Object Memory Transformer for Object Goal Navigation

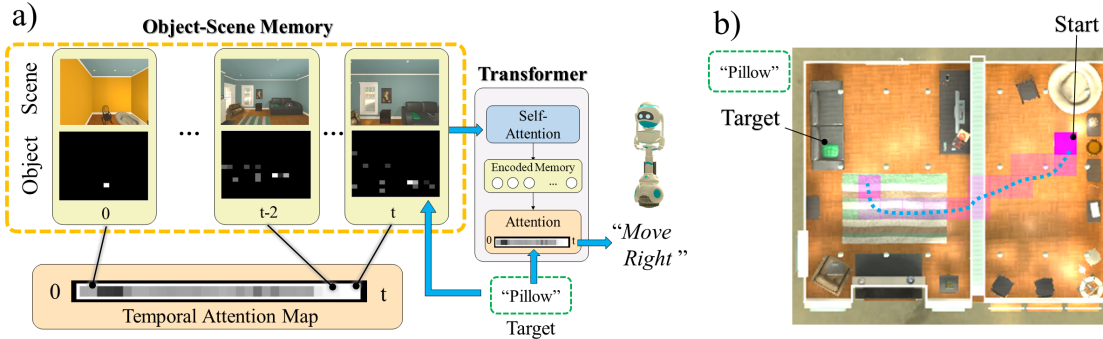Rui Fukushima[1], Kei Ota[2,3], Asako Kanezaki[2], Yoko Sasaki[1] and Yusuke Yoshiyasu[1]

Fig. 1: a) Object Memory Transformer (OMT) for efficient indoor navigation to find the target object. Here, the agent looks for "Pillow" on the sofa, which is in the next room and the agent cannot observe it from the start position. Even in this complex case, OMT can take an efficient path to the target shown as a trajectory in the blue dots in (b) by exploiting long-term visual cues while taking into account the relevance of the feature at each time step. Specifically, OMT stores long-term history of observed scenes and objects into Object-Scene Memory (OSM), attends to the crucial ones in OSM, and produces useful features for the RL agent. At the begging of the episode, when the agent is looking at a wall, it rarely attends to those histories (notice at the dark gray region in the attention map of (a)). Once the target comes into the agent's sight, the attention weight gets higher (shown in white).

Abstract— This paper presents a reinforcement learning method for object goal navigation (ObjNav) where an agent navigates in 3D indoor environments to reach a target object based on long-term observations of objects and scenes. To this end, we propose Object Memory Transformer (OMT) that consists of two key ideas: 1) Object-Scene Memory (OSM) that enables to store long-term scenes and object semantics, and 2) Transformer that attends to salient objects in the sequence of previously observed scenes and objects stored in OSM. This mechanism allows the agent to efficiently navigate in the indoor environment without prior knowledge about the environments, such as topological maps or 3D meshes. To the best of our knowledge, this is the first work that uses a long-term memory of object semantics in a goal-oriented navigation task. Experimental results conducted on the AI2-THOR dataset show that OMT outperforms previous approaches in navigating in unknown environments. In particular, we show that utilizing the long-term object semantics information improves the efficiency of navigation.

I. INTRODUCTION

Object goal navigation (ObjNav) is a task of navigating an agent toward a target object given by a word with reference to the first-person view [1], [2]. To solve the task, a navigation agent is required to learn highly nonlinear mappings from images and words to actions. Unlike in game-playing tasks such as GO, where Deep Reinforcement Learning (DRL) agents defeated human professionals [3], [4], DRL agents' performances on ObjNav are still far behind from that of average humans [2].

The main challenge in ObjNav is that the task requires complex scene understanding and can sometimes become long-horizon. In many cases, the target object is not visible from the start position and thus the agent needs to get close to the object by understanding the surrounding environment to plan its paths from the first-person views in which the goal may not be present. Further, these views drastically change according to the agent's moves and sometimes do not contain any meaningful information. For example, when the agent faces to a white wall, there is no visual clue in the view to decide the next move. As a result, the learning agent may easily be stuck and pivot at the same place almost forever or may take an unexpectedly none-smooth trajectory that repeatedly arrives at the same locations back and forth. Another difficult point in ObjNav is its diversity: the trained agent model is required to generalize to unknown environments with different room structures, scenes, objects, etc., which makes this task even harder.

Recent DRL navigation approaches for ObjNav have achieved remarkable improvements in performance using semantic and spatial knowledge about objects, such as object relationship, spatial object context, and object class hierarchy [1], [2], [5], [6], [7], [8]. However, the knowledge used in these studies is independent from the temporal context, which means that it lacks the information about the objects

---

[1]Rui Fukushima, Yoko Sasaki and Yusuke Yoshiyasu are with National Institute of Advanced Industrial Science and Technology (AIST), Japan `rui.fukushima, y-sasaki, yusuke-yoshiyasu@aist.go.jp`

[2]Kei Ota and Asako Kanezaki are with Tokyo Institute of Technology, Tokyo, Japan `kanezaki@c.titech.ac.jp`

[3]Kei Ota is with Information Technology R&D Center, Mitsubishi Electric Corporation, Kanagawa, Japan. `Ota.Kei@ds.MitsubishiElectric.co.jp`

that are previously seen within the episode or it does not respect the order in which objects appeared. Although some of these studies have used Recurrent Neural Networks (RNNs), such as LSTM, it is not clear if they can understand the ordering of long-range sequences [9].

Our goal is to investigate the effects of long-term observation of scenes and objects in ObjNav, and how we can effectively use them to improve navigation performance. To this end, we propose Object Memory Transformer (OMT) that consists of the following two key ideas, detailed in Fig. 1. Firstly, we encode the current scene and object to fixed dimensional feature vectors, and store them in an external memory named Object-Scene Memory (OSM). This memory is capable of storing scene observations and object semantics throughout a long-term sequence in a temporally consistent manner. Then, a Transformer network attends to the crucial scenes and objects, which are stored in OSM, and produces useful features for the RL agent. We test our model on publicly available synthetic indoor environments, and show that OMT is able to navigate by taking efficient routes toward the target object. We also investigate whether our memory architecture improves the performance and efficiency in the ObjNav task.

II. RELATED WORK

A. Object-goal Navigation

Yang et al. [1] proposed the first navigation RL agent to tackle ObjNav, using Graph Convolution Networks (GCNs) [10] for incorporating scene priors. Since then, various studies have attempted to use semantic knowledge about objects to improve generalization performance to unseen targets and environments. In particular, the approaches that combine semantics and spatial knowledge about objects have achieved superior performance [2], [5], [6], [7], [8]. However, in ObjNav a long-term sequence of observations has been rarely considered or exploited. In this paper, we propose a new external memory called Object-Scene Memory (OSM) to store the long-term time series of observed objects, and then use Transformer [11] to extract useful features for ObjNav.

B. Memory Based Network and Transformer

Recurrent Neural Networks (RNNs) [12] have been commonly used to capture time-series information. However, RNNs are known to be insufficient to handle long-term time-dependency. Therefore, a mechanism to store variables in an external memory outside the network has attracted attentions [13], [14], [15], [16]. In the field of natural language processing (NLP), a method called Transformer [11] has been proposed and has shown its ability to process long sentences with many words based on the self-attention mechanism. The transformer architecture has become the de facto standard in NLP and has recently been applied to computer vision problems [17], [18].

C. Navigation agents with memory

Savinov et al. [19] proposed a memory architecture called Topological Memory, referring to landmark-based navigation in animals. Khan et al. [20] used the Differentiable Neural Computer (DNC) [16] in grid-world navigation. Zhu et al. [15] extended Neural Episodic Control (NEC) and proposed Episodic Reinforcement Learning with Associative Memory (ERLAM) [21] to improve sample efficiency. ERLAM associates related experience trajectories by considering the relationship between states. Fang et al. [22] proposed Scene Memory Transformer (SMT) that stores long-term scene observations for none goal-directed navigation tasks. Du et al. [23] proposed a Tentative Policy that avoids deadlock and loop by utilizing attention mechanisms for attending to all past actions stored in memory. Unlike these works, our Object-Scene Memory stores long-term object semantics that can support the agent to take an efficient trajectory to reach the target object in ObjNav.

III. PROPOSED METHOD

In this section, we describe our approach to learning a navigation agent for the ObjNav task. We start with explaining the task settings, and then introduce our novel model architecture that consists of two key components: 1) Object Scene Memory (OSM) that stores a history of observed scenes and objects; 2) Object Memory Transformer (OMT) that takes the history of observations from OSM, and produces useful features that will be taken as input to the subsequent RL agent.

A. Task Setting

ObjNav is a task that considers a setting where an agent is placed at a random initial location in a 3D indoor environment, and required to reach a target object defined in the target class set $G = \{g_0, ..., g_M\}$ while minimizing the number of steps. Each task $\tau \in \mathcal{T}$ is represented by $\tau = \{e_i, p_i, g_i\}$, where an environment e_i is given from a set of rooms $E = \{e_0, ..., e_N\}$, a starting position p_i, and a target object $g_i \in G$. At each state, the agent takes a RGB image in a first-person view, which we call *scene*, and *objects* in the form of bounding boxes, which exist in the scene. We use the terms *scene* and *object* in this manner.

The action space from which the agent takes an action at each time step consists of the following nine commands: $\mathcal{A} = \{$*Move Forward, Move Backward, Move Right, Move Left, Rotate Right, Rotate Left, Look Up, Look Down, Done*$\}$. The agent moves on the discretized scene space divided by $0.5\,\mathrm{m}$, and the rotation and the vertical tilt angle are 45 degrees and 30 degrees, respectively.

Each episode is considered to be success if the agent chooses *Done* action at positions where the target object is *visible*, otherwise it is considered to be failure. Here, *visible* becomes true when the target object is within threshold distance, which we set to be $1.5\,\mathrm{m}$, from the agent.

B. Model Architecture

The proposed model architecture is shown in Fig. 3. The agent model is based on Transformer for handling the long-term history of object semantics, which is inspired by Druon et al. [2] and Fang et al. [22]. Our method consists of 1) Feature Extractor, 2) Object-Scene Memory, 3) Transformer,
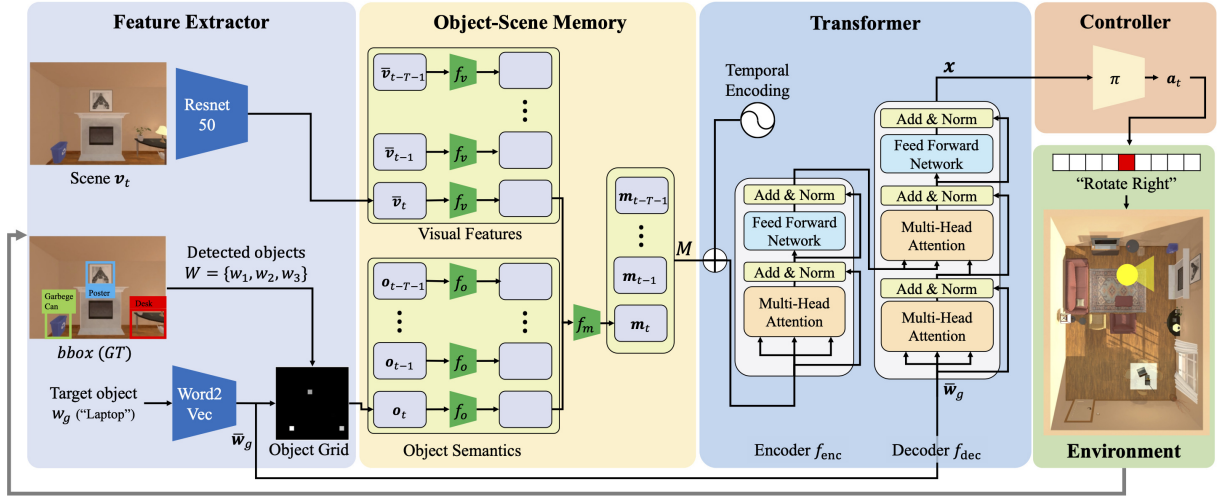
Fig. 3: Architecture of OMT that consists of four steps: 1) Feature Extractor extracts visual features $\bar{v}_t$ from RGB image v_t using ResNet-50. It also encodes the relationship between target object w_g and the objects observed in the scene, $W = \{w_1, ..., w_{N^{\mathrm{obj}}}\}$. This is done by computing the word embeddings of those objects using word2vec and cosine similarities between them, which will be assigned to the values in the object context grid, o_t. 2) OSM stores scene appearance $\bar{v}_t$ and object semantics o_t for the last T-history length, and fuses both of those feature vectors into m_t using three learnable networks f_v, f_o, and f_m. We apply this operation to all data stored in OSM, and obtains a set of features $M_t = \{m_{t-T-1}, ..., m_t\}$; 3) Transformer then takes M_t^{TE} so that the model can learn useful temporal feature representations for ObjNav, which are conditioned by the target object; 4) Finally, an A3C-based controller outputs action a_t to move in the 3D indoor environment.

and 4) Controller. We describe the above components in details.

1) Feature Extractor: The feature extractor takes the current scene v_t and object words $W = \{w_1, ..., w_{N^{\mathrm{obj}}}\}$ and w_g that specify the objects shown in the current scene and the target object, and outputs the features of the current scene and the target object. To extract the visual feature, we use pre-trained ResNet-50 [24] that takes the 400×300 RGB image v_t and produces 2048-dimensional visual feature $\bar{v}_t \in \mathbb{R}^{2048}$. The semantic feature of the target, which is specified by the word w_g, is converted into 300-dimensional word embedding vector $\bar{w}_g \in \mathbb{R}^{300}$ using word2vec [25]. We do not update the parameters of these feature extractors during the whole process.

In order to efficiently extract objects' semantics in the current view, we use the object context grid representation (object grid) proposed in [2]. The object grid o_t consists of an array with 16×16 cells, $o_t \in \mathbb{R}^{16 \times 16}$. We assign a value to each cell that indicates the similarities between the target object and the detected objects. The positions to which these values are assigned correspond to the centers of the bounding boxes of the detected objects. The similarities between target object and the detected objects are calculated by the cosine similarity as:

$$o_{i,j} = \frac{\bar{w}_{i,j} \cdot \bar{w}_g}{||\bar{w}_{i,j}||||\bar{w}_g||},$$

where $\bar{w}_g$ and $\bar{w}_{i,j}$ denote the word embedding vectors of the target and an object whose center of bounding box detected at a location corresponding to i-th row and j-th column of the object grid o_t, respectively. When no object located at

$o_{i,j}$, the value becomes to be 0. As for the object set W, we assume the ground truth object labels and bounding boxes for all the objects in the view are available following [2], [5].

2) Object-Scene Memory: Next, we introduce Object-Scene Memory (OSM) that has two functionalities: 1) storage of the visual appearance of the scene $\bar{v}_t$ and the object semantics o_t; 2) extraction of useful features from the stored data. Focusing on the first one, OSM can be considered as a ring-buffer that can store T-history length. We prepare this ring-buffer for scene appearance $\bar{v}_t$ and the object semantics o_t (see Fig. 3). Secondly, OSM fuses the spatial and object semantics to the 300-dimensional fused vector $m_t \in \mathbb{R}^{300}$. To that end, OSM employs three feature extractors f_v, f_o, f_m that take scene appearance v and object semantics o as input. They fuse v and o to produce fused features m as:

$$m_t = f_m(f_v(\bar{v}_t), f_o(o_t)). \qquad (1)$$

We do this operation for all time frames of OSM, producing a set of fused features, $M_t = \{m_{t-T-1}, ..., m_t\}$.

3) Transformer: Transformer receives features stored in memory and produces useful features to the subsequent RL agent. We implemented temporal encoding and provided it to Transformer so that it can get a reference to temporal ordering of the memory.

Temporal Encoding. Since we hypothesize that temporal information can improve the performance in the ObjNav task, we make use of the order of what the agent has observed in an episode by utilizing the temporal encoding. Among different implementations are proposed, we specifically use the trigonometric functions proposed in [11]. Let i be the current dimension of temporal encoding. Also, let T be the

size of M. Then, the memory with temporal order encoded, M^{TE}, can be formulated as:

$$M^{TE}_{(pos,2i)} = \boldsymbol{m}^{pos}_{t(2i)} + \sin\left(\frac{pos}{10000^{2i/T}}\right) \quad (2)$$

$$M^{TE}_{(pos,2i+1)} = \boldsymbol{m}^{pos}_{t(2i+1)} + \cos\left(\frac{pos}{10000^{2i/T}}\right), \quad (3)$$

where $pos \in \{0, ..., T-1\}$ and $\boldsymbol{m}^{pos}_t$ represent the location of the memory slot and feature $\boldsymbol{m}_t$ located at the pos-th slot of memory, respectively.

Encoder-Decoder networks. The temporally encoded scene and object context features are then fed into the Transformer encoder-decoder network [11] to obtain a useful representation for the RL agent. Transformer employs a technique called self-attention to determine the relevance of each element to other elements in a temporal data and outputs contextualized features.

The output of OSM along with temporal encoding M^{TE} is taken as an input to the Tranformer encoder f_{enc}, and the encoder extracts spatiotemporal and object semantics in the memory. Then, the decoder f_{dec} produces useful representations $\boldsymbol{x}$ acquired from the attention between the encoded memory and the target object to the controller.

$$\boldsymbol{x} = f_{\text{dec}}(\boldsymbol{w}_g, f_{\text{enc}}(M^{TE})) \quad (4)$$

Note that the encoder takes masked attention for both self-attention and source-target attention when the memory slot is empty, i.e., when the current time step is shorter than the size of the OSM.

4) Controller: Finally, the features computed by the Transformer are passed to the controller. We specifically use Asynchronous Advantage Actor-Critic (A3C) [26] for an RL agent. In our implementation, the model has two fully-connected layers and outputs both Q-values and the $|\mathcal{A}|$-length vector that has probability of each categorical action.

Overall, the policy at time t can be expressed as:

$$s_t = (\boldsymbol{v}_t, \boldsymbol{w_g}, M_{t-1}) \quad (5)$$

$$a_t \sim \pi(a_t \mid s_t, \boldsymbol{\theta}), \quad (6)$$

where s_t and a_t are the state and action at time t and $\boldsymbol{\theta}$ are the model parameters.

C. Reward Design

The RL agent is trained to maximize the expected return defined as the expected cumulative reward $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$. We define the reward function in Eq. (7), which is a mixture of the one proposed in [27] and [28] (details are described in [2]).

$$r = \begin{cases} 5.0 & \text{if success} \\ S_{bbox} & \text{if } S_{bbox} \text{ is the highest in the episode} \\ -0.01 & \text{otherwise,} \end{cases} \quad (7)$$

where S_{bbox} is computed by the ratio of the bounding box area in the field of view, and $S_{bbox} = 1$ when the entire field of view is covered by the target object[1].

---

[1] If S_{bbox} is not used, the agent immediately selects the "$Done$" signal to avoid a decrease in the cumulative reward due to the huge state space and reward sparsity, resulting in a failure of learning. To avoid this, pre-training by imitation learning [6], [7], [23] or meta learning [8], [29] could be used.

The first condition encourages the agent to accomplish the task, and the second condition encourages the agent to move closer to the target object. The third condition promotes to learn to navigate in the shortest path as the agent receives a penalty at each time step and tries to minimize it during training.

IV. EXPERIMENTAL SETTINGS

This section briefly describes the experimental settings including the dataset, evaluation metrics, and baseline models we used in the experiments to validate the proposed method. For more information about implementation details, please refer to the longer pre-print version on the arXiv.

A. Dataset

We use the AI2-THOR framework (The House Of inteR-actions) [30], which consists of 120 different photo-realistic 3D indoor environments for our experiments. We use 80 environments for training, i.e., 20 environments from each of 4 categories: Kitchen, Living room, Bedroom, Bathroom. The target class G for each category is manually selected in advance from predefined 146 objects class as follows: Kitchen: {*Toaster, Microwave, Fridge, Coffee Machine, Garbage Can, Bowl*}; Living room: {*Pillow, Laptop, Television, Garbage Can, Bowl*}; Bedroom: {*House Plant, Lamp, Book, Alarm Clock*}; Bathroom: {*Sink, Toilet Paper, Soap Bottle, Light Switch*}. Objects are placed in the room at random locations determined by the AI2-THOR framework.

After training, we test the model in novel environments which are not included in the training environments to evaluate its generalization capability. Specifically, we used 20 different environments, i.e., 5 rooms from each of 4 categories. Note that when the AI2-THOR environments were created, some targets were either not exist in the environment or not visible from any possible locations. In such cases, they were omitted from the target set for that environment.

B. Evaluation Metrics

We employ success rate (SR) and success weighted by path length (SPL) [31] for our evaluation metrics. SR is defined as $\frac{1}{N}\sum_{i=1}^{N} \mathbb{I}^i_{\text{goal}}$, and SPL is defined as $\frac{1}{N}\sum_{i=1}^{N} \mathbb{I}^i_{\text{goal}}\frac{d^*}{d_i}$, where N represents the total number of evaluation episodes and $\mathbb{I}^i_{\text{goal}}$ is a success indicator, which is 1 if the i-th episode is successfully finished, and 0 otherwise. The success condition is described in Sec. III-A. The symbols d^* and d_i stand for the shortest path length in a given scene from the initial position to the target position, and the path length of i-th episode, respectively. In short, we can say a policy is *good* if it achieves high score on both SR and SPL.

C. Models

We used the following models to compare the performance.

1) Random: This model was implemented so that it can be shown how complex the experimental environments setup is. The agent uniformly samples one action from the set of actions $\mathcal{A}$ at each step.

2) Scene Prior (SP): This is an implementation of Scene Prior (SP) [1], which is the simplest method of implementing semantic information of objects, implemented by [2]. SP uses Graph Convolution Networks (GCNs) to extract object semantics, where each node represents object class and edge represents relationship such as "next" or "under/on." Our model differs from SP in dealing with observations: SP does not use spatiotemporal context of objects while ours does using OSM and OMT. SP takes four staked frames of scenes as input, while GCNs take only the current observation.

3) Baseline: We use the model proposed in [2] for the baseline, as this method also employs the object context grid calculated with the ground truth object detector. While this method also uses semantic and spatial knowledge about objects as input, it does not explicitly utilize temporal information. Therefore, we can evaluate the effectiveness of our memory structure by comparing the results against this method. Note that this model takes four frames of observations as input. We also test against the model with three LSTM layers, which enables the model to handle temporal information. The LSTM layers take a single frame of observation as input, and the object grid is fused with it.

4) OMT: We evaluate three models for our proposed method, whose size of the memory are $\{4, 16, 32\}$, respectively, to see the relationship between history length and its performance. We report the result with one layer of Transformer, but we have also obtained similar results with two layers. Please refer to Sec. III for more details.

V. EXPERIMENTAL RESULTS

In this section, we try to quantify the performance of OMT. Specifically, we try to answer the following questions:

- How does OMT perform on ObjNav in terms of SR and SPL compared against previous approaches?
- Which components of OMT contribute to the performance gain?

A. Performance of OMT

Comparison with baselines First, we evaluated the performance of the proposed model. TABLE I shows the comparison between different approaches using the two different metrics. We can clearly see that our method outperforms prior works in both SR and SPL metrics. The result of Random shows that the ObjNav task considers a huge navigation space and the agent cannot easily navigate to the goal state by chance.

Comparing the two Baseline models, Baseline showed SR and SPL of around 60% and 20%, respectively. In addition, Baseline-LSTM 3 layers, which can handle time-series information, showed a slight performance gain, i.e., 1.70% higher SR and 3.13% higher SPL than Baseline.

Focusing on our model, it achieves the best performance in terms of SR and SPL. Our model with 4 histories, which has the same history length as Baseline, shows higher SR (71.13%) than Baseline and Baseline-LSTM 3 layers. Furthermore, when the history length is long, our agent model is more likely to take the shortest path. In fact, our model with 32 histories shows the best SPL (27.51%), which suggests

TABLE I: Evaluation results on unknown scenes and known objects

Method	SR [%].	SPL [%]
Random	6.10	1.30
SP [1]	18.24	4.39
Baseline [2]	61.55	20.83
Baseline - LSTM 3 layer [2]	63.25	23.96
OMT - 32 hist.	69.39 (0.16)	**27.51** (0.10)
OMT - 16 hist.	70.00 (1.56)	27.12 (0.22)
OMT - 4 hist.	**71.13** (1.42)	26.66 (0.10)

TABLE II: Results without each component.

Method	32 hist.		4 hist	
	SR [%]	SPL [%]	SR [%]	SPL [%]
OMT(full)	69.39	**27.51**	71.13	26.66
w/o Object Memory	67.99	25.49	**71.23**	26.69
w/o Scene Memory	65.38	23.15	62.60	25.10
w/o Scene	62.30	24.28	65.59	24.81
w/o Temporal Encoding	59.69	21.95	61.41	23.29
w/o Transformer	64.27	25.43	68.84	25.71

that long-term histories can improve navigation efficiency by choosing a shorter path to the target.

Comparison with SOTA We also compared the performance against the current state-of-the-art (SOTA) techniques [23], [5] based on the SR and SPL metrics reported in the papers. Note that the experimental settings used in those works are not exactly the same as ours. Thus some care must be taken when comparing each other.

Qui et al. [5] proposes a navigation method that leverages a hierarchical object relationship in the ObjNav task. While the method is tested on the different grid step size ($0.25\,\mathrm{m}$) from ours ($0.5\,\mathrm{m}$), its action space is $|\mathcal{A}| = 5$. Thus it explores smaller action space and makes the task easier. The reported scores are SR of 65.3% and SPL of 21.1%. Hence, we can conclude that our method works better than [5], as our method achieves better performance for both of the two metrics on a harder problem (action space is bigger).

Du et al. [23] proposes the memory-augmented tentative policy networks to promote the agent to escape from deadlock states, such as looping or being stuck. Although we could not compare SPL as their definition is different from ours, the reported SR of 69.3% is 2% lower than our best result. Thus, we can conclude that our OMT outperforms this SOTA method as well.

B. Ablation study—Effects of each component

To analyze what component of our method contributes to the performance gain, we conducted an ablation study by removing one component from the full model and measuring the performance in the two metrics as in Sec. V-A. The results are shown in TABLE II. The model *w/o Object Memory* removes the context grid, which encodes object semantics in the current scene, from OSM and directly passes to the Controller. Likewise, for a model *w/o Scene Memory*, the current scene is also concatenated. On the other hand, the *w/o Scene*, the current scene is not used, and only the context grid is used as input. In the model of *w/o Temporal encoding*,

11292

the temporal order is given by a soft manner like [22] with using an exponential function which is concatenated with visual features. In the *w/o Transformer* model, Transformer is replaced with two fully connected layers and ReLU function.

To evaluate how much scene information (first-person views) contributes to the navigation performance, we first see the results of *w/o Scene Memory* and *w/o Scene*. To our surprise, we can see that both methods perform surprisingly well even without scene information, whose scores are comparable with the baseline model with LSTM. This shows that a decent level of navigation is possible only using spatiotemporal knowledge of objects without recognizing a whole scene image. Next, we removed the object memory from the full model and saw SR dropped 1.40% for 32 histories, but it was 0.1% higher than 4 histories. On the other hand, when all the components are incorporated into our model with 32 histories, it shows the highest SPL score. This indicates that navigation efficiency is the consequence of interacts between scene, object, and memory. Especially, the temporal encoding of Transformer was found to have the most prominent impact on navigation performance, indicating that the temporal ordering of histories in memory is vital for ObjNav.

C. Qualitative study

To qualitatively show the effectiveness of our key idea in using the long-term history in ObjNav, we plotted agents' trajectories for the episodes where the start and goal are far from each other. In these cases, it is likely that there are obstacles in its route and that the target is not visible from the start location.

Fig. 4 shows the top-down maps with the trajectories of Baseline, Baseline-LSTM 3 Layer, OMT-4 hist., and OMT-32 hist. Focusing on the *Fridge* results (left column), Baseline and OMT-4 histories move straight toward or move left to get close to the target from the start, but are unable to move due to an obstacle. In contrast, OMT-32 histories once got stuck in an obstacle, but is able to get out of it and reach the target. We believe this is because OMT can perceive the continuity and changes of observations over time. In particular, the OMT recognizes its deadlock state when the same observations continue to be perceived over a long time. Similarly in *Television* (middle column), OMT-32 histories can recover to move to goal position even it gets stuck in bumping into the desk before the television, while Baseline and Baseline-LSTM 3 layer select *Done* signal from outside the threshold range. In *Pillow* (right column), where the agent is unable to see the target from the start location because of the wall in the middle, OMT-32 histories once gets stuck in a loop at the beginning of the episode, but is able to get out of it and reached the target while Baseline stuck in a rotation loop until the episode exceeds the maximum number of steps. These results suggest that our proposal of utilizing the long-term information can help guide the agent in complex ObjNav tasks, where the agent cannot see the goal object at the start position, especially in settings where the distance between the start and goal is long.
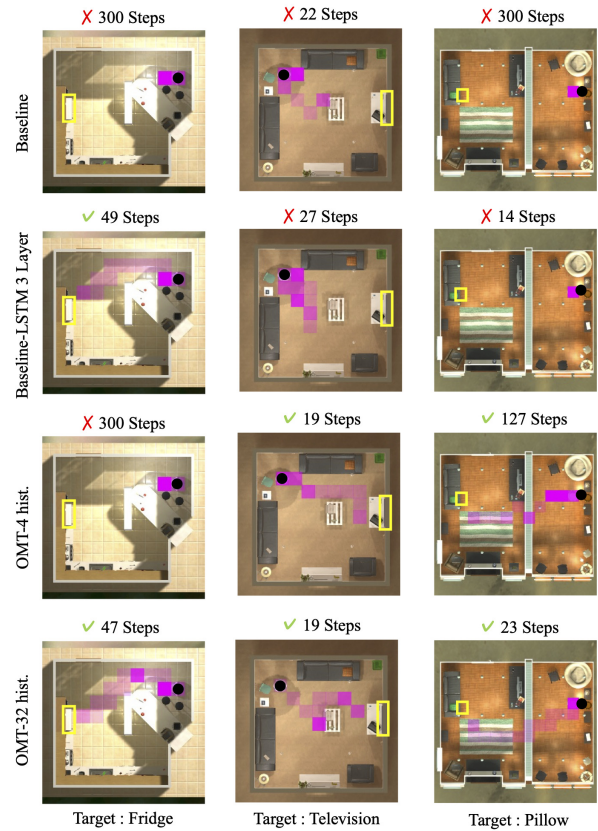

Fig. 4: Qualitative results. The top-down map shows the trajectories of different methods. The initial location of the agent is shown as a black dot, and the target object is circled by a yellow box. Trajectories are indicated by purple squares of varying density. The thicker color, the more number of times the agent stays at the location. The number above each top-down map shows the total number of steps in this episode.

VI. CONCLUSION

In this paper, we investigated how long-term histories of objects and the first-person views can improve navigation performance in ObjNav task. To this end, we presented Object Memory Transformer (OMT) that utilizes the histories of information that an agent observes during navigation. The proposed method consists of two key components: 1) Object-Scene Memory (OSM) that enables to store long-term scene and object semantics, and 2) Transformer that attends to salient objects in the sequence of previously observed scenes and objects. Our experimental evaluation demonstrated that OMT can achieve state-of-the-art performance on the AI2-THOR benchmark compared against prior works that do not use long-term histories or use them in different manner. Additionally, we showed that just using the long-term histories does not help improve the performance, but combining them with temporal encoding is essential.

REFERENCES

[1] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," *arXiv*, vol. abs/1810.06543, 2018.

[2] R. Druon, Y. Yoshiyasu, A. Kanezaki, and A. Watt, "Visual object search by learning spatial context," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1279–1286, 2020.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[4] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell, "Agent57: Outperforming the Atari human benchmark," in *ICML*, ser. PMLR, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 507–517.

[5] Y. Qiu, A. Pal, and H. Christensen, "Learning hierarchical relationships for object-goal navigation," *ArXiv*, vol. abs/2003.06749, 2020.

[6] H. Du, X. Yu, and L. Zheng, "VTNet: Visual transformer network for object goal navigation," in *ICLR*, 2021.

[7] Y. Lv, N. Xie, Y. Shi, Z. Wang, and H. T. Shen, "Improving target-driven visual navigation with attention on 3d spatial relationships," *arXiv*, vol. abs/2005.02153, 2020.

[8] B. Mayo, T. Hazan, and A. Tal, "Visual navigation with spatial attention," in *CVPR*, 2021.

[9] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *ICML*, no. PART 3, pp. 2347–2355, 2013.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv*, vol. abs/1609.02907, 2016.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Nips*, pp. 5999–6009, 2017.

[12] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of Recurrent Network architectures," *ICML*, vol. 3, pp. 2332–2340, 2015.

[13] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Control of memory, active perception, and action in minecraft," *ICML*, vol. 6, pp. 4067–4089, 2016.

[14] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv*, vol. abs/1410.5401, 2014.

[15] A. Pritzel, B. Uria, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell, "Neural episodic control," *ICML*, vol. 6, pp. 4320–4331, 2017.

[16] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[17] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," 2021.

[18] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *ECCV*, 2020, pp. 213–229.

[19] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," *arXiv*, pp. 1–16, 2018.

[20] A. Khan, C. Zhang, N. Atanasov, K. Karydis, V. Kumar, and D. D. Lee, "Memory augmented control networks," in *ICLR*, 2018.

[21] G. Zhu*, Z. Lin*, G. Yang, and C. Zhang, "Episodic reinforcement learning with associative memory," in *ICLR*, 2020.

[22] K. Fang, A. Toshev, L. Fei-Fei, and S. Savarese, "Scene memory transformer for embodied agents in long-horizon tasks," *CVPR*, vol. 2019-June, pp. 538–547, 2019.

[23] H. Du, X. Yu, and L. Zheng, "Learning Object Relation Graph and Tentative Policy for Visual Navigation," *ECCV*, pp. 19–34, 2020.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CVPR*, pp. 770–778, 2016.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ICLR Workshop*, pp. 1–12, 2013.

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1928–1937.

[27] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *ICRA*, jul 2017, pp. 3357–3364.

[28] X. Ye, Z. Lin, H. Li, S. Zheng, and Y. Yang, "Active Object Perceiver: Recognition-Guided Policy Learning for Object Searching on Mobile Robots," *IROS*, pp. 6857–6863, 2018.

[29] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to learn how to learn: Self-adaptive visual navigation using meta-learning," *2019 IEEE/CVF CVPR*, pp. 6743–6752, 2019.

[30] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv*, 2017.

[31] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. Zamir, "On evaluation of embodied navigation agents," *ArXiv*, vol. abs/1807.06757, 2018.