and SAE (ALL/L >= 5, %). The results demonstrate the significant advantage of the ED method with only three parameters in complex models. This is consistent with our intuition that due to the complexity of object navigation tasks, the learning model must be simplified; otherwise, the strong coupling of the complex parameters between modules can cause the overall model to be difficult to learn.

5.3.4 Overall Model. The ablation experiments on the overall model with UAOA, UAIA and ABED are shown in Table 5. Compared with our proposed baseline, applying the complete model increases SR, SPL and SAE by 5.18/7.46, 2.40/3.08 and 2.02/3.86 (ALL/L >= 5, %). The results indicate that our method is capable of effectively guiding agents to navigate in unseen environments. Compared with the UAIA and ABED methods, the UAOA method improves the model more significantly. This is because the UAOA method essentially solves the object attention bias problem, increasing the agent’s understanding of the relationships between objects.

5.4 Elimination of Object Attention Bias

Object attention bias is the phenomenon in which objects with low visibility are ignored. Figure 4 (a) shows the average attention of the agent on all objects before and after using the DOA graph for all test floorplans. Without the use of our DOA graph approach, the agent suffers from severe long-tail distribution in the object attention. The average attention difference between the most popular object and the most neglected object is more than tenfold. Objects with high visibility, such as the floor lamp, fridge and sink, dominate the agent’s decision-making, while objects with low visibility, such as the cell phone and remote control, cannot play their proper guiding roles. Figure 4 (b) shows that although the DOA method makes the average attention for each object to be similar across the entire dataset, there is a correct attention tendency in different scenes. Our DOA graph-based attention consists of two parts (section 4.2): intrinsic attention and adaptive attention. The agent adjusts the two-part attention in different scenes to pay more attention on critical objects. In summary, the proposed DOA graph significantly improves the rationality of attention allocation for most objects, indicating that the improvement in SR, SPL and SAE is indeed from solving the object attention bias problem. We emphasize that DOA...
Table 6: Comparison with SOTA methods (%). More experiments are in the appendix D.1.

| Method         | ALL |       |       | L>=5 |       |       |
|----------------|-----|-------|-------|------|-------|-------|
|                | SR  | SPL   | SAE   | SR   | SPL   | SAE   |
| Random         | 4.12| 2.21  | 0.43  | 0.23 | 0.08  | 0.05  |
| SP [33]        | 62.98| 38.56 | 24.99 | 52.73| 33.84 | 23.02 |
| SAVN [32]      | 63.12| 37.81 | 20.98 | 52.01| 34.94 | 23.01 |
| ORG [6]        | 67.32| 37.01 | 25.17 | 58.13| 35.90 | 27.04 |
| HOZ [36]       | 68.53| 37.50 | 25.98 | 60.27| 36.61 | 27.68 |
| Ours (Baseline)| 69.14| 37.87 | 27.77 | 60.42| 37.38 | 28.70 |
| Ours (DOA+ABED)| 74.32| 40.27 | 29.79 | 67.88| 40.36 | 32.56 |

As shown in Table 6, we compare the test results of our method and other similar methods on the AI2-Thor dataset. All of the random decision navigation indicators are close to 0. Notably, our baseline model outperforms the SOTA method by 0.61/0.15, 0.37/0.67 and 1.97/1.02 in SR, SPL and SAE (ALL/L >= 5, %). The redundant operations in previous networks aggravate the object attention bias, which explains why the subtraction in our baseline model facilitates learning. Finally, our model with DOA graph-based modules and ABED method outperforms the proposed baseline with the gains of 5.18/7.46, 2.40/3.08 and 2.02/3.86 in SR, SPL and SAE (ALL/L >= 5, %).

5.5 Comparisons to the State-of-the-art

As shown in Table 6, we compare the test results of our method and other similar methods on the AI2-Thor dataset. All of the random decision navigation indicators are close to 0. Notably, our baseline model outperforms the SOTA method by 0.61/0.15, 0.37/0.67 and 1.97/1.02 in SR, SPL and SAE (ALL/L >= 5, %). The redundant operations in previous networks aggravate the object attention bias, which explains why the subtraction in our baseline model facilitates learning. Finally, our model with DOA graph-based modules and ABED method outperforms the proposed baseline with the gains of 5.18/7.46, 2.40/3.08 and 2.02/3.86 in SR, SPL and SAE (ALL/L >= 5, %).

5.6 Qualitative Analysis

We visualize the navigation effect of the agent in Figure 5. The direction and stop timing of the rotation are critical, as seen in the trajectories of the success and failure cases. These two decisions are mainly determined by the agent’s interpretation of the scene at keyframes when multiple objects can provide rich information. Our DOA graph-based method provides the agent with a more reasonable and unbiased attention allocation at these keyframes, allowing the algorithm to choose the correct rotation direction and stop timing.

6 CONCLUSION

Based on the analysis of the network structure and the failed navigation of previous methods, we identify the agent’s object attention bias problem in navigation tasks. To address this problem, we use a directed object attention (DOA) graph, which allows the agent to unbiasedly redistribute object attention. Cross-attention modules (UAIA and UAOA) between the object branch and image branch are designed according to the DOA graph. Our experimental results show that our approach can effectively address the object attention bias problem, greatly improving the agent’s navigation ability. Furthermore, we propose an adaptive branch energy distribution (ABED) method for optimizing the aggregation of branch features that performs well in complex models. It is worth noting that we prioritize simplicity in our approach. In future work, we will attempt to determine more concrete relationships between objects so that the agent’s navigation can be more clearly interpreted.

ACKNOWLEDGMENTS

This paper is supported by the National Natural Science Foundation of China under Grants 61733013, 62173248, 62073245. Suzhou Key Industry Technological Innovation-Core Technology R&D Program, No. SGC2021035.
REFERENCES

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. 2018. On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06577 (2018).

[2] Remi Cadene, Corentin Danonceau, Matthieu Cord, Devi Parikh, et al. 2019. Ruby: Reducing unimodal biases for visual question answering. Advances in neural information processing systems 32 (2019).

[3] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. 2020. Object goal navigation using goal-oriented semantic exploration. Advances in Neural Information Processing Systems 33 (2020), 4247–4258.

[4] Matt Deitke, Winson Han, Alvaro Herrasti, Arinudhri Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenn, Eli VanderBilt, Matthew Wallingford, et al. 2020. Roborobo: An open simulation-to-real embodied ai platform. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 3164–3174.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is a sequence of tokens. arXiv preprint arXiv:2103.00020 (2021).

[6] Heming Du, Xin Yu, and Liang Zheng. 2020. 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 (Glasgow, United Kingdom). 19–34.

[7] Qiang Fang, Xin Xu, Xiaogang Wang, and Yujun Zeng. 2021. Target-driven visual navigation in indoor scenes using reinforcement learning and imitation learning. CAAI Transactions on Intelligence Technology (2021).

[8] Chen Gao, Jinyu Chen, Si Liu, Luting Wang, Qiong Zhang, and Qi Wu. 2021. Room-and-object aware knowledge reasoning for remote embodied referring expression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 3064–3073.

[9] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. 2019. Perception of texture; increasing shape bias improves accuracy and robustness. In International Conference on Learning Representations.

[10] Kaiqing He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.

[11] Aditya Khosla, Tinghui Zhou, Jana Kosecka, Jitendra Malik, Alan L Yuille, and Alexei A. Efros. 2011. Unbiased look at dataset bias. In Computer Vision – ECCV 2012. Springer Berlin Heidelberg, Berlin, Heidelberg, 158–171.

[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In ICLR 2015, Bellevue, WA, May 7-9, 2015.

[13] Qiushuang Li, Xinhang Song, Yubing Bai, Sixian Zhang, and Shuqiang Jiang. 2021. ION: Unbiased Directed Object Attention Graph for Object Navigation MM ’22, October 10–14, 2022, Lisboa, Portugal.

A MORE ABLATION EXPERIMENTS

A.1 Choice of Backbone

In the main text, we use ResNet18 as the backbone network for extracting the global image features. Additionally, we also experiment with backbones of different depths. Table 7 shows the results after we replace the backbone with ResNet50 and ResNet101. Compared to ResNet18, the use of ResNet50 decreases the SR, SPL and SAE by 0.20/1.11, 4.29/6.86 and 0.13/0.02 (ALL/L=5%). The losses in the SR, SPL and SAE when using ResNet101 as the backbone network are even greater. It is clear that the SAE and the episode length are most severely affected. The explanation may be that excessive high-level global image information degrades local information, thereby weakening the effect of cross-attention with the object branch and
imparing the action selection at each step in the overall path. As a result, the navigation distance is greatly increased, which affects the SPL.

### A.2 Pixel Index Embedding

In the UAIA module, the pixel index embeddings provide each pixel’s position token, which is important in the multi-head attention mechanism. In Table 7, we compare the results of the following encodings:

1) Providing no positional information (No): Considering the image feature maps as a bag of pixel vectors.
2) 1-dimensional pixel index embedding (Our Best Model): Considering the image feature maps as a sequence of pixel vectors in order.
3) 2-dimensional pixel index embedding (2-D): Considering the image feature maps as a grid of pixel vectors in two dimensions.
4) Relative pixel index embedding (Relative): Considering the relative distance between pixels rather than their absolute position to encode the spatial information.

The results show that without pixel index embeddings, the SPL is reduced by approximately 2%. However, when using 1-D, 2-D or relative pixel index embeddings, the effect is essentially the same. We speculate that this occurs because we perform position embeddings on feature maps with smaller spatial dimensions instead of on the original image, which has larger spatial dimensions. Hence, the process of learning the spatial relations at this resolution is equally easy for the different positional encoding strategies.

### A.3 Directivity in DOA Graph

In the main text, we repeatedly emphasize that our proposed DOA graph is a directed graph; thus, the correlation between object B and object A when searching for object A is different from the correlation between object A and object B when searching for object B. Table 7 illustrates that the navigation improvement decreases significantly when the DOA graph is an undirected graph. Fundamentally, the directed object graph follows the relative relationships between objects, while the undirected object graph follows the absolute relationships between objects. The absolute object relationships reduce the adaptability of the agent in modeling object relationships in different scenes, thus reducing its performance in unseen environments.

### B MORE COMPARISONS WITH THE RELATED WORKS

#### B.1 Comparisons in RoboTHOR Dataset

We compare our method with the other related works on the AI2-Thor dataset in the main text. We consider that the scenes in AI2-Thor are relatively simple, so the RoboTHOR [4] dataset is used.

---

Table 7: Navigation performance after replacing some modules in the model (%). Our Best Model represents the model used in our main text.

| Module Replacement | ALL | SAE | L>=5 | Episode Length |
|--------------------|-----|-----|------|----------------|
| Our Best Model     | 74.32 ±0.89 | 40.27 ±1.01 | 29.79 ±0.80 | 67.88 ±1.05 | 40.36 ±1.23 | 32.56 ±1.76 | 22.86 ±1.32 |
| Backbone           | ResNet50 | 74.12 ±1.23 | 35.98 ±1.56 | 28.76 ±0.90 | 66.77 ±1.12 | 33.50 ±1.27 | 32.34 ±1.52 | 27.96 ±0.88 |
| Object Detector    | Ground Truth | 70.21 ±1.45 | 35.56 ±1.67 | 26.77 ±1.34 | 65.43 ±0.81 | 31.45 ±1.23 | 29.70 ±0.99 | 28.21 ±1.29 |
| Pixel Index Emb.   | No | 74.54 ±1.20 | 38.14 ±0.82 | 29.13 ±1.03 | 67.76 ±1.38 | 38.55 ±1.07 | 32.67 ±1.21 | 24.21 ±0.94 |
|                    | 2-D | 74.29 ±1.22 | 40.54 ±0.79 | 29.38 ±1.62 | 67.81 ±1.09 | 40.35 ±1.67 | 32.29 ±1.24 | 22.59 ±0.93 |
|                    | Relative | 74.10 ±0.85 | 40.34 ±1.07 | 29.33 ±1.55 | 67.45 ±1.38 | 40.19 ±1.43 | 31.98 ±1.33 | 23.45 ±0.74 |
| DOA Graph          | Undirected | 73.15 ±1.17 | 38.44 ±1.26 | 27.43 ±1.63 | 66.33 ±1.44 | 37.60 ±1.10 | 30.95 ±1.28 | 26.84 ±1.36 |

Figure 6: Learned adjacency matrices of the intrinsic object graph and view adaptive graph.
which represents the ubiquitous inherent relationship between the value is on the diagonal, indicating that the self-connecting edge has the largest weight. This result is consistent with our assumption that when the target object is identified, most of the attention is focused on the target object. For the adjacency matrix of the view adaptive graph, we randomly select the view adaptive attention when looking for different targets as a column. Interestingly, at each timestamp in the view adaptive graph, only one object’s attention weight is close to 1, while the other object’s attention weight is close to 0. We speculate that because there are often very few objects observed at each timestamp and most of these objects are essentially irrelevant to the target object, the view adaptive graph directs all attention to the most important object. In the weighted sum of the two graphs, the weight of the intrinsic object graph is 0.95, while the weight of the view adaptive graph is only 0.05. Therefore, although the attention distribution of the view adaptive graph is extreme, it does not affect the robustness of the overall attention distribution. In contrast, the attention concentration in the view adaptive graph reasonably increases the weight of important objects in view.

Table 8: Comparison with SOTA methods in RoboTHOR [4] (%)

| Method        | SR  | SPL | SAE | SR  | SPL | SAE | Episode Length |
|---------------|-----|-----|-----|-----|-----|-----|----------------|
| Random        | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 3.01±0.32     |
| SP [33]       | 27.43±0.60 | 17.49±0.36 | 15.21±0.33 | 20.98±0.97 | 16.03±0.48 | 13.99±0.44 | 68.18±1.27    |
| SAVN [32]     | 28.97±0.84 | 16.59±0.65 | 13.21±0.37 | 22.89±0.98 | 15.21±0.41 | 13.77±0.30 | 67.22±0.93    |
| ORG [6]       | 30.51±0.21 | 18.62±0.19 | 14.71±0.47 | 23.89±0.55 | 14.91±0.38 | 13.73±0.50 | 69.17±2.13    |
| HOZ [36]      | 31.67±0.87 | 19.02±0.49 | 15.44±0.63 | 24.32±0.48 | 14.81±0.23 | 14.22±0.88 | 66.26±1.88    |
| Ours (Baseline)| 31.92±0.37 | 18.88±0.18 | 16.37±0.33 | 24.72±0.26 | 15.56±0.35 | 15.18±0.61 | 65.31±1.69    |
| Ours (DOA+ABED)| 36.22±0.36 | 22.12±0.42 | 18.70±0.59 | 30.16±0.28 | 18.32±0.34 | 19.21±0.52 | 61.24±1.23    |

Figure 7: Visualization of the navigation trajectory with the RoboTHOR dataset. The trajectory of the agent is indicated by the green and blue arrows, with the green indicating the beginning and the blue indicating the end of the trajectory. The red value in the object detection box represents the attention weight of the object.

Table 8 illustrates the performance of our method and other related methods on the RoboTHOR. Obviously, the various metrics on the RoboTHOR are generally low, which indicates that object navigation on RoboTHOR is much more difficult than AI2-Thor. Moreover, the episode length of navigation on RoboTHOR is 3 times that of AI2-Thor, which indicates that each navigation scene in RoboTHOR is much more complicated. Nevertheless, our method still outperforms the proposed baseline with an obvious margin by 4.3/5.44, 3.24/2.76 and 2.33/4.03 in SR, SPL and SAE (ALL/L>=5, %).

C QUALITATIVE RESULTS
C.1 The DOA Graph Visualization
Our proposed DOA graph has two parts: the intrinsic object graph, which represents the ubiquitous inherent relationship between the objects, and the view adaptive graph, which changes adaptively according to the input at each timestep. A weighted sum of the two graphs gives the attention score assigned to each object at this timestep. Figure 6 shows the adjacency matrix of the two graphs. For the adjacency matrix of the intrinsic object graph, the largest value is on the diagonal, indicating that the self-connecting edge has the largest weight. This result is consistent with our assumption that when the target object is identified, most of the attention is focused on the target object. For the adjacency matrix of the view adaptive graph, we randomly select the view adaptive attention when looking for different targets as a column. Interestingly, at each timestamp in the view adaptive graph, only one object’s attention weight is close to 1, while the other object’s attention weight is close to 0. We speculate that because there are often very few objects observed at each timestamp and most of these objects are essentially irrelevant to the target object, the view adaptive graph directs all attention to the most important object. In the weighted sum of the two graphs, the weight of the intrinsic object graph is 0.95, while the weight of the view adaptive graph is only 0.05. Therefore, although the attention distribution of the view adaptive graph is extreme, it does not affect the robustness of the overall attention distribution. In contrast, the attention concentration in the view adaptive graph reasonably increases the weight of important objects in view.

C.2 Navigation Trajectory in RoboTHOR
Since the AI2-Thor dataset only includes simple room constructions, we also use the RoboTHOR dataset, which includes more complex indoor environments, to verify the effectiveness of our model. As shown in Figure 7, in the RoboTHOR environment, the agent spends more time searching for the target object. Compared with the baseline, the agent can approach the target step-by-step in a continuous exploration with our proposed DOA graph.