Search for or Navigate to? Dual Adaptive Thinking for Object Navigation

Ronghao Dang[1] Liuyi Wang[1] Zongtao He[1] Shuai Su[1] Jiagui Tang[1] Chengju Liu[1,2*]
Qijun Chen[1]

[1]Tongji University [2]Tongji Artificial Intelligence (Suzhou) Research Institute
{dangronghao, wly, xingchen327, sushuai, 2130701, liuchengju, qjchen}@tongji.edu.cn

Abstract

"Search for" or "Navigate to"? When we find a specific object in an unknown environment, the two choices always arise in our subconscious mind. Before we see the target, we ***search for*** *the target based on prior experience. Once we have seen the target, we can* ***navigate to*** *it by remembering the target location. However, recent object navigation methods consider using object association mostly to enhance the "search for" phase while neglecting the importance of the "navigate to" phase. Therefore, this paper proposes a* ***dual adaptive thinking (DAT)*** *method that flexibly adjusts thinking strategies in different navigation stages. Dual thinking includes both search thinking according to the object association ability and navigation thinking according to the target location ability. To make navigation thinking more effective, we design a target-oriented memory graph (TOMG) (which stores historical target information) and a target-aware multi-scale aggregator (TAMSA) (which encodes the relative position of the target). We assess our methods based on the AI2-Thor and RoboTHOR datasets. Compared with state-of-the-art (SOTA) methods, our approach significantly raises the overall success rate (SR) and success weighted by path length (SPL) while enhancing the agent's performance in the "navigate to" phase.*

1. Introduction

Object navigation [27, 20, 23, 40] is a challenging task that requires an agent to find a target object in an unknown environment with first-person visual observations. Some researchers [29, 13, 17] recently introduced scene prior knowledge into end-to-end navigation networks. These methods have been applied to address various issues, including object associations [37], object attention bias [5], and the lack of universal knowledge [14]. However, these methods improve the efficiency of only the "search for" phase (start→the target is first seen) while neglecting the "navigate to" phase (the target is first seen→end). Our experiments show that for the current SOTA end-to-end methods, the "navigate to" steps account for 62.75% of the whole path, while only 45.78% for humans; the success rate after seeing the target is only 81.09%, while humans can reach 99.92%. Therefore, the primary issue with current end-to-end object navigation techniques is the low navigation efficiency in the "navigate to" phase.

Some modular approaches [4, 30] model the environment by using top-down semantic maps [28, 32]. With the help of detailed semantic maps, the object navigation task can be decoupled into two training subtasks: predicting the subtarget point and navigating to the subtarget point, thus optimizing the agent navigation ability after seeing the target. However, these methods depend strongly on semantic maps, which are hypersensitive to sensory noise and scene changes. Furthermore, generating high-quality semantic maps requires considerable computational resources.

To address the above issues, we aim to integrate this task decoupling concept in modular methods into end-to-end methods. Therefore, we propose the dual adaptive thinking (DAT) method. As shown in Figure 1, the agent's thinking modes are divided into search thinking and navigation thinking. Search thinking guides the agent to quickly locate the target according to prior knowledge. Navigation thinking assists the agent in efficiently navigating to the target position after locating the target. The agent adaptively adjusts the dominance of the two thinking methods in an end-to-end network according to the navigation progress.

Specifically, we develop different designs for the search thinking network and navigation thinking network. For the search thinking network, we adapt the directed object attention (DOA) graph method proposed in [5] to design object association and attention allocation strategies. For the navigation thinking network, we propose a target-oriented memory graph (TOMG) to store the simplified agent state and

*Corresponding author

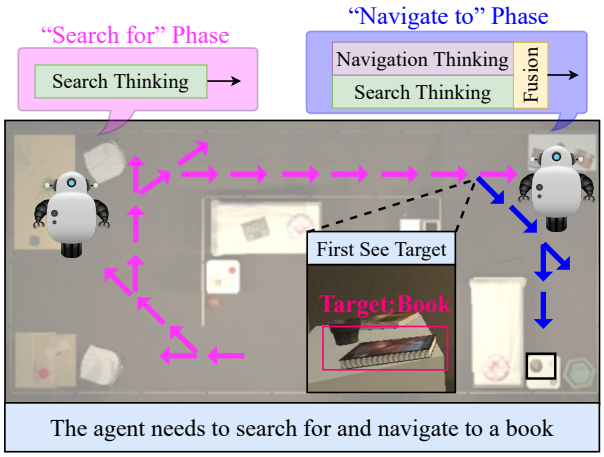


Figure 1. The first target-visible frame divides the agent's navigation process into two phases: "search for" (pink) and "navigate to" (blue). During the "search for" phase, the agent uses only search thinking to search for the target. During the "navigate to" phase, navigation thinking assists the agent in quickly navigating to the target location.

target orientation information. Furthermore, we design a target-aware multi-scale aggregator (TAMSA) to refine the features in the TOMG to guide the agent's navigation.

Extensive experiments on the AI2-Thor [18] and RoboTHOR [6] datasets show that our DAT method not only optimizes the "navigate to" phase in the end-to-end network but also outperforms the state-of-the-art (SOTA) method [5] by 8.07% and 8.66% in the success rate (SR) and success weighted by path length (SPL). Moreover, we propose three new metrics, search success rate (SSR), navigation success rate (NSR) and navigation success weighted by navigation path length (NSNPL), to respectively assess the agent's search ability during the "search for" phase and the navigation ability during the "navigate to" phase. As a general concept, the proposed multiple-thinking strategy can be applied in various other embodied artificial intelligence tasks. Our contributions can be summarized as follows:

- We propose a dual adaptive thinking (DAT) method that allows the agent to flexibly use different modes of thinking during navigation.
- We carefully design a navigation thinking network with a selective memory module (TOMG) and a feature refinement module (TAMSA) to implicitly encode the target location into the end-to-end network.
- We demonstrate that our DAT method not only addresses inefficiencies in the "navigate to" phase but also substantially outperforms existing object navigation models.

2. Related Works

2.1. Object Navigation

Object navigation tasks [2, 35, 33] require an agent to navigate to a target object in an unknown environment while considering only visual inputs. Recently, the relationships between objects have been introduced into navigation networks, allowing agents to locate targets more quickly by considering object associations. Zhang et al. [39] proposed the hierarchical object-to-zone (HOZ) graph to guide an agent in a coarse-to-fine manner. Moreover, Dang et al. [5] utilized a directed object attention (DOA) graph to address the object attention bias problem. These works allow agents to locate targets faster but do not address how to navigate to these targets more quickly. Our dual adaptive thinking (DAT) method divides agents' thinking into two types: search thinking and navigation thinking, which can collaborate adaptively to make every navigation stage efficient.

2.2. Modular Navigation

Modular navigation methods [4, 30] have been proposed to solve the generalizability problem of end-to-end models in complex environments. It has been proven that using a top-down semantic map to predict distant subgoal points [4] is feasible on the Habitat dataset. The PONI [30] method trains two potential function networks by using supervised learning to determine where to search for an unseen object. These modular methods require considerable computing and storage resources to generate semantic maps in real time and are sensitive to image segmentation quality. Our method implicitly incorporates different thinking during navigation into an end-to-end network without relying on semantic maps.

3. Necessity of Dual Thinking

3.1. Dual Thinking in Humans

Embodied AI [10] is a challenging research topic that requires agents to use well-developed intuitive tasks (e.g., classification [34] and detection [22]) to complete complex logical tasks (e.g., navigation [41] and interaction [31]) in real-world environments. Humans often use multiple ways of thinking when completing these complex logical tasks. For example, when we need an object, we first use associative thinking to locate the object and then use navigational thinking to reach the object location; when we answer a question about an object, we first use exploratory thinking to fully understand the object and then use reasoning and language-organized thinking to draw conclusions. Therefore, multiple thinking approaches can be introduced in end-to-end networks to develop interpretable hierarchical models that are more consistent with how humans address complex logic problems.

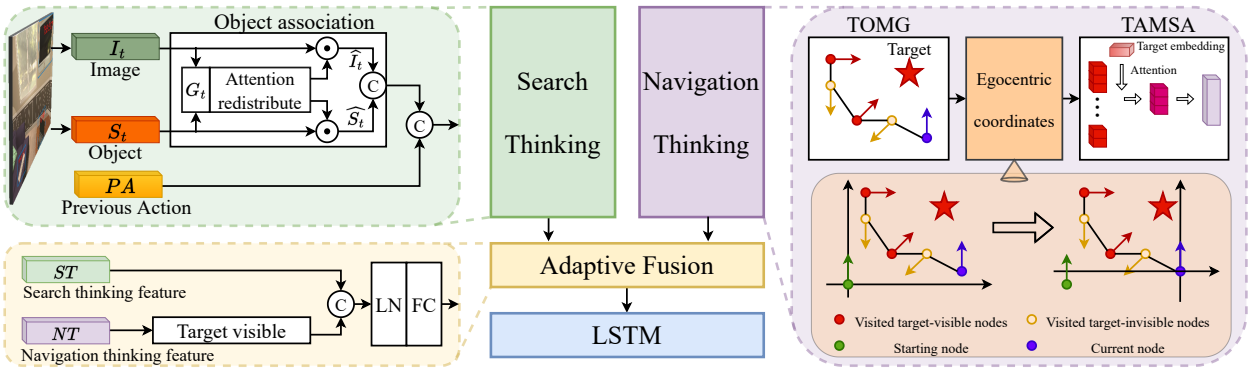


Figure 2. Model overview. TOMG: target-oriented memory graph. TAMSA: target-aware multi-scale aggregator. Our model includes three modules: search thinking, navigation thinking and adaptive fusion. In the search thinking network, we endow the model with an object association ability according to the directed object attention (DOA) graph method proposed in [5]. In the navigation thinking network, we provide the model with the ability to remember the target orientation. In the adaptive fusion network, we make the dual thinking work in harmony according to the navigation progress.

3.2. Repeated Target Search Problem

In current methods, if the agent loses the target in view, the target must still be searched for again to locate it. Consequently, the agent wastes considerable time in re-searching for the target, potentially leading to constant loops. This problem is especially common in environments with many obstacles. A clear orientation memory for the target is the key to solve this problem. Therefore, we design a target-oriented memory graph (TOMG) and a target-aware multi-scale aggregator (TAMSA) in the navigation thinking network to ensure that the agent navigates to the target efficiently without repeatedly re-searching.

4. Dual Adaptive Thinking Network

Our goal is to endow agents with both search and navigation thinking and to adjust their status based on the navigation process. To achieve this goal, we design three networks, as illustrated in Figure 2: (i) search thinking network; (ii) navigation thinking network; (iii) adaptive fusion network. (i) and (ii) are connected by (iii) to form the dual adaptive thinking (DAT) network.

4.1. Task Definition

The agent is initialized to a random state $s = \{x, y, \theta, \beta\}$ and random target object p. Here, (x, y) represents the coordinates of the agent, (θ, β) represents the yaw and pitch angles of the agent. At each timestamp t, according to the single view RGB image o_t and target p, the agent learns a navigation strategy $\pi(a_t|o_t, p)$, where $a_t \in A = \{MoveAhead; RotateLeft; RotateRight; LookDown; LookUp; Done\}$ and $Done$ is the output if the agent believes that it has navigated to the target location. Ultimately, if the agent is within a threshold (i.e., 1.5 meters [8]) of the target object when $Done$ is output, the navigation episode is considered successful.

4.2. Search Thinking Network

Search thinking aims to enable the agent to quickly capture the target with the fewest steps when the target is not in view. To use efficient object association, we adopt the unbiased directed object attention (DOA) graph method proposed in [5]. As shown in the green box in Figure 2, according to the object-target association score G_t calculated by the DOA method, we redistribute the attention to the object features S_t (from DETR [3]) and image features I_t (from ResNet18 [15]) to ensure that the agent pays attention to objects and image regions that are more relevant to the target.

In the object attention redistribution process, the object-target association score of each object q is multiplied by the object features S_t to generate the final object embedding $\widehat{S}_t$:

$$\widehat{S}_t^q = S_t^q G_t^q \qquad q = 1, 2, \cdots, N \tag{1}$$

where $\widehat{S}_t = \{\widehat{S}_t^1, \widehat{S}_t^2, \cdots, \widehat{S}_t^N\}$, and N is the number of objects.

In the image attention redistribution process, we assign attention to image features I_t according to the object semantic embeddings generated by the one-hot encodings. Initially, the semantic embeddings are weighted by $G_t \in \mathbb{R}^{N\times 1}$ to obtain the attention-aware object semantics D. We use D as the query and I_t as the key and value in the multi-head image attention to generate the final image embedding $\widehat{I}_t$:

$$Q_i = DW_i^Q \quad K_i = I_t W_i^K \quad V_i = I_t W_i^V \quad i = 1, \cdots, NH \tag{2}$$

$$head_i = softmax(\frac{Q_i K_i^T}{\sqrt{HD}})V_i \tag{3}$$

$$\widehat{I}_t = Concat(head_1, \cdots, head_{NH})W^O \tag{4}$$

where HD and NH denote the hidden dimensionality and number of heads in the multi-head attention.

Finally, the attention-aware object features $\widehat{S}_t$ and image features $\widehat{I}_t$ are concatenated with the previous action embedding PA to obtain the output ST of the search thinking network.

4.3. Navigation Thinking Network

Target-Oriented Memory Graph (TOMG) Different from search thinking, navigation thinking requires the ability to memorize, locate and navigate to the target. Thus, we design a target-oriented memory graph (TOMG) as the input feature M. There are two types of nodes in the history route map (see the purple box in Figure 2): (i) visited target-visible nodes where the agent detects the target in view; (ii) visited target-invisible nodes where the agent does not detect the target in view. Navigation thinking only focuses target-related information; thus the TOMG is composed of the visited target-visible nodes. Previous historical memory methods [12, 42] which store both the images $m_i \in \mathbb{R}^{7\times7\times512}$ and objects $m_o \in \mathbb{R}^{N\times256}$ features contain too much redundant noise. In contrast, our TOMG node only stores high-level features $m \in \mathbb{R}^{1\times9}$ which is concatenated by three parts: the target bounding box, the target confidence and the agent's coordinates. By eliminating redundant inputs, our target-oriented storing method uses $3000\times$ less storage than previous methods [12, 42].

Since the agent cannot obtain its own absolute position and orientation in unknown environments, the stored coordinates $(x_i, y_i, \theta_i, \beta_i)$ are calculated relative to the starting coordinate $(x_0, y_0, \theta_0, \beta_0)$. Target-visible nodes are filtered by a confidence threshold cf of target recognition. Finally, to ensure the reliability of target orientation prediction, only the L closest target-visible nodes to the current node in the path are stored.

Egocentric Coordinate Transformation As the agent navigates during each step, the decisions (e.g., rotate right) are made relative to the current agent's own coordinate system. Therefore, before using the TOMG features, we convert the coordinates of each node in the TOMG to the coordinate system of the current node $(x_c, y_c, \theta_c, \beta_c)$ (see the orange box in Figure 2):

$$\begin{aligned}(\widetilde{x}_i, \widetilde{y}_i) &= (x_i, y_i) - (x_c, y_c) \\ (\widetilde{\theta}_i^x, \widetilde{\beta}_i^x) &= sin((\theta_i, \beta_i) - (\theta_c, \beta_c)) \\ (\widetilde{\theta}_i^y, \widetilde{\beta}_i^y) &= cos((\theta_i, \beta_i) - (\theta_c, \beta_c)) \quad i \in \Delta_M\end{aligned} \tag{5}$$

where Δ_M represents the index collection of target-visible nodes. To ensure that the angle and position coordinates have the same order of magnitude, we use sin and cos to normalize the angle coordinates to $[-1, 1]$. After this egocentric coordinate transformation, we obtain egocentric TOMG features $\widetilde{M} \in \mathbb{R}^{L\times11}$.

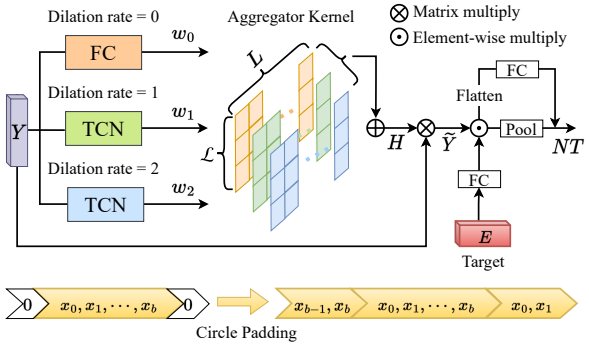


Figure 3. A detailed explanation of the target-aware multi-scale aggregator (TAMSA). We first use the multi-scale TCNs to obtain aggregator kernels that aggregate the target-oriented memory graph (TOMG) with L nodes into graph with $\mathcal{L}$ nodes. Then, the aggregated features allocate attention to the channel dimension using the target semantics. We describe the circle padding method applied in our TCNs below the figure.

Target-Aware Multi-Scale Aggregator (TAMSA) To encode navigation thinking into the network, we design a target-aware multi-scale aggregator (TAMSA) to aggregate the egocentric TOMG feature $\widetilde{M}$ into an implicit representation NT. In contrast to typical methods that use transformers or temporal convolutions as encoders, we devise a unique dynamic encoder that better leverages the memory graph features, as described below.

First, to improve the feature expression ability of the navigation thinking network, we use fully connected (FC) layers to map the features $\widetilde{M}$ to higher dimensional spaces. Inspired by some advanced works [7, 24] on vision transformers, we add layer normalization between the two FC layers to stabilize the forward input distribution and backpropagation gradient [36]. The encoding details can be formulated as follows:

$$Y = \delta(LN(\widetilde{M}W^{M_1})W^{M_2}) \tag{6}$$

where δ denotes the ReLU function, LN denotes layer normalization, and $W^{M_1} \in \mathbb{R}^{11\times16}$ and $W^{M_2} \in \mathbb{R}^{16\times32}$ are learnable parameters.

Then, a multi-scale dynamic kernel is calculated to refine the target orientation features into implicit nodes. As shown in Figure 3, we use three temporal convolution networks (TCNs) with different dilation rates d to generate three dynamic kernels with distinct scales. It is worth noting that the TCN with $d = 0$ degenerates to an FC layer. In the early stages of the "navigate to" phase, the TOMG contains fewer valid nodes; thus, the boundary degradation caused by zero padding has a greater impact. To avoid padding with zero,

Table 1. Ablation results on each module in the three sub-networks: search, navigate and fusion.

ID	Search Thinking		Navigation Thinking			Fusion		ALL (%)					Episode
	Associate	Pretrain	TOMG	Egocentric	TAMSA	AF	LN	SR↑	SPL↑	SSR↑	NSR↑	NSNPL↑	Time (s)↓
1								71.34	40.36	92.41	76.19	43.74	0.258
2	✓							74.43	40.93	95.82	77.67	44.11	0.334
3	✓	✓						76.78	43.88	94.19	81.40	47.09	0.334
4	✓	✓	✓					76.02	43.15	93.41	80.21	47.12	0.336
5	✓	✓	✓	✓				78.12	42.01	95.12	82.13	45.80	0.336
6	✓	✓	✓		✓			78.04	45.67	94.83	82.29	48.44	0.345
7	✓	✓	✓	✓	✓			80.88	45.71	95.46	**84.72**	48.31	0.346
8	✓	✓	✓	✓	✓	✓		81.34	47.53	96.38	84.39	49.74	0.350
9	✓	✓	✓	✓	✓	✓	✓	**82.39**	**48.93**	**97.25**	84.71	**50.32**	0.352

inspired by [38], we design the circle padding (CP) which fills the sequence edge with the features at the other end of the sequence (Figure 3). The different scale kernels are added after multiplying by the learnable parameter w_d:

$$H(l) = \sum_{d=0}^{2} w_d (\sum_{j \in \Psi} Y(l + j * d) * f_d(j) + b_d) \quad (7)$$

where $H = \{H(1), \cdots, H(L)\}$, l is the central node of the convolution kernel, Ψ refers to the set of offsets in the neighborhood considering convolution conducted on the central node, $Y(\cdot)$ takes out the node features in Y, and f_d and b_d denote the weights and biases in the convolution kernel with dilation rate d. The multi-scale dynamic kernel $H \in \mathbb{R}^{L \times \pounds}$ refines $Y \in \mathbb{R}^{L \times 32}$ to $\widetilde{Y} \in \mathbb{R}^{\pounds \times 32}$.

Intuitively, the mappings between the observation data and target azimuth differ when searching for different targets. For example, when looking for a TV, even if the TV is located far from the agent, the agent can clearly identify the target and obtain a larger target bounding box; however, when looking for a mobile phone, the agent can only obtain a smaller target bounding box, even if the agent is close to the mobile phone. Therefore, we enhance the TAMSA representation by considering the target semantic information. To achieve this goal, the one-hot target index E is encoded to the same channel dimension as $\widetilde{Y}$ through two FC layers, whose result is channel-wise multiplied with $\widetilde{Y}$ to get the target-aware feature representation $\widehat{Y}$:

$$\widehat{Y} = H^T Y \odot \delta(\delta(E W^{E_1}) W^{E_2}) \quad (8)$$

Finally, to obtain the final output NT of the navigation thinking network, we flatten $\widehat{Y}$ from $\mathbb{R}^{\pounds \times 32}$ to $\mathbb{R}^{1 \times 32\pounds}$ and use an FC layer to reduce the output dimension. Furthermore, we add residual connections to ensure the stability of the feature transfer process.

$$NT = \delta(Flatten(\widehat{Y}) W^Y) + \frac{1}{\pounds} \sum_{l=1}^{\pounds} \widehat{Y}(l) \quad (9)$$

A dropout layer is added before the output to reduce overfitting in the navigation thinking network.

4.4. Adaptive Fusion (AF) of Dual Thinking Networks

Search thinking and navigation thinking have different work strategies according to the navigation progress. During the "search for" phase, since there are no visited target-visible nodes, NT is an all-zero matrix. Therefore, the navigation thinking network does not affect the action decision when the target has not yet been seen. During the "navigate to" phase, to ensure navigation robustness, search thinking and navigation thinking work together to guide the action decision. As the number of visited target-visible nodes increases, navigation thinking gradually dominates. The fusion process of the two thinking methods can be expressed as:

$$DT = (LN(Concat(NT, ST)))W \quad (10)$$

where W is a learnable parameter matrix that adaptively adjusts the proportion of the two thinking networks, and LN is demonstrated to be significantly beneficial to the generalizability of the model.

Finally, the dual adaptive thinking output DT is used to learn an LSTM [16] action policy $\pi(a_t | DT_t, p)$.

4.5. Policy Learning

Following the previous works [25, 11], we treat this task as a reinforcement learning problem and utilize the asynchronous advantage actor-critic (A3C) algorithm [26]. However, in the search thinking network, the complex multi-head attention calculations are difficult to directly learn by reinforcement learning [9]; thus, we use imitation learning to pretrain the search thinking network. We divide the continuous action process into step-by-step action predictions and teach the agent to rely on only object associations to determine actions without considering historical navigation information. After pretraining, we obtain a search thinking network with a basic object association ability. Finally, the search thinking network and the navigation thinking network are jointly trained via reinforcement learning.

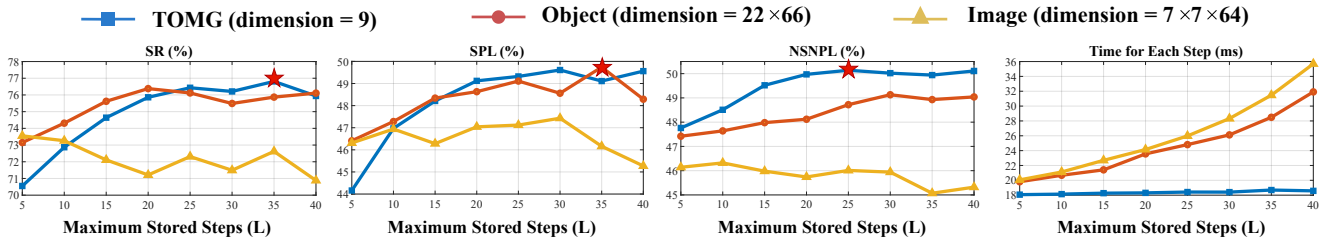


Figure 4. We compare the metrics in paths with $\mathbb{L} \geq 5$ while storing different features and path lengths for navigation thinking. The red five-pointed star indicates the choices that optimize the given indicator.

5. Experiment

5.1. Experimental Setup

Datasets AI2-Thor [18] is our main experimental platform, which 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. Additionally, we employ the RoboTHOR [6] dataset, which has 2.4 times larger area and 5.5 times longer trajectory length than AI2-Thor.

Evaluation Metrics We use the success rate (SR) and success weighted by path length (SPL) [1] metrics to evaluate the overall performance of our method. Our proposed matrics, search success rate (SSR), navigation success rate (NSR) and navigation success weighted by navigation path length (NSNPL), are used to clearly reflect the agent's ability in the "search for" phase and "navigate to" phase.

SR is formulated as $SR = \frac{1}{F}\sum_{i=1}^{F} Suc_i$, where F is the number of episodes and Suc_i indicates whether the i-th episode succeeds. SPL considers the path length more comprehensively and is defined as $SPL = \frac{1}{F}\sum_{i=1}^{F} Suc_i \frac{\mathbb{L}_i^*}{max(\mathbb{L}_i, \mathbb{L}_i^*)}$, where $\mathbb{L}_i$ is the path length taken by the agent and $\mathbb{L}_i^*$ is the theoretical shortest path.

SSR is the success rate for the "search for" phase and is formulated as $SSR = \frac{1}{F}\sum_{i=1}^{F} Nav_i$, where Nav_i indicates whether the i-th episode enters the "navigate to" phase. NSR is the success rate for the "navigate to" phase and is formulated as $NSR = \frac{1}{F_{Nav}}\sum_{i=1}^{F} Suc_i Nav_i$, where F_{Nav} is the number of episodes that enter the "navigate to" phase. NSNPL considers the navigation efficiency during the "navigate to" phase and is defined as:

$$NSNPL = \frac{1}{F_{Nav}} \sum_{i=1}^{F} Suc_i Nav_i \frac{\mathbb{L}_i^{*Nav}}{max(\mathbb{L}_i^{Nav}, \mathbb{L}_i^{*Nav})} \tag{11}$$

where $\mathbb{L}_i^{Nav}$ is the path length in the "navigate to" phase and $\mathbb{L}_i^{*Nav}$ is the theoretical shortest path length in the "navigate to" phase. During testing, we calculate $\mathbb{L}_i^{*Nav}$ in real time according to the starting position of the "navigate to" phase (the position where the agent first recognizes the target) in each task path. Intuitively, NSNPL can be conceptualized as the SPL of "navigate to" phase.

Table 2. Ablation experiments on each module in the target-aware multi-scale aggregator (TAMSA). Dynamic: dynamic aggregator kernel, TA: target-aware, MS: multi-scale, CP: circle padding.

Method			ALL (%)				
			SR↑	SPL↑	SSR↑	NSR↑	NSNPL↑
Average Pooling			79.67	45.14	97.29	81.88	47.89
Transformer			77.23	43.24	96.31	80.06	46.34
TCN			78.66	43.41	96.16	81.52	46.61
TAMSA	A1	Dynamic	80.15	44.26	97.04	82.59	47.31
	A2	A1+TA	81.20	46.71	97.17	83.56	48.93
	A3	A1+MS	81.14	47.28	96.93	83.64	49.51
	A4	A2+MS	81.32	47.41	**97.42**	83.47	49.28
	A5	A4+CP	**82.39**	**48.93**	97.25	**84.71**	**50.32**

Implementation Details We train our model with 18 workers on 2 RTX 2080Ti Nvidia GPUs. The dropout rate and target-visible filter cf in our model are set to 0.3 and 0.4, respectively. The number of implicit nodes $\mathcal{L}$ in TAMSA is set to 3. We report the results for all targets (ALL) and for a subset of targets ($\mathbb{L} \geq 5$) with optimal trajectory lengths greater than 5.

5.2. Ablation Experiments

Baseline Similar to [8, 39, 5], our baseline model adopts the features concatenated from the image branch (from ResNet18 [15]), object branch (from DETR [3]) and previous action branch as the environment perception encoding. Next, an LSTM network is used to model the temporal implicit features. The first row in Table 1 shows the performance of our baseline.

Dual Thinking As shown in Table 1, the model with search thinking outperforms the baseline with the gains of 5.44% and 3.52% in SR and SPL. The search thinking network enables the agent to quickly locate the object through object associations. Incorporating our proposed navigation thinking directly improves the NSR and NSNPL by 3.32%

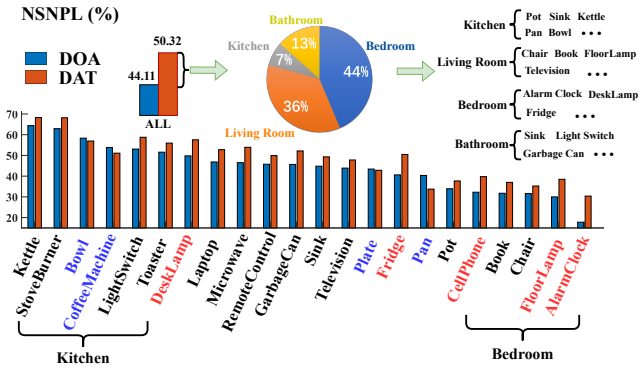


Figure 5. Compare the NSNPL of DOA[5] method and our propose DAT method target-by-target on the AI2-Thor. After using the DAT method, red target objects improve significantly and blue target objects decrease. Pie chart summarizes the contributions of all scenes. The right side of the pie chart shows common objects in each scene.

and 1.22%, demonstrating that the navigation thinking improves the agent's navigation ability after seeing the target.

Navigation Thinking Network The navigation thinking network includes three key modules: the target-oriented memory graph (TOMG), the egocentric coordinate transformation module and the target-aware multi-scale aggregator (TAMSA). Rows 4 through 7 in Table 1 show the ablation results on the three modules. The navigation thinking network without the TAMSA increases the SR by 1.34% but decreases the SPL by 1.87%. TAMSA improves the SPL back by refining the introduction of navigation thinking.

The simplified and highly abstract storage features in the TOMG facilitate the subsequent feature refinement and thinking integration. Figure 4 displays various metrics and computation speeds while using different storage features (TOMG, object and image) and maximum stored steps L. Image features perform the worst. Compared with object features, our TOMG considerably improves the NSNPL. Most importantly, the TOMG is substantially less complex than other storage methods. In terms of computational efficiency, when the number of stored steps is set to 40, compared with storing object and image features, the TOMG improves the computational speed by 41.43% and 47.69%, respectively. In terms of memory usage, the TOMG requires only 0.64% and 0.29% of the memory required by the object and image features. Furthermore, as the number of stored steps increases, the computational burden of the TOMG storage method remains essentially constant.

Target-Aware Multi-Scale Aggregator (TAMSA) Our proposed TAMSA uses a dynamic kernel to achieve automatic sequence length reduction without applying global pooling at the end. As shown in Table 2, the use of either TCNs or transformers exhibits worse performance than using average pooling directly. This finding suggests that our navigation thinking network is incompatible with these widely used encoders. Based on the initial aggregator model (A1), the target-aware (TA) property brings improvements of 0.97%, 1.62%, and the multi-scale (MS) property brings improvements of 1.05%, 2.20% in NSR and NSNPL. Furthermore, we utilize circle padding (CP) to prevent serious information loss in limited target-visible nodes, thereby optimizing the path during short-distance navigation.

Fusion of Dual Thinking Modules Our proposed adaptive fusion module (rows 8 and 9 in Table 1) effectively integrates the two separately designed thinking networks and improves the SR, SPL and SSR metrics by 1.51%, 3.22% and 1.79%. Fundamentally, adaptive fusion and layer norm decouple diverse thinking and increase the specificity of varied thinking.

5.3. Comparative Analysis of Different Targets

Figure 5 visualizes the NSNPL for different targets using the DOA [5] model and our DAT model. Obviously, the objects with the highest NSNPL belong to the kitchen scene, and the objects with the lowest NSNPL belong to the bed-

Table 3. Comparison with SOTA methods on the AI2-Thor [18] / RoboTHOR [6] datasets. ✗ indicates unacceptable resource consumption.

ID	Method	ALL (%) SR↑	SPL↑	SSR↑	NSR↑	NSNPL↑	Episode Time (s)↓
I	SSCNav [21]	77.14 / 38.12	31.09 / 14.10	89.14 / 61.37	86.54 / 62.13	51.72 / 35.14	1.342 / 4.145 ✗
	PONI [30]	78.58 / 38.42	33.78 / 16.30	89.48 / 58.46	**87.81 / 65.72**	**52.39 / 39.83**	1.591 / 4.582 ✗
II	OMT [12]	71.13 / 32.17	37.27 / 20.09	93.17 / 61.77	76.34 / 52.08	41.36 / 24.51	0.645 / 2.011
	VGM [19]	73.95 / 35.82	40.69 / 23.71	94.42 / 62.93	78.32 / 56.92	42.62 / 25.80	0.731 / 2.458
III	ORG [8]	67.32 / 30.51	37.01 / 18.62	91.07 / 59.64	73.88 / 51.15	40.24 / 20.64	0.241 / 0.769
	HOZ [39]	68.53 / 31.67	37.50 / 19.02	91.44 / 60.11	74.94 / 52.68	40.83 / 21.02	0.283 / 0.808
	VTNet [9]	72.24 / 33.92	44.57 / 23.88	94.18 / 63.29	76.62 / 53.59	46.74 / 28.26	0.321 / 1.325
	DOA [5]	74.32 / 36.22	40.27 / 22.12	95.73 / 64.18	77.63 / 56.43	44.11 / 25.88	0.334 / 1.247
IV	**Ours (DAT)**	**82.39 / 41.72**	**48.93 / 27.91**	**97.25 / 67.24**	84.71 / 62.04	50.32 / 34.27	0.352 / 1.211

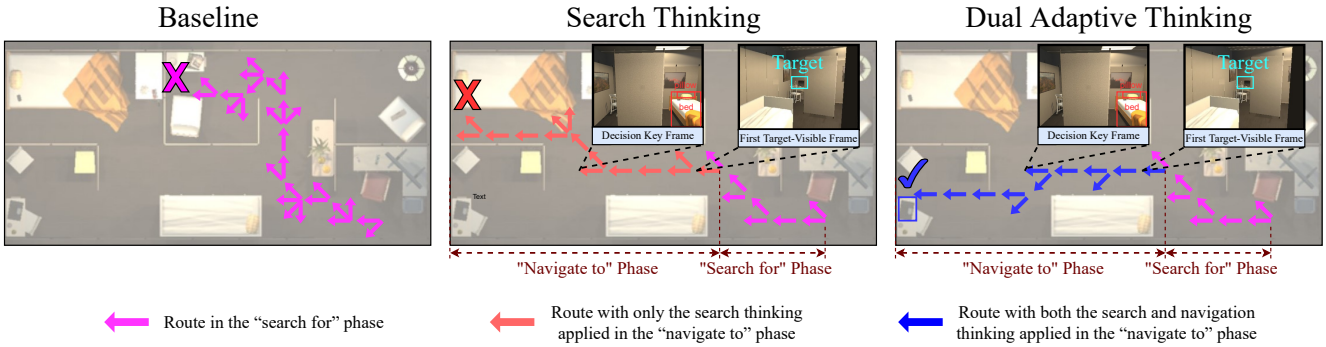


Figure 6. We show the navigation routes of three different models to complete the same task in the same environment. The baseline (ORG [8]) fails to navigate in the "search for" phase. The search thinking method (DOA [5]) and our DAT method diverge at the decision key frame in the "navigate to" phase.

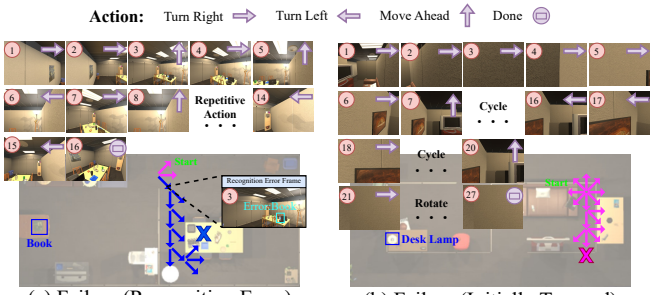
room scene. The bedroom has more complex obstacles than the kitchen, which leads to the gap in difficulty of the "navigate to" phase. The NSNPL of most objects has improved thanks to our DAT method, notably those in the intricate bedroom scene and the small, challenging-to-identify target objects. However, our approach yields a minor decline for several items, such as plates and coffee machines, in the simple kitchen scene. This may be improved by predicting scene complexity in real time during navigation.

5.4. Comparisons with the State-of-the-Art

Our DAT method is compared with three categories of relevant SOTA methods, as shown in Table 3. **(I) Modular methods based on active SLAM.** An agent with a semantic map can directly use the path planning method to quickly navigate to the target after locating it; thus, these methods obtain a higher NSNPL. Nevertheless, these methods require considerable efforts to explore the environment which makes finding targets ineffective. The SPL of the current state-of-the-art modular method PONI [30] is 15.15/11.61 lower (AI2-Thor/RoboTHOR, %) than that of our DAT method. More seriously, maintaining the semantic map at all times causes each step to consume several times as long as our method. **(II) Long-term memory methods.** These methods theoretically depend on historical information to model environments more clearly; however, methods such as OMT [12] and VGM [19] store overcomplicated features, increasing the difficulty of network learning. Therefore, the current memory modules do not exert their full strength. **(III) Search thinking methods.** These methods enhance search capabilities through object association. Compared to the best search thinking model DOA [5], our DAT model brings 8.07/5.50, 8.66/5.79 and 6.21/8.39 improvements in SR, SPL and NSNPL (AI2-Thor/RoboTHOR, %).

5.5. Qualitative Analysis

Routes of different methods are visualized in Figure 6. The baseline model is stuck in the wrong room due to its limited capability to search for targets. The search thinking model is disturbed by extraneous objects in the "navigate to" phase, leading it to choose the wrong direction at the keyframe. In contrast, our DAT method chooses the appropriate left room based on the representation of the target relative position generated by navigation thinking.



(a) Failure (Recognition Error) (b) Failure (Initially Trapped)

Figure 7. Failure cases with the first-person views. For brevity, the camera's up-and-down motion has been omitted.

6. Limitations and Failure Cases

Some potential limitations are observed in testing. (i) The model is sensitive to object detection accuracy (Figure 7(a)). How to improve the robustness to error recognition is worth exploring. (ii) The agent sometimes gets stuck in narrow and complex initial environments without historical information (Figure 7(b)). Endowing agents with the ability to escape from the deadlock in end-to-end learning may be the key to solving this problem and we leave this for future works.

7. Conclusion

In this paper, we propose the dual adaptive thinking (DAT) method, such innovation enables agents to efficiently and reliably reach the target position after locating the target. Dual thinking includes the search thinking responsible for searching the target and the navigation thinking

responsible for navigating to the target. Extensive experiments prove that dual adaptive thinking flexibly adjusts the thinking methods according to the navigation stage, thereby improving the success rate and navigation efficiency. It is worth noting that beyond the current object navigation task, multiple adaptive thinking can theoretically be applied to various time-series embodied AI tasks.

Acknowledgement

This paper is supported by the National Natural Science Foundation of China under Grants (62233013, 62073245, 62173248). Suzhou Key Industry Technological Innovation-Core Technology R&D Program (SGC2021035); Special funds for Jiangsu Science and Technology Plan (BE2022119), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0100) and the Fundamental Research Funds for the Central Universities.

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. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[2] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *CoRR*, abs/2006.13171, 2020.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

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

[5] Ronghao Dang, Zhuofan Shi, Liuyi Wang, Zongtao He, Chengju Liu, and Qijun Chen. Unbiased directed object attention graph for object navigation. *arXiv preprint arXiv:2204.04421*, 2022.

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

[7] 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. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[8] 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*, page 19–34, 2020.

[9] Heming Du, Xin Yu, and Liang Zheng. Vtnet: Visual transformer network for object goal navigation. *arXiv preprint arXiv:2105.09447*, 2021.

[10] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

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

[12] Rui Fukushima, Kei Ota, Asako Kanezaki, Yoko Sasaki, and Yusuke Yoshiyasu. Object memory transformer for object goal navigation. *arXiv preprint arXiv:2203.14708*, 2022.

[13] Samir Yitzhak Gadre, Kiana Ehsani, Shuran Song, and Roozbeh Mottaghi. Continuous scene representations for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14849–14859, 2022.

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

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

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Vidhi Jain, Prakhar Agarwal, Shishir Patil, and Katia Sycara. Learning embeddings that capture spatial semantics for indoor navigation. *arXiv preprint arXiv:2108.00159*, 2021.

[18] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[19] Obin Kwon, Nuri Kim, Yunho Choi, Hwiyeon Yoo, Jeongho Park, and Songhwai Oh. Visual graph memory with unsupervised representation for visual navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15890–15899, 2021.

[20] Xinghang Li, Di Guo, Huaping Liu, and Fuchun Sun. Revece: Remote embodied visual referring expression in continuous environment. *IEEE Robotics and Automation Letters*, 7(2):1494–1501, 2022.

[21] Yiqing Liang, Boyuan Chen, and Shuran Song. Sscnav: Confidence-aware semantic scene completion for visual semantic navigation. In *2021 IEEE International Conference*

on Robotics and Automation (ICRA), pages 13194–13200. IEEE, 2021.

[22] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[23] Xinzhu Liu, Di Guo, Huaping Liu, and Fuchun Sun. Multi-agent embodied visual semantic navigation with scene prior knowledge. *IEEE Robotics and Automation Letters*, 7(2):3154–3161, 2022.

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[25] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. In *5th International Conference on Learning Representations*, 2017.

[26] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[27] Mahdi Kazemi Moghaddam, Ehsan Abbasnejad, Qi Wu, Javen Qinfeng Shi, and Anton Van Den Hengel. Foresi: Success-aware visual navigation agent. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 691–700, 2022.

[28] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[29] Yiding Qiu, Anwesan Pal, and Henrik I Christensen. Target driven visual navigation exploiting object relationships. *arXiv preprint arXiv:2003.06749*, 2(7), 2020.

[30] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18890–18900, 2022.

[31] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[32] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1746–1754, 2017.

[33] Liuyi Wang, Zongtao He, Ronghao Dang, Huiyi Chen, Chengju Liu, and Qijun Chen. Res-sts: Referring expression speaker via self-training with scorer for goal-oriented vision-language navigation. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2022.

[34] Wei Wang, Yujing Yang, Xin Wang, Weizheng Wang, and Ji Li. Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering*, 58(4):040901, 2019.

[35] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6750–6759, 2019.

[36] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[37] 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.

[38] Haokui Zhang, Wenze Hu, and Xiaoyu Wang. Edgeformer: Improving light-weight convnets by learning from vision transformers. *arXiv preprint arXiv:2203.03952*, 2022.

[39] 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*, pages 15130–15140, 2021.

[40] Qianfan Zhao, Lu Zhang, Bin He, Hong Qiao, and Zhiyong Liu. Zero-shot object goal visual navigation. *arXiv preprint arXiv:2206.07423*, 2022.

[41] Chen Zhu, Michael Meurer, and Christoph Günther. Integrity of visual navigation—developments, challenges, and prospects. *NAVIGATION: Journal of the Institute of Navigation*, 69(2), 2022.

[42] Fengda Zhu, Xiwen Liang, Yi Zhu, Qizhi Yu, Xiaojun Chang, and Xiaodan Liang. Soon: scenario oriented object navigation with graph-based exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12689–12699, 2021.