

000
001
002003

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

004
005
006
007
008
009
010
011

Anonymous ICCV submission

012

Abstract

013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053

“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.

040

1. Introduction

041
042
043
044
045
046
047
048
049
050
051
052
053

Object navigation [29, 22, 25, 42] is a challenging task that requires an agent to find a target object in an unknown environment with first-person visual observations. Some researchers [31, 15, 19] recently introduced scene prior knowledge into end-to-end navigation networks. These methods have been applied to address various issues, including object associations [39], object attention bias [6], and the lack of universal knowledge [16]. 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 (Table 6 in Appendix B) show that for the current SOTA end-to-end methods, the “navigate to” steps account

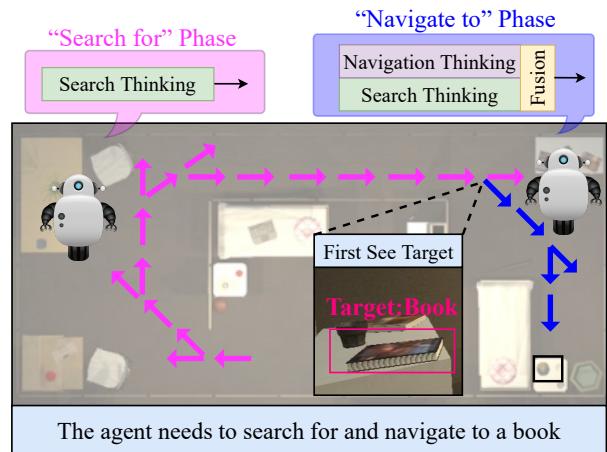
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107

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.

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 [5, 32] model the environment by using top-down semantic maps [30, 34]. 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 [6] 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 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 [20] and RoboTHOR [8] 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 [6] 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 [3, 37, 35] 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. [41] proposed the hierarchical object-to-zone (HOZ) graph to guide an agent in a coarse-to-fine manner. Moreover, Dang et al. [6] 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 [5, 32] 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 [5] is feasible on the Habitat dataset. The PONI [32] 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 [12] is a challenging research topic that requires agents to use well-developed intuitive tasks (e.g., classification [36] and detection [24]) to complete complex logical tasks (e.g., navigation [43] and interaction [33]) 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.

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

162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

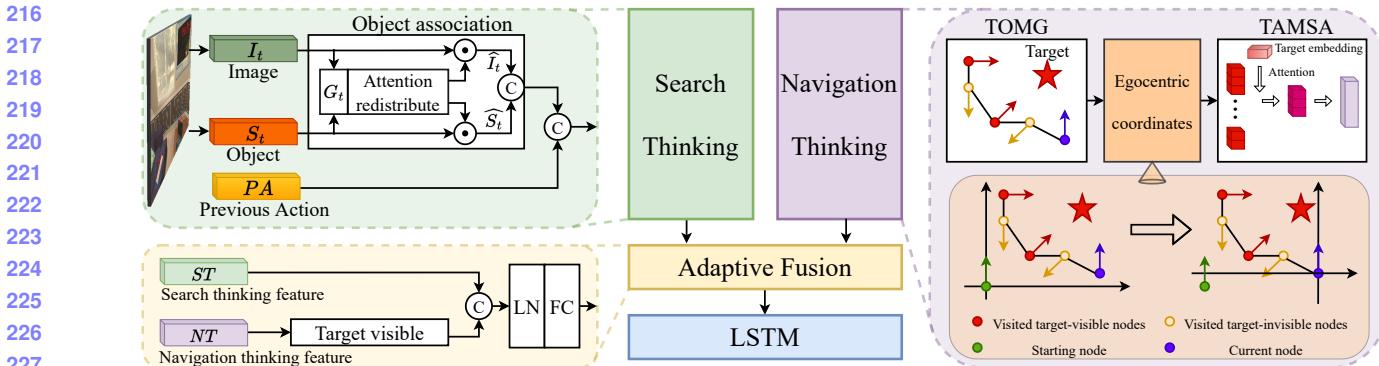


Figure 2. Model overview. TOMG: target-oriented memory graph. TAMSA: target-aware multi-scale aggregator. Our model includes three modules: search thinking, navigation thinking and adaptive fusion. In the search thinking network, we endow the model with an object association ability according to the directed object attention (DOA) graph method proposed in [6]. 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.

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 $a_t \in A = \{\text{MoveAhead}; \text{RotateLeft}; \text{RotateRight}; \text{LookDown}; \text{LookUp}; \text{Done}\}$ 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 [10]) 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 [6]. As shown in the green box in Figure 2, accord-

ing 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 [4]) and image features I_t (from ResNet18 [17]) 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 \hat{S}_t^q :

$$\hat{S}_t^q = S_t^q G_t^q \quad q = 1, 2, \dots, N \quad (1)$$

where $\hat{S}_t = \{\hat{S}_t^1, \hat{S}_t^2, \dots, \hat{S}_t^N\}$, 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 \hat{I}_t :

$$Q_i = DW_i^Q \quad K_i = I_t W_i^K \quad V_i = I_t W_i^V \quad i = 1, \dots, NH \quad (2)$$

$$\text{head}_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{H}D}\right) V_i \quad (3)$$

$$\hat{I}_t = \text{Concat}(\text{head}_1, \dots, \text{head}_{NH}) W^O \quad (4)$$

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

Finally, the attention-aware object features \hat{S}_t and image features \hat{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 abil-

324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377

ity 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 [14, 44] which store both the images $m_i \in \mathbb{R}^{7 \times 7 \times 512}$ and objects $m_o \in \mathbb{R}^{N \times 256}$ features contain too much redundant noise. In contrast, our TOMG node only stores high-level features $m \in \mathbb{R}^{1 \times 9}$ 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 [14, 44].

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):

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

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 $\tilde{M} \in \mathbb{R}^{L \times 11}$.

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 \tilde{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.

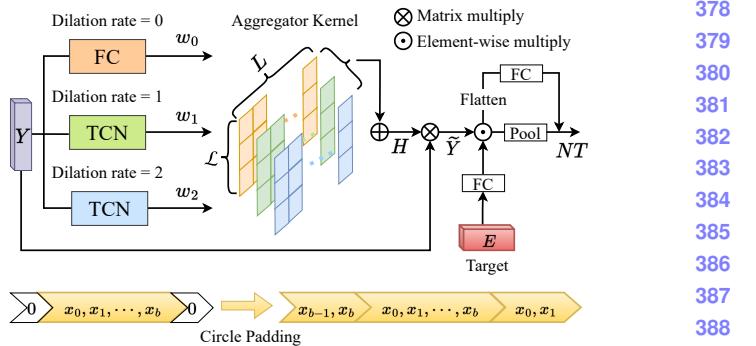


Figure 3. A detailed explanation of the target-aware multi-scale aggregator (TAMSA). We first use the multi-scale TCNs to obtain aggregator kernels that aggregate the target-oriented memory graph (TOMG) with L nodes into graph with \mathcal{L} nodes. Then, the aggregated features allocate attention to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

First, to improve the feature expression ability of the navigation thinking network, we use fully connected (FC) layers to map the features \tilde{M} to higher dimensional spaces. Inspired by some advanced works [9, 26] on vision transformers, we add layer normalization between the two FC layers to stabilize the forward input distribution and back-propagation gradient [38]. The encoding details can be formulated as follows:

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

where δ denotes the ReLU function, LN denotes layer normalization, and $W^{M_1} \in \mathbb{R}^{11 \times 16}$ and $W^{M_2} \in \mathbb{R}^{16 \times 32}$ 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, inspired by [40], 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 :

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

where $H = \{H(1), \dots, H(L)\}$, 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

432 Table 1. Ablation results on each module in the three sub-networks: search, navigate and fusion. 486

433 ID	Search Thinking			Navigation Thinking			Fusion		ALL (%)				Episode Time (s)↓
	Associate	Thinking Pretrain	TOMG	Egocentric	TAMSA	AF	LN	SR↑	SPL↑	SSR↑	NSR↑	NSNPL↑	
435 1								71.34	40.36	92.41	76.19	43.74	0.258
436 2	✓							74.43	40.93	95.82	77.67	44.11	0.334
437 3	✓	✓						76.78	43.88	94.19	81.40	47.09	0.334
438 4	✓	✓	✓					76.02	43.15	93.41	80.21	47.12	0.336
439 5	✓	✓	✓	✓				78.12	42.01	95.12	82.13	45.80	0.336
440 6	✓	✓	✓	✓		✓		78.04	45.67	94.83	82.29	48.44	0.345
441 7	✓	✓	✓	✓	✓	✓		80.88	45.71	95.46	84.72	48.31	0.346
442 8	✓	✓	✓	✓	✓	✓	✓	81.34	47.53	96.38	84.39	49.74	0.350
443 9	✓	✓	✓	✓	✓	✓	✓	82.39	48.93	97.25	84.71	50.32	0.352

444 f_d and b_d denote the weights and biases in the convolution
445 kernel with dilation rate d . The multi-scale dynamic kernel
446 $H \in \mathbb{R}^{L \times \mathcal{L}}$ refines $Y \in \mathbb{R}^{L \times 32}$ to $\tilde{Y} \in \mathbb{R}^{\mathcal{L} \times 32}$.

447 Intuitively, the mappings between the observation data
448 and target azimuth differ when searching for different tar-
449 gets. For example, when looking for a TV, even if the TV
450 is located far from the agent, the agent can clearly identify
451 the target and obtain a larger target bounding box; however,
452 when looking for a mobile phone, the agent can only obtain
453 a smaller target bounding box, even if the agent is close to
454 the mobile phone. Therefore, we enhance the TAMSA repre-
455 sentation by considering the target semantic information.
456 To achieve this goal, the one-hot target index E is encoded
457 to the same channel dimension as \tilde{Y} through two FC layers,
458 whose result is channel-wise multiplied with \tilde{Y} to get the
459 target-aware feature representation \hat{Y} :

$$460 \hat{Y} = H^T Y \odot \delta(\delta(EW^{E_1})W^{E_2}) \quad (8)$$

461 Finally, to obtain the final output NT of the navigation
462 thinking network, we flatten \hat{Y} from $\mathbb{R}^{\mathcal{L} \times 32}$ to $\mathbb{R}^{1 \times 32\mathcal{L}}$ and
463 use an FC layer to reduce the output dimension. Further-
464 more, we add residual connections to ensure the stability of
465 the feature transfer process.

$$466 NT = \delta(Flatten(\hat{Y})W^Y) + \frac{1}{\mathcal{L}} \sum_{l=1}^{\mathcal{L}} \hat{Y}(l) \quad (9)$$

467 A dropout layer is added before the output to reduce over-
468 fitting in the navigation thinking network.

475 4.4. Adaptive Fusion (AF) of Dual Thinking Net- 476 works

477 Search thinking and navigation thinking have different
478 work strategies according to the navigation progress. Dur-
479 ing the “search for” phase, since there are no visited target-
480 visible nodes, NT is an all-zero matrix. Therefore, the nav-
481 igation thinking network does not affect the action decision
482 when the target has not yet been seen. During the “navigate
483 to” phase, to ensure navigation robustness, search thinking
484 and navigation thinking work together to guide the action

485 decision. As the number of visited target-visible nodes in-
486 creases, navigation thinking gradually dominates. The fu-
487 sion process of the two thinking methods can be expressed
488 as:

$$489 DT = (LN(Concat(NT, ST)))W \quad (10)$$

490 where W is a learnable parameter matrix that adaptively ad-
491 justs the proportion of the two thinking networks, and LN is
492 demonstrated to be significantly beneficial to the generaliz-
493 ability of the model.

494 Finally, the dual adaptive thinking output DT is used to
495 learn an LSTM [18] action policy $\pi(a_t|DT_t, p)$.

496 4.5. Policy Learning

497 Following the previous works [27, 13], we treat this
498 task as a reinforcement learning problem and utilize the
499 asynchronous advantage actor-critic (A3C) algorithm [28].
500 However, in the search thinking network, the complex
501 multi-head attention calculations are difficult to directly
502 learn by reinforcement learning [11]; thus, we use imita-
503 tion learning to pretrain the search thinking network. We
504 divide the continuous action process into step-by-step ac-
505 tion predictions and teach the agent to rely on only object
506 associations to determine actions without considering his-
507 torical navigation information. After pretraining, we obtain
508 a search thinking network with a basic object association
509 ability. Finally, the search thinking network and the naviga-
510 tion thinking network are jointly trained via reinforcement
511 learning.

5. Experiment

511 5.1. Experimental Setup

512 **Datasets** AI2-Thor [20] is our main experimental plat-
513 form, which includes 30 different floorplans for each of 4
514 room layouts: kitchen, living room, bedroom, and bath-
515 room. For each scene type, we use 20 rooms for training,
516 5 rooms for validation, and 5 rooms for testing. Addition-
517 ally, we employ the RoboTHOR [8] dataset, which has 2.4
518 times larger area and 5.5 times longer trajectory length than
519 AI2-Thor.

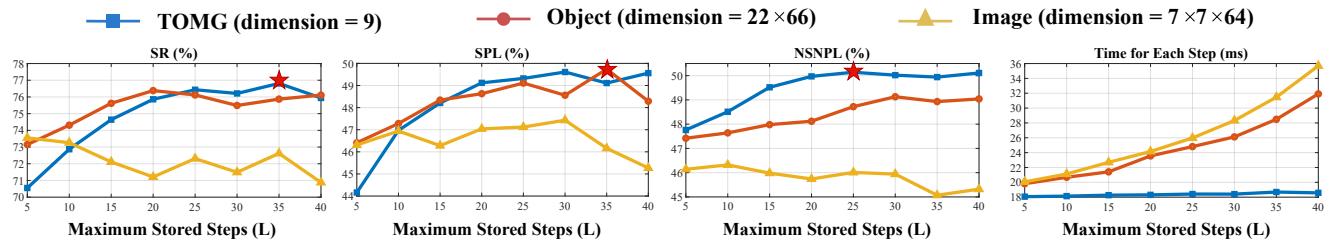


Figure 4. We compare the metrics in paths with $\mathbb{L} \geq 5$ while storing different features and path lengths for navigation thinking. The red five-pointed star indicates the choices that optimize the given indicator.

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 metrics, 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:

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

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.

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. More network details are given in Appendix A.

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

5.2. Ablation Experiments

Baseline Similar to [10, 41, 6], our baseline model adopts the features concatenated from the image branch (from ResNet18 [17]), object branch (from DETR [4]) 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% 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.

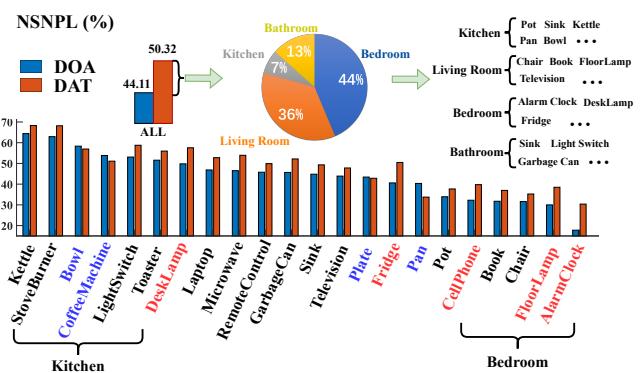


Figure 5. Compare the NSNPL of DOA[6] method and our propose DAT method target-by-target on the AI2-Thor. After using the DAT method, red target objects improve significantly and blue target objects decrease. Pie chart summarizes the contributions of all scenes. The right side of the pie chart shows common objects in each scene.

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 auto-

matic 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 [6] 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 bedroom 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. More results and analysis can be found in Appendix E and F.

Table 3. Comparison with SOTA methods on the AI2-Thor [20] / RoboTHOR [8] datasets. \times indicates unacceptable resource consumption.

ID	Method	ALL (%)					Episode Time (s)↓
		SR↑	SPL↑	SSR↑	NSR↑	NSNPL↑	
I	SSCNav [23]	77.14 / 38.12	31.09 / 14.10	89.14 / 61.37	86.54 / 62.13	51.72 / 35.14	1.342 / 4.145 \times
	PONI [32]	78.58 / 38.42	33.78 / 16.30	89.48 / 58.46	87.81 / 65.72	52.39 / 39.83	1.591 / 4.582 \times
II	OMT [14]	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 [21]	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 [10]	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 [41]	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 [11]	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 [6]	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

756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772

Baseline



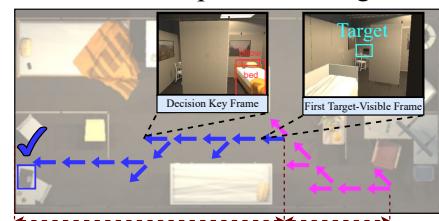
← Route in the “search for” phase

Search Thinking



← Route with only the search thinking applied in the “navigate to” phase

Dual Adaptive Thinking



← Route with both the search and navigation thinking applied in the “navigate to” phase

Figure 6. We show the navigation routes of three different models to complete the same task in the same environment. The baseline (ORG [10]) fails to navigate in the “search for” phase. The search thinking method (DOA [6]) and our DAT method diverge at the decision key frame in the “navigate to” phase. The navigation routes in more scenarios are shown in Appendix G.

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 [32] 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 [14] and VGM [21] 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 [6], 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, %). More experiments are in Appendix D.

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.

810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851

Action: Turn Right → Turn Left ← Move Ahead ↑ Done ⏺

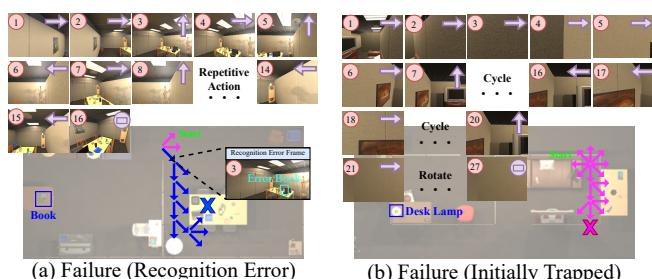


Figure 7. Failure cases with the first-person views. For brevity, the camera’s up-and-down motion has been omitted.

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.

864

References

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

- [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. 6
- [2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018. 15
- [3] 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. 2
- [4] 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. 3, 6, 11
- [5] 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. 1, 2
- [6] 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. 1, 2, 3, 6, 7, 8, 11, 13, 14
- [7] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2018. 15
- [8] 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. 2, 5, 7, 14
- [9] 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. 4
- [10] 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. 3, 6, 7, 8, 14

- [11] Heming Du, Xin Yu, and Liang Zheng. Vtnet: Visual transformer network for object goal navigation. *arXiv preprint arXiv:2105.09447*, 2021. 5, 7, 14
- [12] 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. 2
- [13] 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. 5
- [14] Rui Fukushima, Kei Ota, Asako Kanezaki, Yoko Sasaki, and Yusuke Yoshiyasu. Object memory transformer for object goal navigation. *arXiv preprint arXiv:2203.14708*, 2022. 4, 7, 8, 14
- [15] 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. 1
- [16] 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. 1
- [17] 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. 3, 6, 11
- [18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 5
- [19] Vidhi Jain, Prakhar Agarwal, Shishir Patil, and Katia Sycara. Learning embeddings that capture spatial semantics for indoor navigation. *arXiv preprint arXiv:2108.00159*, 2021. 1
- [20] 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. 2, 5, 7, 14
- [21] 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. 7, 8
- [22] 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. 1
- [23] 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. 7, 14
- [24] 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. 2
- [25] Xinzhu Liu, Di Guo, Huaping Liu, and Fuchun Sun. Multi-agent embodied visual semantic navigation with scene

- 972 prior knowledge. *IEEE Robotics and Automation Letters*,
973 7(2):3154–3161, 2022. 1
- 974 [26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng
975 Zhang, Stephen Lin, and Baining Guo. Swin transformer:
976 Hierarchical vision transformer using shifted windows. In
977 *Proceedings of the IEEE/CVF International Conference on*
978 *Computer Vision*, pages 10012–10022, 2021. 4
- 979 [27] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer,
980 Andrew J Ballard, Andrea Banino, Misha Denil, Ross
981 Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning
982 to navigate in complex environments. In *5th International*
983 *Conference on Learning Representations*, 2017. 5
- 984 [28] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza,
985 Alex Graves, Timothy Lillicrap, Tim Harley, David Silver,
986 and Koray Kavukcuoglu. Asynchronous methods for deep
987 reinforcement learning. In *International conference on machine*
988 *learning*, pages 1928–1937. PMLR, 2016. 5
- 989 [29] Mahdi Kazemi Moghaddam, Ehsan Abbasnejad, Qi Wu,
990 Javen Qinfeng Shi, and Anton Van Den Hengel. Foresi:
991 Success-aware visual navigation agent. In *Proceedings of the*
992 *IEEE/CVF Winter Conference on Applications of Computer*
993 *Vision*, pages 691–700, 2022. 1
- 994 [30] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor
995 Darrell, and Alexei A Efros. Context encoders: Feature
996 learning by inpainting. In *Proceedings of the IEEE conference*
997 *on computer vision and pattern recognition*, pages
998 2536–2544, 2016. 1
- 999 [31] Yiding Qiu, Anwesan Pal, and Henrik I Christensen. Target
1000 driven visual navigation exploiting object relationships.
arXiv preprint arXiv:2003.06749, 2(7), 2020. 1
- 1001 [32] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot,
1002 Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Poni:
1003 Potential functions for objectgoal navigation with
1004 interaction-free learning. In *Proceedings of the IEEE/CVF*
1005 *Conference on Computer Vision and Pattern Recognition*,
1006 pages 18890–18900, 2022. 1, 2, 7, 8, 14
- 1007 [33] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan
1008 Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer,
1009 and Dieter Fox. Alfred: A benchmark for interpreting
1010 grounded instructions for everyday tasks. In *Proceedings of*
1011 *the IEEE/CVF conference on computer vision and pattern*
1012 *recognition*, pages 10740–10749, 2020. 2
- 1013 [34] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis
1014 Savva, and Thomas Funkhouser. Semantic scene completion
1015 from a single depth image. In *Proceedings of the IEEE*
1016 *conference on computer vision and pattern recognition*,
1017 pages 1746–1754, 2017. 1
- 1018 [35] Liuyi Wang, Zongtao He, Ronghao Dang, Huiyi Chen,
1019 Chengju Liu, and Qijun Chen. Res-sts: Referring expression
1020 speaker via self-training with scorer for goal-oriented
1021 vision-language navigation. *IEEE Transactions on Circuits*
1022 *and Systems for Video Technology*, pages 1–1, 2022. 2
- 1023 [36] Wei Wang, Yujing Yang, Xin Wang, Weizheng Wang, and
1024 Ji Li. Development of convolutional neural network and its
1025 application in image classification: a survey. *Optical Engineering*,
1026 58(4):040901, 2019. 2
- 1027 [37] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari,
1028 Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to
1029 learn: Self-adaptive visual navigation using meta-learning.
1030 In *Proceedings of the IEEE/CVF Conference on Computer*
1031 *Vision and Pattern Recognition*, pages 6750–6759, 2019. 2,
1032 14
- 1033 [38] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and
1034 Junyang Lin. Understanding and improving layer normalization.
1035 *Advances in Neural Information Processing Systems*,
1036 32, 2019. 4
- 1037 [39] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and
1038 Roozbeh Mottaghi. Visual semantic navigation using scene
1039 priors. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9,*
1040 2019, 2019. 1, 14
- 1041 [40] Haokui Zhang, Wenze Hu, and Xiaoyu Wang. Edgeformer:
1042 Improving light-weight convnets by learning from vision
1043 transformers. *arXiv preprint arXiv:2203.03952*, 2022. 4
- 1044 [41] Sixian Zhang, Xinhang Song, Yubing Bai, Weijie Li, Yakui
1045 Chu, and Shuqiang Jiang. Hierarchical object-to-zone graph
1046 for object navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15130–
1047 15140, 2021. 2, 6, 7, 14
- 1048 [42] Qianfan Zhao, Lu Zhang, Bin He, Hong Qiao, and Zhiyong
1049 Liu. Zero-shot object goal visual navigation. *arXiv preprint*
1050 *arXiv:2206.07423*, 2022. 1
- 1051 [43] Chen Zhu, Michael Meurer, and Christoph Günther. Integrity
1052 of visual navigation—developments, challenges, and
1053 prospects. *NAVIGATION: Journal of the Institute of Navigation*, 69(2), 2022. 2
- 1054 [44] Fengda Zhu, Xiwen Liang, Yi Zhu, Qizhi Yu, Xiaojun
1055 Chang, and Xiaodan Liang. Soon: scenario oriented object
1056 navigation with graph-based exploration. In *Proceedings of*
1057 *the IEEE/CVF Conference on Computer Vision and Pattern*
1058 *Recognition*, pages 12689–12699, 2021. 4
- 1059
- 1060
- 1061
- 1062
- 1063
- 1064
- 1065
- 1066
- 1067
- 1068
- 1069
- 1070
- 1071
- 1072
- 1073
- 1074
- 1075
- 1076
- 1077
- 1078
- 1079

1080 Table 4. The details of dual adaptive thinking network for object navigation. 1134
1081

Module	Index	Input	Operation	Output
Search Thinking	(1)	S_t	FC+ReLU	(22×49)
	(2)	(1)	Attention Distribution	(22×49)
	(3)	(2)	Reshape+0.3Dropout	$(22 \times 7 \times 7)$
	(4)	$D(Q) I_t(K, V)$	Multi-head Attention ($HD = 512, NH = 8$)	$(64 \times 7 \times 7)$
	(5)	PA	FC+ReLU	(1×10)
	(6)	(5)	Repeat	$(10 \times 7 \times 7)$
	(7)	(3,4,6)	Concatenation	$(96 \times 7 \times 7)$
Navigation Thinking	(8)	TOMG	Coordinate Ego-centric	(27×11)
	(9)	(8)	FC+ReLU	(27×16)
	(10)	(9)	TAMSA ($\ell = 3$)	(1×32)
	(11)	(10)	Repeat+0.3Dropout	$(32 \times 7 \times 7)$
Adaptive Fusion	(12)	(7,11)	Concatenation+LayerNorm	$(128 \times 7 \times 7)$
	(13)	(12)	FC+ReLU	$(64 \times 7 \times 7)$
	(14)	(13)	0.3Dropout+Flatten	3136
Policy Learning	(15)	(14)	Two Layers LSTM	512
	(16)	(15)	Actor FC	6
	(17)	(15)	Critic FC	1

1101 Table 5. Hyperparameters during training. 1154
1102

Reward	penalize each step	-0.01
	encourage move ahead	+0.01
	succeed	+5.00
A3C	discount factor	0.99
	lambda parameter for GAE	1.00
	entropy term coefficient	0.01
	value loss coefficient	0.50
Learning Rate	pretrain	0.0005
	reinforcement learning	0.0001

A. More Network Details

A.1. Simulated Environment

We discretize the scene into a 0.25m grid of navigable points. To speed up reinforcement learning training, we pre-compute the environmental data from the simulator for each point offline. Therefore, the agent can only move between these points. The step size is 0.25m, the yaw rotation is 45 degrees, and the pitch rotation is 30 degrees.

A.2. Model Architecture

The model architecture and key details are presented in formulas and images in the main paper. Table 4 gives more details on the input, operation, hyperparameters and feature dimensions of each layer.

The search thinking module has three branches: objects (1-3), images (4), past actions (5-6). The object branch input $S_t \in \mathbb{R}^{22 \times 262}$ is obtained by DETR [4], which encodes object appearance (class labels and bounding boxes) as well as instance relations and global contexts. The image branch

input $S_t \in \mathbb{R}^{512 \times 7 \times 7}$ uses ResNet18 [17] features, which preserve spatial relationships better than deeper features. To ensure consistent dimensions for concatenation, object features and past action features are reshaped to match image features (7×7).

The navigation thinking module uses TOMG as input, which stores only target-related information. If an object with target confidence $cf \geq 0.4$ is visible, its related information is stored in TOMG.

A.3. Training Hyperparameters

The hyperparameters for training our DAT model are given in Table 5. We use the same reward setting as DOA [6], which has three components: (i) A small negative reward of -0.01 for each step. (ii) A positive reward of 0.01 for moving ahead. (iii) A large positive reward of 5.0 for reaching any instance of the target object category within a certain number of steps.

We set the discount factor to 0.99 to balance long-term and short-term rewards. The object navigation task is complex and hard to converge, so we use n-step bootstrapping advantage estimation by setting lambda to 1 in GAE. The entropy term coefficient and value loss coefficient in A3C are 0.01 and 0.5 respectively.

We pre-train our model with supervised learning using a learning rate of 0.0005. Then we train it with reinforcement learning using a learning rate of 0.0001.

B. Disadvantages of Single Thinking

In the introduction, we explain the problems with single thinking in the “navigate to” phase. In this section, quantitative experimental results (Table 6) are used to explain

1188 Table 6. The performance of methods with different thinking levels in the processes of “navigate to” and “search for”. 1242

Method	Path Length			Success Rate			Rotation Action Rate	
	ALL	Search	Navigate	ALL	Search	Navigate	Search	Navigate
Single Thinking	19.92 \pm 0.31	7.42 \pm 0.14	12.50 \pm 0.21	76.87% \pm 0.52%	94.80% \pm 0.22%	81.09% \pm 1.03%	66.33% \pm 0.75%	60.53% \pm 0.92%
Dual Thinking	18.26 \pm 0.20	7.96 \pm 0.11	10.30 \pm 0.18	82.39% \pm 1.21%	97.25% \pm 0.34%	84.71% \pm 0.82%	71.62% \pm 0.79%	49.04% \pm 0.66%
Human	13.15 \pm 0.27	7.13 \pm 0.20	6.02 \pm 0.15	97.14% \pm 0.84%	97.22% \pm 0.79%	99.92% \pm 0.03%	78.88% \pm 0.42%	31.45% \pm 0.34%

in detail the navigation behavior gap between single thinking and human thinking. Moreover, how our dual adaptive thinking (DAT) approach bridges this gap is presented.

B.1. Acquisition of Data

In Table 6, we display the metrics for three different thinking. **The single thinking** model removes the navigation thinking module from our overall model, leaving only the search thinking module. **The dual thinking** model is our complete DAT model. **Human thinking** is based on real people tests. We invited 10 subjects to independently complete all tasks that the agent must complete in all test environments. To acquire the final human indicators, the indicators of these 10 subjects are averaged after removing the highest value and the lowest value.

We analyze three main categories of indicators: path length, success rate and rotation action rate.

- 1) The path length is the average number of steps taken by the agent to complete the test tasks. The overall path consists of two phases: “search for” phase and “navigate to” phase; the first target-visible frame is the dividing point. Therefore, the overall path length is equal to the path length of the “search for” phase plus the path length of the “navigate to” phase.
- 2) The success rate is the ratio of the agent successfully reaching the target position and outputting Done. In addition to the overall success rate, we also counted the search success rate and the navigate success rate. In particular, if the agent recognizes the target object, then the “search for” phase is successful.

3) The rotation action rate is the ratio of actions that do not change the agent’s position. We counted the agent’s rotation action rate in the “search for” phase and the “navigate to” phase.

B.2. Comparative Analysis of Different Thinking

In terms of the path length, the human result is 6.77 steps shorter than the single thinking method, which is attributed mainly to the gap in the “navigate to” phase. By introducing dual thinking, our DAT method significantly optimizes the route in the “navigate to” phase, and the navigation efficiency in the “search for” phase is maintained.

Regarding the success rate, the overall success rate is affected by the combination of the search success rate and the navigation success rate. The navigation success rate is increased since the dual thinking network can memorize and generate the target’s approximate orientation thanks to our navigation thinking. Introducing navigation thinking has also improved the search success rate because navigation thinking shares the navigation pressure for search thinking so that search thinking can focus more on searching targets. Finally, end-to-end networks are determined to perform comparably to humans in terms of the search success rate, but there is still a significant difference in the navigation success rate. Therefore, how to improve the navigation success rate is still the main problem to be solved in the future.

Concerning the rotation and straight actions, as the thinking complexity increases, rotation actions are gradually concentrated in the “search for” phase. This trend is consistent with the different demand characteristics between

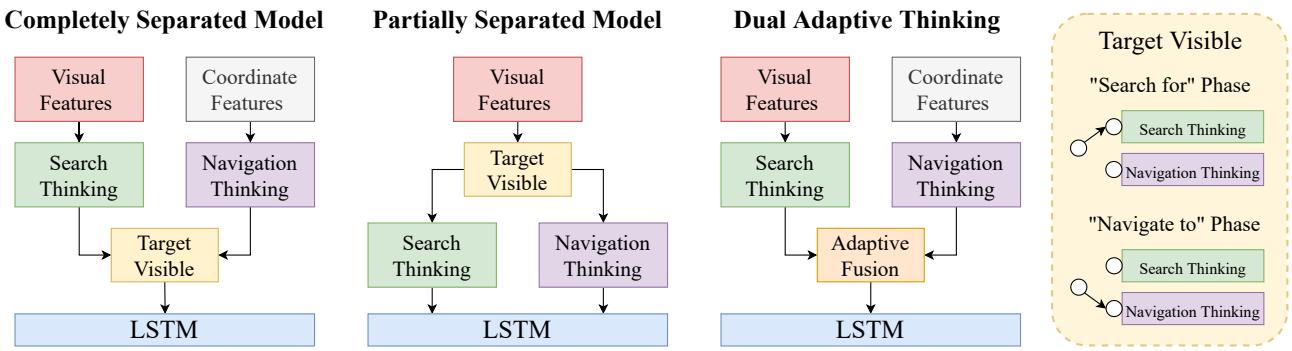


Figure 8. Different model structures for implementing dual thinking.

1296 Table 7. Comparison with different model structures for implementing dual thinking.
1297

Method	ALL (%)					$\mathbb{L} \geq 5$ (%)					Episode Length \downarrow
	SR \uparrow	SPL \uparrow	SSR \uparrow	NSR \uparrow	NSNPL \uparrow	SR \uparrow	SPL \uparrow	SSR \uparrow	NSR \uparrow	NSNPL \uparrow	
Partially Separated	77.49	46.61	95.68	80.99	48.26	69.48	45.62	94.97	73.66	48.14	27.19
Completely Separated	72.09	40.16	94.53	76.26	40.21	63.14	38.57	93.55	67.49	39.83	23.65
DAT (Ours)	82.39	48.93	97.25	84.71	50.32	76.21	49.32	96.22	79.20	50.14	19.87

the “search for” phase and the “navigate to” phase. In the “search for” phase, the agent needs to efficiently obtain environmental information and capture the target through abundant rotation actions. In the “navigate to” phase, the agent already knows the position of the target and needs to move forward to approach the target. The clear division of labor in each stage ensures efficient navigation.

C. Different Dual Thinking Structures

The essence of dual thinking is to use different decision networks in different navigation stages, which can be accomplished by using various model structures. As shown in Figure 8, in addition to our dual adaptive thinking (DAT) structure, we also experiment with completely and partially separated model structures.

C.1. Completely Separated Structure

The completely separated structure entirely decouples the thinking used in different stages. The difference with respect to our DAT method is that in the “navigate to” phase, the agent does not engage in search thinking and relies only on navigation thinking to make decisions. As Table 7 shows, the NSR of the completely separated structure is very low. There are two reasons for this model’s failure: (1) At the beginning of the “navigate to” phase, the target information in the target-oriented memory graph (TOMG) is not sufficient to generate a complete target orientation, so it cannot independently support action decisions. (2) If no search thinking is performed, visual information will be lost, resulting in a loss of obstacle avoidance ability. Therefore, in our DAT model, search thinking is active at all times.

C.2. Partially Separated Structure

In the partially separated structure, search thinking and navigation thinking take the same input features but use different encoding networks to model the different types of thinking. The difference with respect to our DAT method is that instead of using our target-oriented coordinate features, navigation thinking relies on the same visual features as search thinking. As seen in Table 7, the performance of the partially separated structure is not ideal, especially the episode length is too long. The comparison results suggest that the presence of too much redundant information in the visual features makes it too difficult for the navigation thinking network to learn the relative position of the

target. More seriously, unreasonable navigation thinking affects the learning of the backbone visual embedding network, causing the whole model to collapse.

C.3. DAT Structure

Our DAT method uses specific input features and a reasonable adaptive fusion method for dual thinking to ensure stable and excellent network performance in different navigation stages.

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

We simply compare the aggregate metrics of all episodes due to space restrictions in the main text. However, the performance differences between various approaches cannot be completely displayed because some objects in the test set are quite close to the agent. Therefore, in Table 8 and Table 9, we supplemented the results of episodes with path lengths larger than 5 ($\mathbb{L} \geq 5$) in the two datasets. In end-to-end methods (II, III, IV), our method outperforms the SOTA method (DOA [6]) by 8.33 / 7.00, 8.96 / 4.14 in SR and SPL ($\mathbb{L} \geq 5$, AI2-Thor / RoboTHOR, %). Additionally, our approach outperforms previous end-to-end approaches during both the “search for” and the “navigate to” phases.

It is worth noting that there is no detailed navigation capability comparison between end-to-end methods (II, III, IV) and modular methods (I) in previous works. Two modular approaches (SSCNav and PONI) are transferred from the Habitat dataset to the AI2-Thor and RoboTHOR datasets in our research. Compared with the end-to-end methods, the module approaches offer clear advantages in the “navigate to” phase because of the strong interpretability of the semantic map. However, the heavy reliance on semantic mapping also results in high computing costs and inefficiencies during the “search for” phase. Therefore, modular approaches have great potential in scenarios where semantic maps can be reused repeatedly, but end-to-end approaches are still preferred in completely unknown environments.

E. Target Level Indicators

Our method has obvious advantages over other methods in terms of overall metrics. However, due to variations in navigational complexity and path characteristics between various target objects, we carried out a comparative experiment of each single target object. The findings not only deepen our understanding of the DAT method’s benefits but

1404 Table 8. Comparison with SOTA methods in AI2-Thor [20]. \times indicates unacceptable resource consumption.
1405

ID	Method	ALL (%)					$\mathbb{L} \geq 5$ (%)					Episode Length \downarrow	Episode Time (s) \downarrow
		SR \uparrow	SPL \uparrow	SSR \uparrow	NSR \uparrow	NSNPL \uparrow	SR \uparrow	SPL \uparrow	SSR \uparrow	NSR \uparrow	NSNPL \uparrow		
	Random	4.12	2.21	—	—	—	0.21	0.08	—	—	—	38.12	—
I	SSCNav [23]	77.14	31.09	89.14	86.54	51.72	71.73	34.33	89.02	80.58	50.73	35.26	1.342 \times
	PONI [32]	78.58	33.78	89.48	87.81	52.39	72.92	36.40	89.13	81.81	51.82	33.64	1.591 \times
II	OMT [14]	71.13	37.27	93.17	76.34	41.36	61.94	38.19	92.23	67.16	42.63	26.26	0.645
III	SP [39]	68.92	38.56	91.31	75.48	40.26	52.73	33.84	89.72	58.77	35.62	27.84	0.219
	SAVN [37]	63.12	37.81	89.16	70.79	40.71	52.01	34.94	88.10	59.04	38.55	27.32	0.272
	ORG [10]	67.32	37.01	91.07	73.88	40.24	58.13	35.90	90.27	64.40	38.69	26.17	0.241
	HOZ [41]	68.53	37.50	91.44	74.94	40.83	60.27	36.61	90.31	66.74	39.82	27.24	0.283
	VTNet [11]	72.24	44.57	94.18	76.62	46.74	63.19	43.84	92.85	68.06	46.15	20.01	0.321
	DOA [6]	74.32	40.27	95.73	77.63	44.11	67.88	40.36	93.92	72.27	44.03	22.86	0.334
IV	Ours (DAT)	82.39	48.93	97.25	84.71	50.32	76.21	49.32	96.22	79.20	50.14	19.87	0.352

1419 Table 9. Comparison with SOTA methods in RoboTHOR [8]. \times indicates unacceptable resource consumption.
1420

ID	Method	ALL (%)					$\mathbb{L} \geq 5$ (%)					Episode Length \downarrow	Episode Time (s) \downarrow
		SR \uparrow	SPL \uparrow	SSR \uparrow	NSR \uparrow	NSNPL \uparrow	SR \uparrow	SPL \uparrow	SSR \uparrow	NSR \uparrow	NSNPL \uparrow		
	Random	0.00	0.00	—	—	—	0.00	0.00	—	—	—	0.00	—
I	SSCNav [23]	38.12	14.10	61.37	62.13	35.14	33.46	11.04	60.91	54.93	33.93	106.37	4.145 \times
	PONI [32]	38.42	16.30	58.46	65.72	39.83	34.72	13.22	58.11	59.75	38.44	92.73	4.582 \times
II	OMT [14]	32.17	20.09	61.77	52.08	24.51	25.33	18.16	57.35	44.17	23.82	62.99	2.011
III	SP [39]	27.43	17.49	57.91	47.37	20.19	20.98	16.03	53.48	39.23	18.27	68.18	0.714
	SAVN [37]	28.97	16.59	57.23	50.62	19.55	22.89	15.21	54.84	41.74	19.14	67.22	0.812
	ORG [10]	30.51	18.62	59.64	51.15	20.64	23.89	14.91	54.64	43.72	19.51	69.17	0.769
	HOZ [41]	31.67	19.02	60.11	52.68	21.02	24.32	14.81	54.23	44.85	20.38	66.26	0.808
	VTNet [11]	33.92	23.88	63.29	53.59	28.26	26.77	19.80	57.72	46.38	27.50	60.27	1.325
	DOA [6]	36.22	22.12	64.18	56.43	25.88	30.16	18.32	61.39	49.13	25.11	61.24	1.247
IV	Ours (DAT)	41.72	27.91	67.24	62.04	34.27	37.16	22.46	66.13	56.19	34.10	58.81	1.211

1433 also assess its drawbacks, offering suggestions for further
1434 study and advancement.1435

E.1. Experimental Setup

1436 In the test floorplans of AI2-Thor, we initialized 8000
1437 tasks (scene, initial position and target object) at random.
1438 We independently count the indicators (SR, SPL, SSR, NSR
1439 and NSNPL) of each target object when the agent completes
1440 these 8000 tasks. To observe how the agent performs
1441 when confronted with long path tasks, we additionally extract
1442 episodes with path lengths higher than 5. The experimental
1443 results for the DOA [6] method and our DAT method
1444 are presented in Tables 10 and Table 11, respectively.1445

E.2. Result Analysis

1446 **Comparison of Different Target Objects** As Tables 10
1447 and Table 11 show, the SR and SPL have a large variance
1448 on various targets, and even the gap between the maximum
1449 and minimum SR can reach more than 70%. This finding
1450 demonstrates that the navigation difficulty varies greatly as
1451 the target changes. According to the definition of SR, SSR
1452 and NSR, we obtain the following relationship:

1453
$$SR \approx SSR \times NSR \quad (12)$$

1454 \approx is used because there are a few instances where navigation
1455 is successful even though the target object is not recognized.
1456 Therefore, the variance fluctuation of the SR consists of the variance
1457 fluctuation of the SSR and the variance fluctuation of the NSR.
1458 Comparing the SSR column and the NSR column in Tables 10,
1459 it can be determined that the large variance of the SR is caused by the NSR and all
1460 targets achieve a high level of SSR. This indicates that the
1461 current end-to-end object navigation algorithm's difficulty
1462 is reflected primarily in the "navigate to" phase while the
1463 ability to find targets in the "search for" phase has basically
1464 matured. For example, when using the DOA method, the
1465 success rate (SR) of the most difficult target alarm clock
1466 is only 31.42%, but the success rate for the "search for"
1467 phase (SSR) has reached 99.28%. From this perspective,
1468 our DAT method is very specifically tailored to address the
1469 main shortcoming of existing end-to-end object navigation
1470 techniques.1471 **Comparison of Different Methods** Figure 9 illustrates
1472 our DAT method's influence on each target object. In the
1473 main text, we analyze the NSNPL which improves most

1512 obviously. In addition, NSR has also been significantly im-
1513 proved on various target objects. This displays that the navi-
1514 gation thinking we introduced not only optimizes the path of
1515 the “navigate to” phase but also substantially raises the suc-
1516 cess rate of the “navigate to” phase. Unexpectedly, our DAT
1517 approach significantly improves the SSR on only several
1518 challenging search objects and even degrades on some tar-
1519 get objects. This phenomenon is attributed to two reasons:
1520 (i) The original DOA method has a strong target search abil-
1521 ity through search thinking, so it is challenging to make a
1522 significant development without introducing more powerful
1523 search strategies. (ii) The search ability for each target is
1524 balanced after introducing navigation thinking and thinking
1525 adaptive fusion techniques.

1526 Additionally, Figure 9 highlights which target objects re-
1527 spond best to our strategy and which ones do not. Obvi-
1528 ously, the alarm clock and the cell phone are the two targets
1529 that our DAT method benefits the most; and of all the target
1530 objects, they are the two tiniest and hardest to locate. The
1531 difficulty in finding them is that the probability of the target
1532 recognition failing is greatly increased. Our DAT method
1533 considerably enhances the information use of each target
1534 identification by abstracting memory, which benefits small
1535 targets that are challenging to recognize. At a higher level,
1536 the differences between the different scenes are also quite
1537 obvious. The only few targets that perform poorly with our
1538 strategy are common objects in the kitchen. As discussed
1539 in the main text, the environment layout of the kitchen is
1540 relatively simple, so a simple end-to-end model sometimes
1541 works better. Therefore, we believe that it is necessary for
1542 the agent to adjust the thinking strategy in response to the
1543 perception of the environment layout. This is also a think-
1544 ing direction we provide for researchers.

F. Scene Level Indicators

1545 We discover that the contrasts between various scenes
1546 are quite clear from the analysis of each target object.
1547 Therefore, we conduct experiments on the DOA method
1548 and our DAT method in different scenes. We randomly se-
1549 lect 1000 tasks from each scene and let the two methods
1550 complete these 1000 tasks simultaneously. The results are
1551 illustrated in Table 12.

E.1. Result Analysis

1552 On all metrics, the bedroom scene benefits the most from
1553 our DAT method. Obviously, the bedroom is the most
1554 challenging to navigate of the four scenes. Therefore, our
1555 method is more helpful for complex and difficult scenes.
1556 Bathroom and Kitchen are two simpler scenes that require
1557 fewer steps to navigate to the target object. The optimiza-
1558 tion of our method for these two scenes is not so obvious.

G. Qualitative Results

1559 In the main text, due to space limitations, we select only
1560 a simple scene for a qualitative analysis of navigation be-
1561 havior. In Figure 10, we visualize the agent’s navigation
1562 paths with different targets in four complex scenarios. We
1563 discover that when the agent needs to make a key decision,
1564 there is often no target in view. At these critical decision-
1565 making moments, our DAT method can provide the agent
1566 with the relative position information of the target, thereby
1567 improving the critical decision-making success rate. More
1568 significantly, our method makes each step of the agent more
1569 stable and purposeful in the “navigate to” phase. Reflecting
1570 on the navigation route, once the target has been seen, our
1571 method reduces the movement for in-situ exploration and
1572 applies more forward movement to navigate to the target
1573 position more efficiently.

H. Multiple Adaptive Thinking in Embodied AI

1574 The dual adaptive thinking (DAT) network proposed in
1575 this paper provides key inspiration for future research. In
1576 object navigation tasks, dual adaptive thinking can be ex-
1577 tended to multiple adaptive thinking. Environment model-
1578 ing thinking, object state understanding thinking, and other
1579 types of thinking can be introduced in multiple adaptive
1580 thinking models. Furthermore, multiple adaptive thinking
1581 is not limited to object navigation tasks. In other embed-
1582 ded AI tasks, such as embodied question answering (EQA)
1583 [7] and visual language navigation (VLN) [2], agents can
1584 use multiple thinking approaches to more flexibly address
1585 real-world problems.

DOA	ALL (%)					$\mathbb{L} \geq 5$ (%)					Episode Length↓
	SR↑	SPL↑	SSR↑	NSR↑	NSNPL↑	SR↑	SPL↑	SSR↑	NSR↑	NSNPL↑	
Alarm Clock	31.42	17.65	99.28	31.65	17.77	25.58	15.67	99.22	25.78	14.87	37.35
Book	54.54	27.31	84.84	64.28	31.74	44.16	25.75	80.00	55.20	31.15	25.43
Bowl	88.33	51.55	100.00	88.33	58.33	84.84	53.59	100.00	84.84	57.48	12.85
Cell Phone	48.90	25.05	86.26	56.69	32.24	35.24	20.28	79.50	44.33	23.95	33.28
Chair	63.24	33.08	98.02	64.52	31.59	53.00	32.28	97.26	54.49	31.08	26.19
Coffee Machine	91.66	53.39	100.00	91.66	53.82	87.23	57.87	100.00	87.23	54.80	12.38
Desk Lamp	78.94	42.74	93.42	84.50	49.79	69.81	42.06	90.56	77.09	46.24	17.42
Floor Lamp	54.13	27.89	95.48	56.69	29.98	49.56	28.90	94.78	52.29	30.35	27.31
Fridge	77.77	42.66	100.00	77.77	40.59	72.09	45.06	100.00	72.09	44.12	21.18
Garbage Can	69.83	44.00	96.83	72.12	45.65	65.67	44.58	96.34	68.16	45.30	21.69
Kettle	87.50	55.95	100.00	87.50	64.39	85.71	59.42	100.00	85.71	64.07	18.00
Laptop	82.49	44.33	95.72	86.19	46.84	74.28	42.12	93.71	79.26	44.59	18.07
Light Switch	82.71	47.95	97.95	83.87	53.07	75.14	50.06	98.22	76.21	50.24	17.22
Microwave	91.23	46.97	100.00	91.22	46.50	87.50	52.92	100.00	87.50	51.16	15.49
Pan	62.26	35.40	94.34	66.00	40.34	56.82	30.99	93.18	60.97	38.83	24.92
Plate	79.63	41.12	96.29	82.69	43.41	72.97	42.23	94.59	77.14	44.35	17.57
Pot	56.86	30.42	92.15	59.57	33.91	41.93	23.56	87.09	44.45	25.75	25.98
Remote Control	77.52	43.35	93.02	82.49	45.75	69.76	40.91	90.69	76.92	41.89	17.07
Sink	90.17	43.43	94.57	95.35	44.80	80.34	44.86	90.34	88.93	47.74	13.46
Stove Burner	93.49	58.89	100.00	93.49	62.92	89.19	60.55	100.00	89.19	60.34	13.11
Television	84.21	44.84	99.12	84.95	43.88	79.01	48.43	98.76	80.00	49.81	18.07
Toaster	76.19	46.00	100.00	76.19	51.56	72.72	46.83	100.00	72.72	50.43	17.16

Table 11. The outcome of applying the DAT method with each object as the target. ✓ indicates the target objects that benefit the most from our method.

DAT	ALL (%)					$\mathbb{L} \geq 5$ (%)					Episode Length↓
	SR↑	SPL↑	SSR↑	NSR↑	NSNPL↑	SR↑	SPL↑	SSR↑	NSR↑	NSNPL↑	
Alarm Clock ✓	45.00	29.22	96.43	46.67	30.37	40.31	27.22	96.12	41.94	28.27	24.79
Book	59.39	32.52	89.69	66.21	36.98	45.83	30.94	85.83	53.39	34.54	29.31
Bowl	86.66	56.08	98.33	88.13	56.94	81.81	58.51	100.00	81.82	51.65	9.98
Cell Phone ✓	68.68	32.79	90.65	75.76	39.76	57.37	28.68	86.88	66.05	34.74	27.86
Chair	67.19	36.01	98.42	68.27	35.24	58.46	36.83	97.81	59.76	36.83	20.38
Coffee Machine	93.05	49.39	97.22	95.71	51.09	91.48	56.89	95.74	95.56	59.01	15.40
Desk Lamp ✓	93.42	54.97	94.74	98.62	57.54	90.56	58.33	92.45	97.95	61.35	13.14
Floor Lamp ✓	54.88	34.01	93.98	58.39	38.49	52.17	35.74	93.04	56.07	40.20	26.00
Fridge ✓	88.88	47.76	100.00	88.88	50.46	90.69	52.72	100.00	90.69	56.14	17.74
Garbage Can	77.83	50.76	96.27	80.85	52.16	73.91	50.66	95.88	77.08	52.22	18.90
Kettle	93.75	61.50	100.00	93.75	68.28	92.86	60.76	100.00	92.86	68.37	18.68
Laptop	84.82	48.96	94.55	89.71	52.77	77.71	49.27	92.00	84.47	50.98	14.53
Light Switch	86.43	52.59	97.58	88.57	58.76	81.36	54.65	97.63	83.34	56.47	16.35
Microwave	96.49	55.64	100.00	96.49	53.92	95.00	60.81	100.00	95.00	59.28	12.63
Pan	47.17	28.02	92.45	51.02	33.74	43.18	27.44	90.91	47.49	31.14	24.73
Plate	79.63	44.63	100.00	79.63	42.83	72.97	41.59	100.00	72.97	43.30	17.37
Pot	50.98	30.94	90.19	56.52	37.68	32.26	83.87	83.87	38.45	24.63	30.88
Remote Control	78.29	46.25	93.02	83.34	49.90	72.09	45.68	90.69	79.49	47.72	17.01
Sink	89.00	48.63	94.86	93.66	49.30	78.27	54.22	91.03	85.98	55.62	12.79
Stove Burner	95.93	64.36	100.00	95.93	68.16	93.24	67.24	100.00	93.24	65.99	10.47
Television	80.70	45.69	98.24	82.14	47.80	74.07	47.84	97.53	75.94	49.92	14.28
Toaster	88.09	51.17	100.00	88.09	55.95	87.88	54.67	100.00	87.88	58.18	15.31

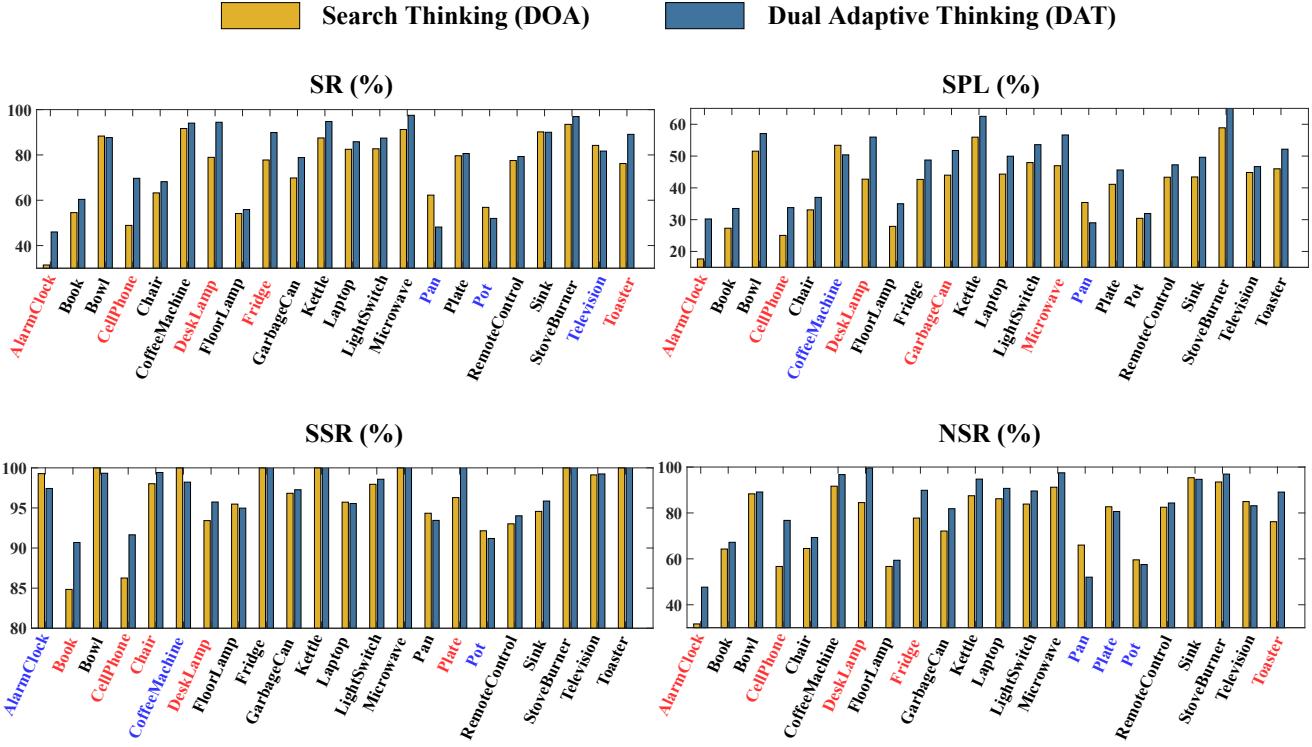


Figure 9. We analyze the SR, SPL, SSR, and SNE metrics for each object. Target objects that have greatly improved after utilizing our DAT method are shown by name in red. Target objects whose performance suffers because of applying our method are shown by name in blue.

Table 12. We compare the improvement after using our DAT method in different scenes. Bold values indicate the scene that benefit the most.

Scene	Method	ALL (%)					$\mathbb{L} \geq 5$ (%)					Episode Length \downarrow
		SR \uparrow	SPL \uparrow	SSR \uparrow	NSR \uparrow	NSNPL \uparrow	SR \uparrow	SPL \uparrow	SSR \uparrow	NSR \uparrow	NSNPL \uparrow	
Living Room	DOA	69.15	38.14	95.88	72.12	39.35	61.81	37.45	94.76	65.22	38.31	23.024
	DAT	72.54	42.32	95.80	75.72	44.76	66.91	43.26	94.49	70.81	44.81	20.726
	DAT - DOA	3.39	4.18	-0.08	3.60	5.41	5.10	5.81	-0.27	5.59	6.50	-2.298
Kitchen	DOA	83.89	49.30	98.78	84.82	52.41	78.76	50.96	98.34	79.94	51.72	16.817
	DAT	86.01	52.31	98.68	87.16	53.53	82.47	54.49	98.34	83.86	53.85	16.453
	DAT - DOA	2.12	3.01	0.10	2.34	1.12	3.71	3.53	0.00	3.92	2.13	-0.364
Bedroom	DOA	62.10	34.51	93.60	66.35	38.17	51.17	31.92	91.31	56.04	33.73	24.418
	DAT	73.10	41.81	95.20	76.79	44.81	63.20	40.11	93.37	67.69	41.96	21.05
	DAT - DOA	11.00	7.30	1.60	10.44	6.64	12.03	8.19	2.06	11.65	8.23	-3.368
Bathroom	DOA	87.10	45.64	95.00	91.47	48.94	78.72	48.54	92.84	84.80	51.80	13.848
	DAT	88.90	50.34	95.10	93.48	50.98	81.12	54.49	93.04	87.19	55.03	13.582
	DAT - DOA	1.80	4.70	0.10	2.01	2.04	2.40	5.95	0.20	2.39	3.23	-0.266

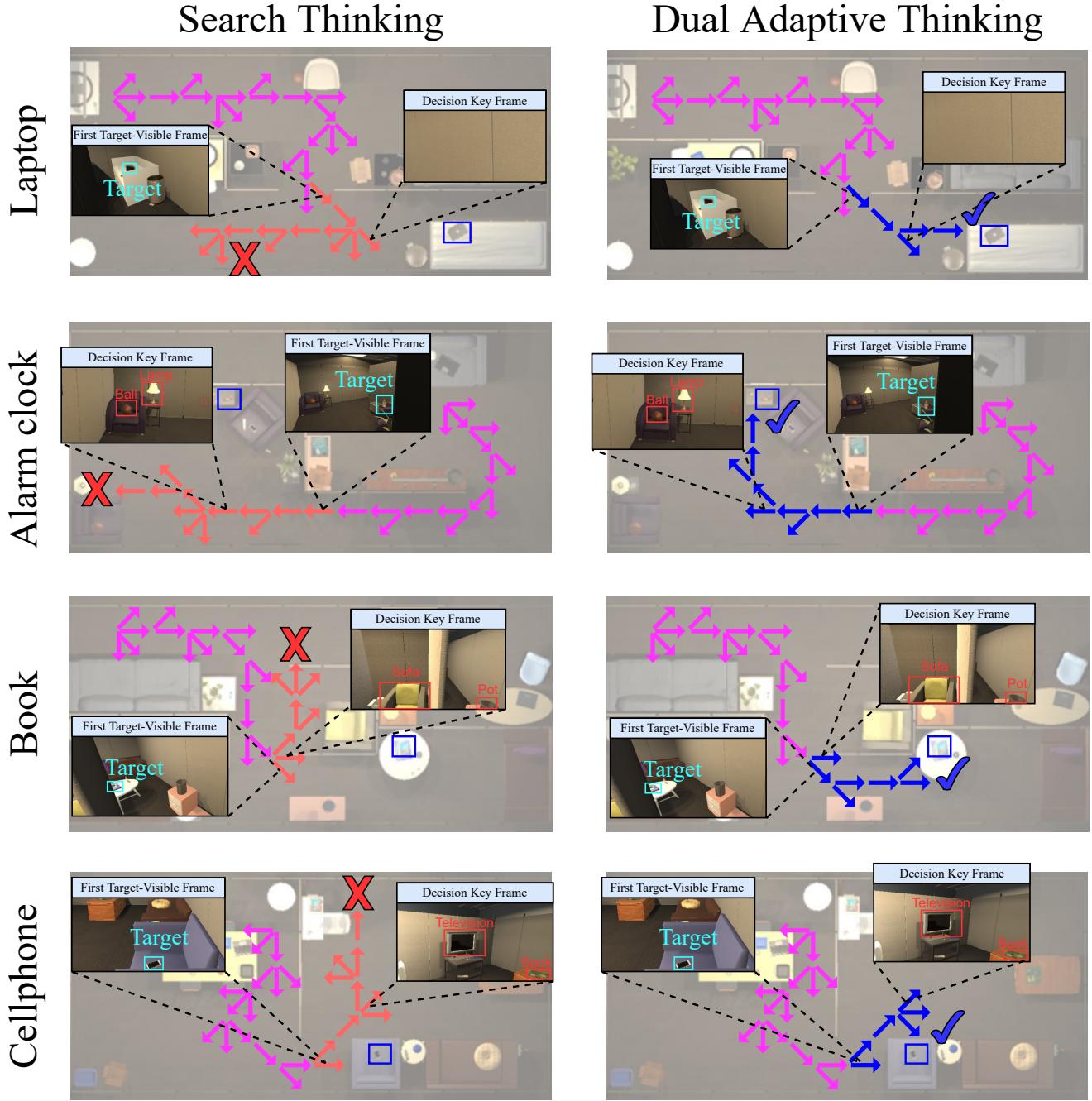
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
18511852
1853
1854
1855
1856
1857
1858
1859
1860
1861
18621863
1864
1865
1866
1867
1868
1869
1870
1871
18721873
1874
1875
1876
1877
1878
1879
1880
1881
1882
18831890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943

Figure 10. Visualization of navigation trajectories on the RoboTHOR dataset. The pink arrow indicates the path of the “search for” phase. The red arrow indicates the path of the “navigate to” phase when only search thinking is used. The blue arrow indicates the path of the “navigate to” phase when DAT is used.