

MemoNav: Working Memory Model for Visual Navigation

Hongxin Li^{1,2} Zeyu Wang^{1,2} Xu Yang^{1,2} Yuran Yang⁵ Shuqi Mei⁵ Zhaoxiang Zhang^{1,2,3,4}✉

¹University of Chinese Academy of Sciences (UCAS)

²State Key Laboratory of Multimodal Artificial Intelligence Systems, CASIA

³Center for Artificial Intelligence and Robotics, HKISI, CAS

⁴Shanghai Artificial Intelligence Laboratory ⁵Tencent Maps, Tencent

Code: <https://github.com/ZJULiHongxin/MemoNav>

Abstract

Image-goal navigation is a challenging task that requires an agent to navigate to a goal indicated by an image in unfamiliar environments. Existing methods utilizing diverse scene memories suffer from inefficient exploration since they use all historical observations for decision-making without considering the goal-relevant fraction. To address this limitation, we present MemoNav, a novel memory model for image-goal navigation, which utilizes a working memory-inspired pipeline to improve navigation performance. Specifically, we employ three types of navigation memory. The node features on a map are stored in the short-term memory (STM), as these features are dynamically updated. A forgetting module then retains the informative STM fraction to increase efficiency. We also introduce long-term memory (LTM) to learn global scene representations by progressively aggregating STM features. Subsequently, a graph attention module encodes the retained STM and the LTM to generate working memory (WM) which contains the scene features essential for efficient navigation. The synergy among these three memory types boosts navigation performance by enabling the agent to learn and leverage goal-relevant scene features within a topological map. Our evaluation on multi-goal tasks demonstrates that MemoNav significantly outperforms previous methods across all difficulty levels in both Gibson and Matterport3D scenes. Qualitative results further illustrate that MemoNav plans more efficient routes.

1. Introduction

Image-goal navigation (ImageNav) is an attractive embodied AI task where an agent is guided toward a destination indicated by an image within unfamiliar environments. This task has garnered significant attention recently, owing to its promising applications in enabling robots to navigate open-

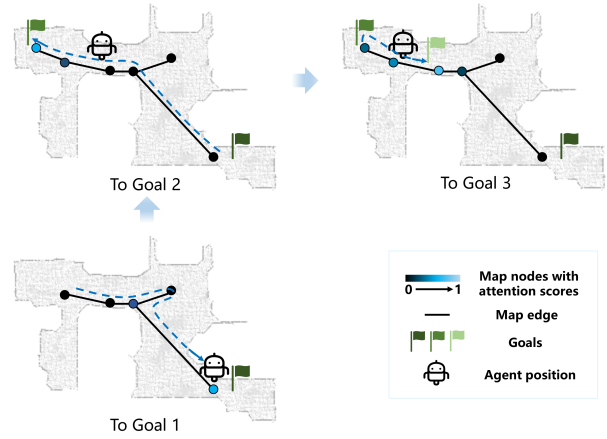


Figure 1. **A brief example of MemoNav.** MemoNav calculates attention scores for each node on the topological map and then excludes the nodes with low scores (the black nodes in the figure) during decision-making. This design helps our agent focus more on goal-relevant scene features, boosting multi-goal visual navigation performance.

world scenarios.

Central to ImageNav is scene memory, which serves as a repository of crucial historical information for decision-making in unseen environments [34]. During navigation, this memory typically stores both scene features and the agent’s navigation history [23], thereby enhancing navigation by mitigating the challenges of partial observability [29]. In literature, various memory mechanisms have been introduced for ImageNav, which can be classified into three categories according to memory structure: (a) metric map-based methods [10, 13] that reconstruct local top-down maps and aggregate them into a global map, (b) stacked memory-based methods [18, 28, 30] that stack the past observations chronologically, and (c) topological map-based methods [5, 11, 22, 23, 34] that store sparse landmark features in graph nodes. Notably, topological map-based meth-

ods leverage the sparsity of topological maps, demonstrating impressive performance in ImageNav.

Nevertheless, existing topological map-based methods still suffer from two major limitations: (a) Unawareness of useful nodes. These methods typically use all node features for generating actions without considering the contribution of each node, thus being easily misled by redundant nodes that are uninformative of the goal. (b) Local representation. Each node feature only represents a small area in a large scene, limiting the agent’s capacity to learn a higher-level semantic and geometric representation of the entire scene.

To address these limitations, we present a novel ImageNav method named MemoNav (refer to Fig. 1), which draws inspiration from the classical concept of working memory in cognitive neuroscience [15] and loosely aligns with the working memory model in human navigation [7].

MemoNav learns three types of scene representations: **Short-term memory (STM)** represents the local and transient features of nodes in a topological map. **Long-term memory (LTM)** represents a global node that acquires a scene-level representation by continuously aggregating STM. **Working memory (WM)** learns goal-relevant features about 3D scenes and is used by a policy network to generate actions. The WM is formed by encoding the informative fraction of the STM and the LTM.

Based on the above three representations, MemoNav navigation pipeline (Fig. 2) contains five steps: (1) STM generation. The map updating module stores landmark features on the map as STM. (2) Selective forgetting. A forgetting module incorporates goal-relevant STM into WM by temporarily removing nodes with attention scores ranking below a predefined threshold. After this process, the navigation pipeline excludes the forgotten nodes in subsequent time steps. (3) LTM generation. To assist STM, a global node is added to the map as LTM. This node links to all map nodes and continuously aggregates their features at each time step. (4) WM generation. A graph attention module encodes the retained STM and LTM to generate WM. The WM combines goal-relevant information from STM with scene-level features from LTM, enhancing the agent’s ability to use informative scene representations for improved navigation. (5) Action generation. Two Transformer decoders use the embeddings of the goal image and the current observation to decode the WM. The decoded features are then used to generate navigation actions.

Consequently, with the synergy of the three representations, MemoNav outperforms state-of-the-art methods in the Gibson scenes [42], enjoying substantial improvements on multi-goal navigation tasks. Comparison in the Matterport3D scenes [9] also highlights MemoNav’s superiority.

The main contributions of this paper are as follows:

- We propose MemoNav, which learns three types of scene representations (STM, LTM, and WM) to improve navigation

performance in the ImageNav task.

- We use a forgetting module to retain informative STM, thereby reducing redundancy in the map and improving navigation efficiency. We also introduce a global node as the LTM, connecting to all STM nodes and providing a global scene-level perspective to the agent.
- We adopt a graph attention module to generate WM from the retained STM and the LTM. This module utilizes adaptive weighting to generate effective WM used for challenging multi-goal navigation.
- The experimental results demonstrate that our method outperforms existing methods on both 1-goal and multi-goal tasks across two popular scene datasets.

2. Related Work

ImageNav methods. Since an early attempt [44] to train agents in a simulator for ImageNav, rapid progress has been made on this task [2, 5, 12, 16, 20, 23, 26, 41, 43]. Several methods have utilized topological scene representations for visual navigation, of which SPTM [34] is an early work. NTS [11], VGM [23], and TSGM [22] incrementally build a topological map during navigation and generalize to unseen environments without exploring the scenes in advance. These methods utilize all features in the map, while our MemoNav flexibly utilizes the informative fraction of these features. Another line of work [27, 43] has introduced self-supervised learning to enhance the scene representations, achieving a promising navigation success rate. In contrast, we enhance the scene representations using the proposed LTM that aggregates the agent’s local observation features.

Memory models for reinforcement learning. Several studies [24, 25, 31, 33, 37, 45] draw inspiration from memory mechanisms of the human brain and design reinforcement learning models for reasoning over long time horizons. Ritter et al. [33] proposed an episodic memory storing state transitions for navigation tasks. Lampinen et al. [24] presented hierarchical attention memory as a form of “mental time-travel” [38], which means recovering goal-oriented information from past experiences. Unlike this method, our model retains such information via a novel forgetting module. Expire-span [37] predicts life spans for each memory fragment and permanently deletes expired ones. Our model is different from this work in that we restore forgotten memory if the agent returns to visited places. [25] shares a similar idea but just solves simple 2D grid-world tasks [14] and its memory capacity is fixed. In contrast, our method employs an adaptive working memory to tackle more complicated long-horizon navigation tasks.

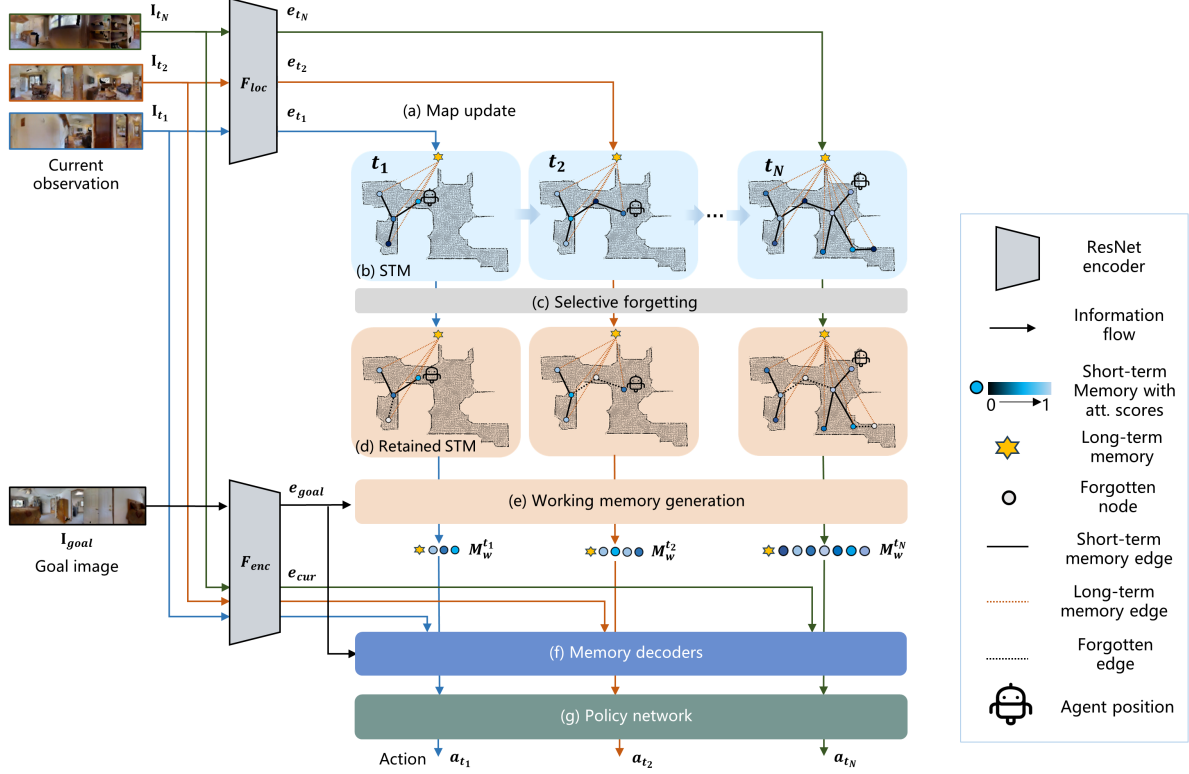


Figure 2. **Overview of MemoNav.** (a) The memory update module builds a topological map using e_t , the embedding of the current image I_t . (b) The node features in the map constitute the STM while a global node that links to each node acts as the LTM. (c) The forgetting module temporarily excludes a fraction of STM whose attention scores rank below a threshold p . (d) The retained STM and the LTM are concatenated and then encoded by (e) a graph attention module to generate the WM M_w^t . (f) The WM is decoded by two Transformer decoders (details in Fig. 7). (g) Lastly, the output of the decoding process is input to a policy network to generate navigation actions.

3. Background

3.1. Task Definition

The objective of ImageNav is to learn a policy π to reach a goal, given an image I_{goal} that contains a view of the goal and a series of observations $\{I_t\}$ captured during the navigation. At the beginning of navigation, the agent receives an RGB image I_{goal} of the goal. At each time step, the agent captures an RGB-D panoramic image I_t of the current location and generates a navigational action. Following [23], any additional sensory data (e.g., GPS and IMU) are not available.

3.2. Brief Review of Visual Graph Memory

Our MemoNav is primarily based on Visual Graph Memory (VGM) [23], which is briefly introduced below. VGM incrementally builds a topological map $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from the agent’s past observations where $\mathcal{V}$ and $\mathcal{E}$ denote nodes and edges, respectively. The node features $V \in \mathbb{R}^{d \times N_t}$ are derived from observations by a pretrained encoder $\mathcal{F}_{loc}$ where d denotes the feature dimension and N_t the number of nodes at time t .

VGM uses a graph convolutional network (GCN) to encode the topological map into a memory representation $M = \text{GCN}(V)$. Before encoding, VGM obtains the goal embedding $e_{goal} = \mathcal{F}_{enc}(I_{goal})$ and fuses each node feature with this embedding through a linear layer.

The encoded memory is then decoded by two Transformer [39] decoders, $\mathcal{D}_{cur}$ and $\mathcal{D}_{goal}$. $\mathcal{D}_{cur}$ takes the current observation embedding $e_{cur} = \mathcal{F}_{enc}(I_{cur})$ as the query and the feature vectors of the encoded memory M as the keys and values, generating a feature vector f_{cur} . Similarly, $\mathcal{D}_{goal}$ takes the goal embedding e_{goal} as the query and generates f_{goal} . Lastly, a LSTM-based policy network takes as input the concatenation of f_{cur} , f_{goal} and e_{cur} to output an action distribution.

4. Method

MemoNav integrates three principal components: the forgetting module, long-term memory generation, and working memory generation. We illustrate the pipeline of the MemoNav in Fig. 2 and detail these components in this section. We also briefly discuss the connection between our method

and working memory studies [4, 15] in the appendix.

4.1. Selective Forgetting Module

MemoNav continually adds nodes to its topological map during exploration. We denote these nodes as short-term memory (STM)¹ as they are dynamically substituted with new ones when the agent revisits corresponding areas.

Our pilot studies indicated that not all STM equally contribute to reaching goals. We visualize the attention scores for the STM calculated in the memory decoder $\mathcal{D}_{goal}$ (Fig. 1 and Fig. 12 in the appendix). The figures show that high scores are assigned to nodes leading to goals while little attention is paid to remote ones. This phenomenon suggests that it is more efficient to use the goal-relevant fraction of scene memory. According to this finding, we devise a forgetting module that enables the agent to forget uninformative experiences. Here, “forgetting” means that STM with attention scores lower than a threshold are temporarily excluded from the navigation pipeline. This means of forgetting via attention is also evidenced by research [19] suggesting that optimal working memory performance depends on focusing on task-relevant information.

The forgetting module retains a fraction of STM according to the attention scores $\{\alpha_i\}_{i=1}^{N_t}$ in $\mathcal{D}_{goal}$. These scores reflect the extent to which the goal embedding e_{goal} attends to each STM feature. After $\mathcal{D}_{goal}$ finishes decoding, the agent temporarily “forgets” a fraction of nodes whose scores rank below a predefined percentage p , meaning these nodes are disconnected from their neighbors and excluded from the navigation pipeline in subsequent steps. If the agent returns to a forgotten node, this node will be re-added to the map and processed by the pipeline again. In multi-goal tasks, once a goal is reached, all forgotten nodes will be restored for potential usefulness in locating the next goal. The forgetting module operates in a plug-and-play manner, which means it is not activated during training but switched on during evaluation and deployment (refer to Fig. 7 for details). p is set as 20% as we empirically find that this suits most tasks. With this module, the agent can selectively retain the informative STM, while avoiding misleading experiences.

4.2. Long-Term Memory Generation

In addition to STM, the information in the long-term memory (LTM) also forms part of working memory (WM) [17]. Inspired by ETC [1] and LongFormer [6], we add a zero-initialized trainable global node $\mathbf{n}_{global} \in \mathbb{R}^d$ to the topological map, representing the LTM (the orange star in Fig. 2), which connects to all nodes in the map and aggregates the STM features at each time step. Unlike

RecBERT [21], which uses a recurrent state token to encode visual-linguistic clues, our LTM aggregates the agent’s past observations by continuously fusing the STM through memory encoding (the encoder is described in the next subsection).

LTM offers two key benefits: it learns a scene-level feature and facilitates feature fusion. A recent study [32] suggests that embodied agents benefit from higher-level environment representations to mitigate partial observability from limited field-of-view sensors. From this viewpoint, the LTM stores a high-level scene representation by aggregating local node features. Moreover, the LTM facilitates feature fusion, especially useful when the topological map is segmented into isolated sub-graphs due to the removal of forgotten nodes. By connecting to every node, the LTM acts as a bypath aiding in feature fusion across these sub-graphs.

4.3. Working Memory Generation

The third type of scene representation WM learns goal-relevant features for action generation. To learn adaptive WM, we utilize a graph attention module GATv2 [8] to encode the retained STM and the LTM, capitalizing on GATv2’s effectiveness in scenarios where nodes have varying neighbor importance. GATv2 adaptively assigns weights to neighboring nodes based on their features, instead of relying on a static Laplacian matrix. This design is suitable for generating WM, especially in multi-goal tasks since the STM features related to a path leading to the goal should obtain high weights while those of irrelevant places should receive lower weights. In addition, as STM features are updated with new features upon revisiting nodes, GATv2’s adaptive weighting is particularly suitable for aggregating the STM features into the WM. The WM generation process is formulated as follows:

$$\mathbf{M}_w = \{\mathbf{V}', \mathbf{n}_{global}^t\} = \text{GATv2}(\{\mathbf{V}, \mathbf{n}_{global}^{t-1}\}) \quad (1)$$

where $\mathbf{M}_w$ represents the generated WM, and $\mathbf{V}'$ the encoded STM. $\{\cdot, \cdot\}$ denotes that the LTM (a vector) is appended to the retained STM (a sequence of vectors). Note that the time step superscript of $\mathbf{n}_{global}$ means that the LTM is recurrent through time.

After GATv2 encoding, the WM aggregates the goal-relevant information from retained STM as well as the scene-level representation from the LTM. Lastly, the decoders $\mathcal{D}_{cur}$ and $\mathcal{D}_{goal}$ take $\mathbf{M}_w$ as keys and values, generating $\mathbf{f}_{cur}$ and $\mathbf{f}_{goal}$, which are further used to generate actions.

5. Experiments

5.1. Datasets

All experiments are conducted in the Habitat [35] simulator with the Gibson [42] and Matterpot3D [9] scene dataset,

¹The two terminologies “Nodes” and “STM” will be used interchangeably in the following sections.

Scene	Methods	1-goal		2-goal		3-goal		4-goal	
		SR	SPL	PR	PPL	PR	PPL	PR	PPL
G	ANS [10]	30.0	11.0	-	-	-	-	-	-
	NTS [11]	43.0	26.0	-	-	-	-	-	-
	CNNLSTM [44]	53.1	39.2	31.5	10.6	18.0	2.8	12.4	1.6
	TSGM [22]	70.3	50.0	27.8	16.1	17.4	10.4	13.4	4.6
	VGM [23]	70.0	55.4	42.9	17.1	29.5	7.0	21.5	4.1
	MemoNav (ours)	74.7	57.9	50.8	20.1	38.0	9.0	28.9	5.1
M	CNNLSTM [44]	16.2	9.8	10.8	2.6	7.7	1.4	-	-
	TSGM [22]	24.0	14.6	13.5	6.2	7.8	3.8	-	-
	VGM [23]	25.1	16.6	16.7	5.0	11.8	2.5	-	-
	MemoNav (ours)	26.1	16.3	19.5	5.6	13.6	2.9	-	-

Table 1. **Comparison between MemoNav and previous methods.** The evaluation results in Gibson (G) and Matterport3D (M) scenes demonstrate that MemoNav outperforms previous methods across all difficulty levels. Note that the 1-goal evaluation in Gibson uses 1007 hard episodes following [23] while the multi-goal evaluation uses our collected episodes. **SR**: success rate (%), **SPL**: success weighted by path length (%), **PR**: progress (%), **PPL**: progress weighted by path length (%).

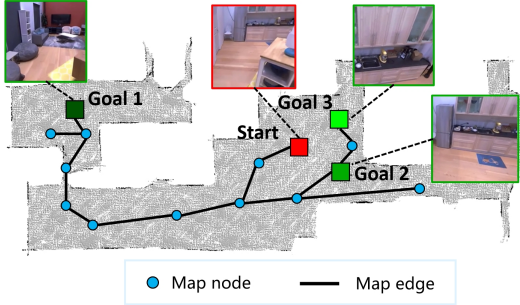


Figure 3. **An example episode for multi-goal tasks in Gibson.** The agent is tasked with navigating to multiple sequential goals.

adopting an action space consistent with VGM [23].

1-goal evaluation. In Gibson, we use 72 scenes for training and a public dataset [28] comprising 14 unseen scenes for evaluation. Following the setting in VGM [23], 1007 out of 1400 1-goal episodes² from this public dataset are used for evaluation, while we still use the full set of 1400 episodes for ablation studies of MemoNav.

Multi-goal dataset. Multi-goal evaluation, which requires an agent to navigate to an ordered sequence of goals, is more suitable for evaluating memory models used for navigation. By enabling the agent to return to visited places we can test whether memory models help the agent plan efficient paths. If not, the agent will probably waste its time re-exploring the scene or traveling randomly. However, recent ImageNav methods seldom conduct multi-goal evaluations. To further investigate the efficacy of MemoNav, we follow MultiON [40] to compile 700-episode multi-goal test

datasets in the Gibson scenes (see Fig. 3 for an example).

We follow five rules to set sequential goals for each episode: (1) No obstacles appear near each goal. (2) The distance between two successive goals is no more than 10 meters. (3) All goals are placed on the same layer. (4) All goals are reachable from each other. (5) The final goal is placed near a certain previous one. Please refer to Fig. 8 for dataset statistics.

In Matterpot3D, we sample 1008 episodes per difficulty level from the multi-goal test datasets used in Multi-ON [40]. The difficulty of an episode is indicated by the number of goals. All methods are trained on the Gibson 1-goal dataset and tested across varying difficulty levels.

Evaluation Metrics. In 1-goal tasks, the success rate (**SR**) and success weighted by path length (**SPL**) [3] are used. In a multi-goal task, two metrics are borrowed from [40]: The progress (**PR**) is the fraction of goals successfully reached, equal to the **SR** for 1-goal tasks; Progress weighted by path length (**PPL**) indicates navigation efficiency and is defined

as
$$PPL = \frac{1}{E} \sum_{i=1}^E Progress_i \frac{l_i}{\max(p_i, l_i)}$$
, where E is the total number of test episodes, l_i and p_i are the shortest path distance to the final goal via midway ones, and the actual path length taken by the agent, respectively. The objective of each goal is to stop within 1 meter of the goal location and each episode is allowed 500 steps.

5.2. Compared Methods and Training Details

We compare with the following methods which adopt various memory types: **CNNLSTM** [44] uses no maps but a hidden vector as implicit memory. **ANS** [10] is a metric map-based model for ImageNav. **NTS** [11] incrementally builds a topological map without pre-exploring and adopts

²The 1-goal difficulty level here denotes the hard level in this public test dataset

	Components			1-goal		2-goal		3-goal		4-goal	
	Forget	LTM	WM	SR	SPL	PR	PPL	PR	PPL	PR	PPL
1				52.1	46.7	42.9	17.1	29.5	7.0	21.5	4.1
2	✓			55.1	46.1	44.9	17.5	29.4	6.5	21.5	4.2
3		✓		58.9	49.7	43.8	17.8	29.6	6.9	25.1	4.0
4	✓	✓		60.6	49.9	48.1	19.5	37.5	9.1	28.8	4.9
5		✓	✓	61.1	48.9	47.6	17.8	33.7	7.9	27.4	5.0
6	✓	✓	✓	62.4	50.7	50.8	20.1	38.0	9.0	28.9	5.1

Table 2. **Network component ablation results.** This ablation uses the entire 1400 1-goal episodes collected by [28] and our collected multi-goal evaluation datasets. Row 1 is the baseline model VGM [23], and row 6 is our full model. The table shows that when applied separately, the forgetting module and the LTM both improves performance and that the combination of these two components brings larger gains. Moreover, the synergy among the three components leads to the best performance. (**Forget**: Forgetting nodes with attention scores below 20%, **LTM**: Using the LTM to continuously fuse the STM, **WM**: Using GATv2 to learn adaptive working memory)

a hierarchical navigation strategy. **VGM** [23] is the baseline for MemoNav and has been elaborated in Sec. 3.2. **TSGM** [23] associates a topological map with detected objects to use more semantic scene information.

We follow the training pipeline in [23] to reproduce CNNLSTM and train our MemoNav. These methods are first trained via imitation learning and then further fine-tuned with proximal policy optimization (PPO) [36] (details in the appendix). The evaluation results for ANS and NTS are borrowed from the VGM paper [23]³. All methods are equipped with a panoramic camera.

5.3. Quantitative Results

Comparison on Gibson. Tab. 1 shows that the MemoNav outperforms all compared methods in SR across all difficulty levels. Notably, CNNLSTM exhibits the poorest performance as its limited memory provides insufficient scene information. MemoNav also outperforms the metric map-based method ANS which requires pre-built maps. Compared with VGM, our model exhibits a noticeable performance gain, especially on the multi-goal tasks, enhancing SR by **7.9%**, **8.5%**, and **7.4%** on the 2, 3, and 4-goal tasks respectively, while relying on less scene memory.

Comparison on Matterport3D. We extend our evaluation to the Matterport3D scenes to assess the models’ ability to generalize to different scene types. Tab. 1 shows that our method achieves consistent performance improvements on this unseen scene dataset. Compared with VGM, MemoNav demonstrates higher SR/PR across the three difficulty levels. TSGM obtains slightly better PPL on multi-goal tasks probably because it uses more object-level clues to locate the goal. As the introduction of object semantics in TSGM is orthogonal to our contributions, we believe adding these semantics to MemoNav will lead to higher performance.

Overall, these results demonstrate that MemoNav benefits from the informative scene memory and the high-level scene representation contained in the WM, obtaining high success rates.

5.4. Ablation Studies and Analysis

We conduct ablation studies in the Gibson scenes to analyze the impact of each proposed component.

Performance gain from each proposed component. We assess the three key components outlined in Sec. 4 and present the results in Tab. 2. We can see that applying the forgetting module achieves improvements in the SR/PR (row 2 vs. row 1) and that the LTM also brings noticeable gains in SR/PR over the baseline (row 3 vs. row 1). Notably, the combined use of the forgetting module and LTM results in even larger increases (row 4 vs. row 1). More importantly, the synergy among the three components increases the SR/PR by **10.3%**, **7.9%**, **8.5%**, and **7.4%** at the 1, 2, 3, and 4-goal levels, respectively (row 6 vs. row 1). Comparing rows 6 and 5, we see that adding the forgetting module to MemoNav leads to notable increases, especially on multi-goal tasks. The qualitative evaluation in Fig. 9 also shows that MemoNav without forgetting tends to take redundant steps, which also justifies the efficacy of the forgetting module. Overall, these results underscore the effectiveness of our components in addressing long-horizon navigation tasks with multiple sequential goals.

The Critical Role of LTM. To study the effect of the LTM proposed in Sec. 4.2, we implement three distinct ways of disabling the function of the LTM and show the results in Tab. 3. The first ablation (row 2) excludes the LTM from the WM, preventing the high-level scene feature from being utilized by the downstream policy network. This modification worsens the performance across all difficulty levels. The second ablation replaces the LTM feature with a randomly selected STM feature each time the WM is generated according to Eq. (1), resulting in inferior multi-goal naviga-

³ANS is designed for exploration while NTS is not open-sourced, so it is not straightforward to reproduce them for multi-goal tasks.

	Variants	1-goal		2-goal		3-goal		4-goal	
		SR	SPL	PR	PPL	PR	PPL	PR	PPL
1	MemoNav	62.4	50.7	50.8	20.1	38.0	9.0	28.9	5.1
2	LTM excluded	60.5	49.1	46.3	16.3	34.2	8.0	26.8	4.9
3	Random replacing	61.0	47.9	45.9	17.4	34.9	8.2	27.0	5.0
4	Averaging STM as LTM	59.1	49.2	46.1	18.6	37.4	8.3	28.4	4.8

Table 3. **Ablation study of LTM.** Row 1 is our default model. Row 2 is a variant using the LTM to aggregates STM but does not incorporate the LTM into WM. Row 3 shows the impact of replacing the LTM with a random STM feature. Row 4 shows the result of replacing the LTM with averaged STM. The three variants disable the LTM, all leading to performance drops.

tion performances.

To justify that aggregating STM features into the LTM using adaptive weighting helps to learn a better high-level scene representation, we replace the LTM feature with an average of all STM features (row 4). After replacing, the LTM no longer learns a scene-level representation but still facilitates message passing among STM. Although this variant exhibits a decline in performance, the decrease is less pronounced for the more challenging 3 and 4-goal tasks. We hypothesize that this is because the simply averaged STM features act as a rudimentary scene-level feature but possess a weak representation capability compared to a learned LTM feature.

Overall, these findings confirm the LTM’s vital role in learning scene-level features that are crucial for improving navigation performance.

Correlation between navigation performance and forgetting threshold. We evaluate our model with different forgetting thresholds p , as defined in Sec. 4.1. The results are shown in Fig. 4. For clarity, the figure shows the performance differences between our full model and the variants. The figure in general shows that MemoNav achieves the best performance on easier tasks with a lower p but requires a higher p for peak performance on harder tasks. Specifically, increasing p from 0.0 (no forgetting) to 0.8 leads to initial improvements in SR/PR and SPL/PPL for 1 and 2-goal tasks, followed by a decline. As the agent seldom revisits explored areas in these easier tasks, using a larger p to remove too much STM leads to a situation where the agent forgets what it has explored and takes more steps to re-explore the scene.

In contrast, for the more demanding 3 and 4-goal tasks, a higher p is more helpful. For instance, at $p = 0.8$, MemoNav exhibits only slight drops in SR/PR and SPL/PPL on 3-goal tasks, indicating that our agent is able to maintain a high success rate with just 20% of STM. Notably, MemoNav achieves top performance in 4-goal tasks at $p = 0.4$, suggesting that a larger portion of the STM can be forgotten when conducting longer-horizon navigation tasks that require an agent to frequently revisit explored places. The rationale behind this phenomenon is that if an agent has ex-

plored most of the scene after finishing several goals, it is supposed to plan shorter paths to subsequent goals by utilizing a small, goal-relevant fraction of STM.

We have also conducted an in-depth analysis of this phenomenon by plotting distributions of distances from retained/forgotten STM to each goal, as shown in Fig. 11 of the appendix. This analysis demonstrates that a substantial portion of retained nodes align closely with the shortest paths, indicating that MemoNav leverages a higher p to focus more on the regions along short paths in multi-goal navigation scenarios.

5.5. Qualitative Comparison

To qualitatively assess MemoNav, we show example episodes of CNLSTM, VGM, and MemoNav in the Gibson scenes in Fig. 5. MemoNav’s efficacy is evident in its shorter and smoother trajectories. Conversely, CNLSTM exhibits extensive exploration steps and often fails in complex scenes due to its simplistic hidden states which struggle to encapsulate an effective scene memory. VGM frequently navigates in redundant circles, particularly in narrow pathways. For instance, as depicted in the second and third columns, MemoNav adeptly avoids the bottlenecks that entrap VGM, exemplifying MemoNav’s capabilities of planning efficient routes.

5.6. Visualization of MemoNav

We plot example trajectories of MemoNav in multi-goal tasks, as shown in Fig. 12. These examples show that MemoNav utilizes the adaptive WM to plan efficient paths in challenging multi-goal navigation. For instance, in the 3-goal trajectory, MemoNav strategically forgets the distant topmost node during the first goal and similarly neglects the goal-irrelevant bottom-left nodes in subsequent goals.

To provide a comprehensive overview, we also analyze MemoNav’s limitations by examining its failure episodes in Fig. 13 of the appendix. These failures predominantly fall into four categories: *Stopping mistakenly*, *Missing the goal*, *Not close enough*, and *Over-exploring*. This analysis not only helps in understanding the limitations of MemoNav but also guides potential improvements in future iterations.

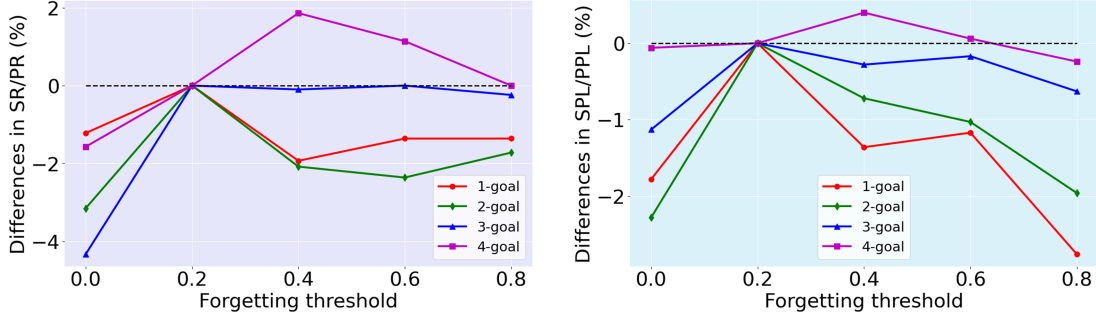


Figure 4. **Navigation performance versus forgetting threshold p in the Gibson scenes.** MemoNav achieves the best performance on easier tasks with a lower p but a higher p is more beneficial for harder tasks. Moreover, MemoNav maintains high SR/PR with just 20% of STM on the 3-goal tasks and enjoys a higher p on the 4-goal tasks.

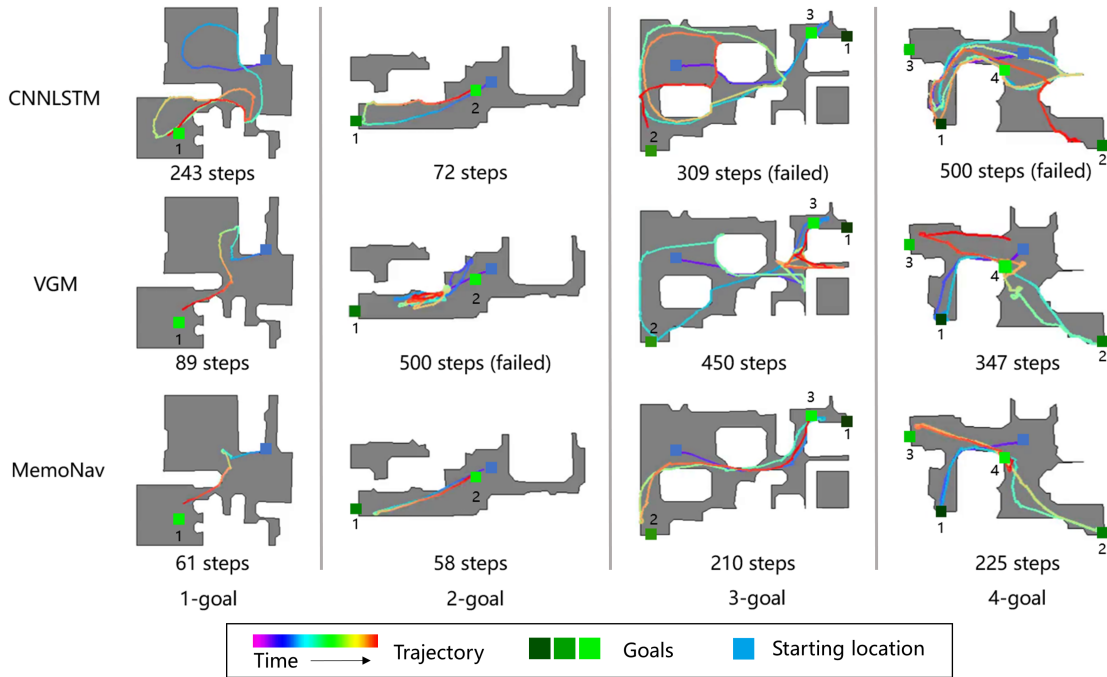


Figure 5. **Visualization of example episodes from a top-down view.** We compare CNNLSTM, VGM, and MemoNav at four difficulty levels in the Gibson scenes. Our MemoNav plans more efficient paths compared to the other two methods. For instance, in the 3-goal example, MemoNav quickly reaches the third goal which is located at an explored area. Best viewed in color.

6. Conclusion

This paper proposes MemoNav, a novel memory model for ImageNav. This model flexibly retains informative short-term navigation memory via a forgetting module. We also introduce an extra global node as long-term memory to learn a scene-level representation. The retained short-term memory and the long-term memory are encoded by a graph attention module to generate the working memory that is used for generating action. The experimental results show that the MemoNav outperforms previous methods in multi-goal tasks and plans more efficient routes.

7. Acknowledgments

This work was supported in part by the National Key R&D Program of China (No. 2022ZD0160102, No. 2022ZD0160100), the National Natural Science Foundation of China (No. U21B2042, No. 62072457), and in part by the 2035 Innovation Program of CAS.

References

- [1] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding

- long and structured inputs in transformers. In *EMNLP*, pages 268–284, 2020. 4
- [2] Ziad Al-Halah, Santhosh Kumar Ramakrishnan, and Kristen Grauman. Zero experience required: Plug & play modular transfer learning for semantic visual navigation. In *CVPR*, pages 17031–17041, 2022. 2
- [3] 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. 5
- [4] Alan Baddeley. Working memory: theories, models, and controversies. *Annual review of psychology*, 63:1–29, 2012. 4, 1
- [5] Edward Beeching, Jilles Dibangoye, Olivier Simonin, and Christian Wolf. Learning to plan with uncertain topological maps. In *ECCV*, pages 1–24, 2020. 1, 2
- [6] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. 4
- [7] Kara J Blacker, Steven M Weisberg, Nora S Newcombe, and Susan M Courtney. Keeping track of where we are: Spatial working memory in navigation. *Visual Cognition*, 25(7-8): 691–702, 2017. 2
- [8] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *ICLR*, 2022. 4
- [9] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, pages 667–676. IEEE, 2017. 2, 4
- [10] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020. 1, 5
- [11] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, pages 12875–12884, 2020. 1, 2, 5
- [12] Kevin Chen, Junshen K Chen, Jo Chuang, Marynel Vázquez, and Silvio Savarese. Topological planning with transformers for vision-and-language navigation. In *CVPR*, pages 11276–11286, 2021. 2
- [13] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *ICLR*, 2019. 1
- [14] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In *ICLR*, 2019. 2
- [15] Nelson Cowan. What are the differences between long-term, short-term, and working memory? *Progress in brain research*, 169:323–338, 2008. 2, 4, 1
- [16] Yilun Du, Chuang Gan, and Phillip Isola. Curious representation learning for embodied intelligence. In *CVPR*, pages 10408–10417, 2021. 2
- [17] K Anders Ericsson and Walter Kintsch. Long-term working memory. *Psychological review*, 102(2):211, 1995. 4
- [18] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *CVPR*, pages 538–547, 2019. 1
- [19] Keisuke Fukuda and Edward K Vogel. Human variation in overriding attentional capture. *Journal of Neuroscience*, 29(27):8726–8733, 2009. 4
- [20] Meera Hahn, Devendra Singh Chaplot, Shubham Tulsiani, Mustafa Mukadam, James M Rehg, and Abhinav Gupta. No rl, no simulation: Learning to navigate without navigating. *NeurIPS*, 34:26661–26673, 2021. 2
- [21] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. Vln bert: A recurrent vision-and-language bert for navigation. In *CVPR*, pages 1643–1653, 2021. 4
- [22] Nuri Kim, Obin Kwon, Hwiyeon Yoo, Yunho Choi, Jeongho Park, and Songhwai Oh. Topological semantic graph memory for image-goal navigation. In *CoRL*, pages 393–402. PMLR, 2023. 1, 2, 5
- [23] Obin Kwon, Nuri Kim, Yunho Choi, Hwiyeon Yoo, Jeongho Park, and Songhwai Oh. Visual graph memory with unsupervised representation for visual navigation. In *CVPR*, pages 15890–15899, 2021. 1, 2, 3, 5, 6
- [24] Andrew Lampinen, Stephanie Chan, Andrea Banino, and Felix Hill. Towards mental time travel: a hierarchical memory for reinforcement learning agents. *NeurIPS*, 34, 2021. 2
- [25] Ricky Loynd, Roland Fernandez, Asli Celikyilmaz, Adith Swaminathan, and Matthew Hausknecht. Working memory graphs. In *ICML*, pages 6404–6414. PMLR, 2020. 2
- [26] Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. *NeurIPS*, 35: 32340–32352, 2022. 2
- [27] Arjun Majumdar, Gunnar A Sigurdsson, Robinson Piramuthu, Jesse Thomason, Dhruv Batra, and Gaurav S Sukhatme. Ssl enables learning from sparse rewards in image-goal navigation. In *ICML*, pages 14774–14785. PMLR, 2022. 2
- [28] Lina Mezghani, Sainbayar Sukhbaatar, Thibaut Lavril, Oleksandr Maksymets, Dhruv Batra, Piotr Bojanowski, and Alahari Karteek. Memory-augmented reinforcement learning for image-goal navigation. *IROS*, pages 3316–3323, 2021. 1, 5, 6
- [29] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *ICLR*, 2018. 1
- [30] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. In *CVPR*, pages 15942–15952, 2021. 1
- [31] Hong Qiao, Yaxiong Wu, Shanlin Zhong, Peijie Yin, and Jiahao Chen. Brain-inspired intelligent robotics: Theoretical analysis and systematic application. *Machine Intelligence Research*, 20:1–18, 2023. 2
- [32] Santhosh Kumar Ramakrishnan, Tushar Nagarajan, Ziad Al-Halah, and Kristen Grauman. Environment predictive coding for visual navigation. In *ICLR*, 2022. 4
- [33] Samuel Ritter, Ryan Faulkner, Laurent Sartran, Adam Santoro, Matthew Botvinick, and David Raposo. Rapid task-solving in novel environments. In *ICLR*, 2021. 2

- [34] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018. 1, 2
- [35] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *CVPR*, pages 9339–9347, 2019. 4
- [36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 6, 1
- [37] Sainbayar Sukhbaatar, Da Ju, Spencer Poff, Stephen Roller, Arthur Szlam, Jason Weston, and Angela Fan. Not all memories are created equal: Learning to forget by expiring. In *ICML*, pages 9902–9912. PMLR, 2021. 2, 4
- [38] Endel Tulving. Memory and consciousness. *Canadian Psychology/Psychologie canadienne*, 26(1):1, 1985. 2
- [39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017. 3
- [40] Saim Wani, Shivansh Patel, Unnat Jain, Angel Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. *NeurIPS*, 33:9700–9712, 2020. 5
- [41] Justin Wasserman, Karmesh Yadav, Girish Chowdhary, Abhinav Gupta, and Unnat Jain. Last-mile embodied visual navigation. In *CoRL*, 2022. 2
- [42] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, pages 9068–9079, 2018. 2, 4
- [43] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. In *Workshop on Reincarnating Reinforcement Learning at ICLR*, 2023. 2
- [44] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364. IEEE, 2017. 2, 5, 1, 3
- [45] Xiaolong Zou, Tie-Jun Huang, and Si Wu. Towards a new paradigm for brain-inspired computer vision. *Machine Intelligence Research*, 19:412 – 424, 2022. 2

MemoNav: Working Memory Model for Visual Navigation

Supplementary Material

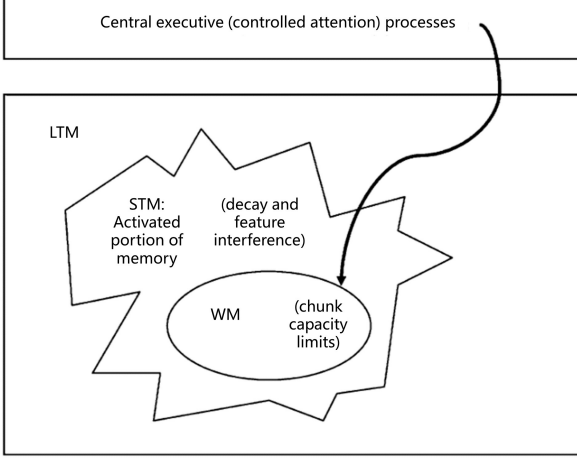


Figure 6. The memory model by Cowan et al. [15]. This figure is borrowed and adapted from its original paper.

8. Relation between MemoNav and Representative Working Memory Models

Human memory consists of complex interactions between long-term memory (LTM), short-term memory (STM), and working memory [15]. As defined by Cowan et al. [15], LTM refers to the vast, stable knowledge base and experiences stored over a lifetime. STM is a transient, limited-capacity memory system that holds information in an accessible state for brief periods. Working memory incorporates selective parts of STM as well as stored LTM knowledge through an attention mechanism, in order to actively process information relevant to the current task or decision. Cowan et al. [15] also designed a framework depicting how WM is formed from STM and LTM (shown in Fig. 6). This framework demonstrates that STM cooperates with LTM and decays as a function of time unless it is refreshed. The useful fraction of STM is incorporated into WM via an attention mechanism to avoid misleading distractions. Subsequent work by Baddeley et al. [4] suggests that the central executive manipulates memory by incorporating not only part of STM but also part of LTM to assist in making a decision.

We draw inspiration from the work by Cowan et al. [15] and Baddeley et al. [4] and formulate the navigation memory of MemoNav as an emulation of the human STM, LTM and working memory systems.

The parallel between MemoNav and the two relevant models above is shown in the following list:

- The map node features are termed “STM”, since they are local and transient.

- The topological map of MemoNav maintains a 100-node queue to store map nodes. This design simulates STM that holds a limited amount of information in a very accessible state temporarily in the human brain.
- MemoNav introduces a global node aggregating prior observation features stored in the topological map, thereby simulating LTM which acts as a large knowledge base.
- MemoNav utilizes a forgetting mechanism to remove a fraction of STM with attention scores lower than a threshold. This mechanism acts as a simple way of decaying STM.
- The forgetting mechanism helps WM include part of STM.
- MemoNav incorporates the retained STM and the LTM into WM, which is subsequently used to generate navigation actions. This design simulates the working memory model by [4].

9. Implementation Details

9.1. Implementation of MemoNav

Built upon VGM, MemoNav inherits its topological map and uses its localization approach to add nodes. In addition, MemoNav improves the memory module while keeping the visual encoder and policy network unchanged of VGM.

We follow the training pipeline in [23] to reproduce CNNLSTM and train our MemoNav. These methods are first trained via imitation learning, minimizing the negative log-likelihood of ground-truth actions. Next, the agents are further fine-tuned with PPO [36] to enhance exploratory ability. The reward setting and auxiliary losses remain the same as in VGM. The reward setting and auxiliary losses remain the same as in VGM.

The detailed MemoNav framework is shown in Fig. 7. The structure of the memory decoding module in MemoNav remains the same as in VGM [23]. The forgetting module of MemoNav requires the attention scores generated in the decoder $\mathcal{D}_{goal}$. Therefore, our model needs to calculate the whole navigation pipeline before deciding which fraction of the STM should be retained. This lag means that the retained STM is incorporated into the WM at the next time step. The pseudo-code of MemoNav is shown in Algorithm 1.

9.2. Reproduction of CNNLSTM

We reproduce CNNLSTM [44] following the description in its original paper, but we also make some modifications to keep the comparison fair. We replace the ResNet-50 in CNNLSTM with the pretrained RGB-D encoder of

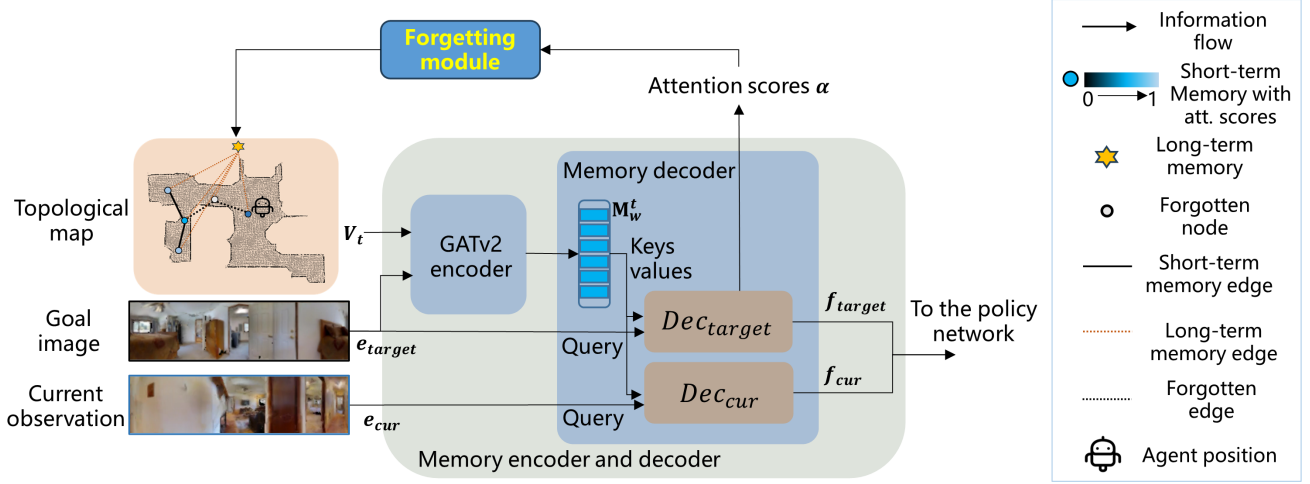


Figure 7. **The detailed structure of MemoNav.** The goal decoder $\mathcal{D}_{target}$ calculates the attention scores α for each STM feature in the topological map. Then the scores are used by the proposed forgetting module to remove redundant STM which will no longer be utilized for downstream action generation. V denotes the retained STM and M_w the working memory.

Algorithm 1: The implementation of the MemoNav

Data: Empty topological map $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, goal image $\mathbf{l}_{goal}$, current time step t , forgetting percentage p , trainable observation encoder $\mathcal{F}_{enc}$, GATv2-based encoder GATv2, Transformer decoders $\mathcal{D}_{goal}$ and $\mathcal{D}_{cur}$, LSTM-based policy network LSTM

Result: Navigation action a_t

```

1 Long-term memory  $\mathbf{n}_{global} \leftarrow \mathbf{0} \in \mathbb{R}^d$ ;
2 Attention scores for graph nodes  $V$ :  $\alpha \leftarrow \mathbf{0} \in \mathbb{R}^{|\mathcal{V}|}$ ;
3 while not AgentCallStop() do
    // Step 1: Update the topological map
4  $\mathbf{l}_t \leftarrow \text{GetCurrentPanorama}()$ ;
5  $G.\text{UpdateMap}(\mathbf{l}_t)$ ;
    // Step 2: Retain the informative fraction of the STM
6 Forgotten number  $n \leftarrow \text{Floor}(p \cdot |\mathcal{V}|)$ ;
7 Sorted indices  $i \leftarrow \text{Argsort}(\alpha)$ ;
8 Forgotten indices  $i_{forgotten} \leftarrow i[0:n]$ ;
9  $G.\text{RemoveNodes}(i_{forgotten})$ ;
10  $V \in \mathbb{R}^{|\mathcal{V}| \times d} \leftarrow G.\text{GetNodeFeatures}()$ ;
11 Working memory  $M_w \leftarrow \text{GATv2}(\{V, \mathbf{n}_{global}\})$  // Note that STM is fused before being
    forgotten in the next step so the features of forgotten STM have been
    fused into LTM.
12  $e_{cur} \leftarrow \mathcal{F}_{enc}(\mathbf{l}_t)$ ,  $e_{goal} \leftarrow \mathcal{F}_{enc}(\mathbf{l}_{goal})$ ;
13  $f_{cur} \leftarrow \mathcal{D}_{cur}(e_{cur}, M_w)$ ,  $f_{goal} \leftarrow \mathcal{D}_{goal}(e_{goal}, M_w)$ ;
14  $\alpha \leftarrow \mathcal{D}_{goal}.\text{GetAttScores}()$ 
    // Step 4: Action generation
15  $x \leftarrow \text{LSTM}(\text{FC}([f_{cur}, f_{goal}, e_{cur}]))$ ;
16  $p(a_t | x) = \sigma(\text{FC}(x))$ ;
17  $a_t \leftarrow \text{SampleFromDistribution}(p(a_t | x))$ ;
18 end

```

VGM [23]. We also add positional embeddings to the encoded RGB-D observations to contain temporal informa-

tion. Moreover, we concatenate the encoded RGB-D observations with the goal image embedding and project the

concatenated feature (1024D) to a 512D feature, so CNLSTM can utilize the information of the goal image. The projected features of four consecutive frames are further condensed and then input to a policy network as in [44]. To use the two auxiliary tasks proposed in VGM [23], we also introduce the linear projection layers (Linear-ReLU-Linear) used in VGM to process the embedded goal image and embedded current observation.

9.3. Compute Requirements

We utilize an RTX TITAN GPU for training and evaluating our models. The imitation learning phase takes 1.5 days to train while the reinforcement learning takes 5 days.

The computation in the GATv2-based encoder and the two Transformer decoders occupy the largest proportion of the run-time of MemoNav. The computation complexity of the encoder and the decoders are $\mathcal{O}(|\mathcal{V}|d^2 + |\mathcal{E}|d)$ and $\mathcal{O}(|\mathcal{V}|d)$, respectively. Using the forgetting module with a percentage threshold p , the computation complexity of MemoNav can be flexibly decreased by reducing the number of nodes to $(1 - p)|\mathcal{V}|$.

10. Comparison between MemoNav with and without Forgetting

We analyze the impact of the forgetting module on MemoNav’s trajectory properties, such as smoothness and length. Fig. 9 illustrates that the inclusion of the forgetting module results in more smooth and efficient trajectories. In contrast, trajectories generated without this module are characterized by numerous abrupt turns and extended paths. This disparity likely arises from a segment of the Short-Term Memory (STM) containing irrelevant information, leading to frequent and erratic alterations in the policy network’s action output. The forgetting module effectively filters out this disruptive portion of STM, thereby enabling the policy network to use task-relevant navigation memory for efficient decision-making.

11. In-depth Analysis of Forgetting Module

An extensive statistical analysis is conducted to comprehend the forgetting module’s functionality. In this experiment, five distance metrics are calculated: (a) distance from a node to the **agent**, (b) distance from a node to the **goal**, (c) distance from a node to **the oracle shortest path**, (d) distance from a node to the shortest path segments closer to the **agent**, and (e) distance from a node to the shortest path segments closer to the **current goal**. Then the histograms of these five metrics are drawn according to the metrics records for each forgotten/retained node at each time step so we can see the patterns of these distance metrics. Please see Fig. 10 to better understand the definitions of the distance metrics (c)(d)(e).

We evaluate MemoNav on the 3-goal Gibson task and draw the histograms **on per-goal basis**, as shown in Fig. 11. The figure provides two interesting findings:

- The distance distribution patterns for forgotten nodes (green bars) and retained ones (orange bars) vary across goals. Notably, as the agent progresses to the third goal, the distributions of the distances from forgotten nodes to goals (column 2) and to shortest path segments near goal (column 5) become uniform. In contrast, these two histograms for the retained nodes become sharper and the peaks shift to smaller distance values. This pattern suggests the forgetting module selectively retain nodes that are proximal and relevant to the current goal.
- The forgetting module has a larger impact on the distance metrics when the navigation task becomes more difficult. Specifically, when the current goal index is 1 (i.e. the task is easy), the averages of the distance metrics for forgotten nodes and retained nodes are close. When the goal index rises to 3 (i.e. the task becomes harder), a larger proportion of the retained nodes are close to the goal, the shortest path, and the shortest path segments near goal. This pattern suggests that MemoNav focuses on critical areas for navigation, such as the goal vicinity and the shortest path.

These results empirically validate that MemoNav is able to retain the information useful for multi-goal navigation via the forgetting module.

12. The Variation of the LTM

We explore the dynamic nature of the LTM during navigation by calculating the L2 distance between consecutive time-step features, as depicted in Fig. 14. The trends observed in these curves – rapid initial increases in L2 difference followed by stabilization and intermittent peaks – are indicative of the LTM’s response to the agent’s environmental interactions.

To understand why the LTM variation shows such a trend, we visualize the agent’s observations at the time steps of the peaks. Specifically, the L2 difference remains low in familiar areas, suggesting stability in the LTM’s feature representation. For instance, in the 2-goal example (top row), the L2 difference steadily decreases in $t = 31 \sim 55$ during which the agent travels around visited areas; (2) The L2 difference increases sharply upon encountering new scenes. These peaks correspond with the agent’s exposure to novel views. For instance, in the 3-goal example (bottom row), the L2 difference curve exhibits peaks at $t = 68$ when the agent passes a corner and at $t = 88$ when the agent observes a novel open area. These results highlighting the LTM’s role in assimilating new exploratory experiences.

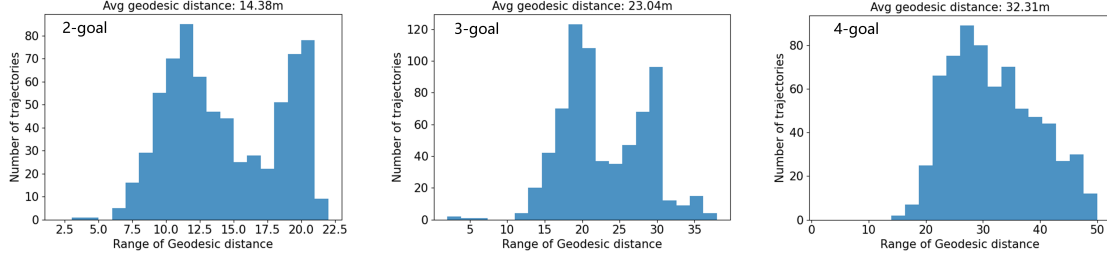


Figure 8. Histograms of geodesic distances for the multi-goal test datasets. As we set a distance limit for goals and discard invalid trajectories, a scene may own its prominent distance range, leading to the nonuniform histograms.

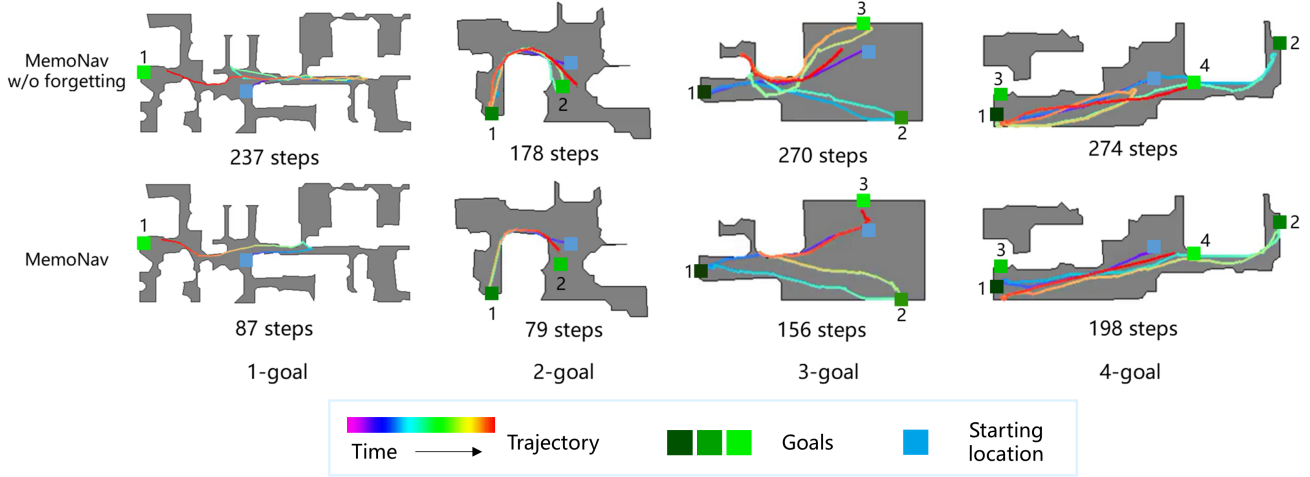


Figure 9. **Visualization comparing the MemoNav with and without the forgetting module.** We compare selected episodes at four difficulty levels in the Gibson scenes and visualize the top-down views. MemoNav without the forgetting module exhibits more sharp turns and tends to take more steps, demonstrating lower efficiency compared to the full MemoNav. The number of navigation steps (the upper limit is 500) are shown at the bottom of each top-down view. Best viewed in color.

13. Limitations

While MemoNav witnesses a large improvement in the navigation success rate in multi-goal navigation tasks, it still encounters limitations. The proposed forgetting module is a post-processing method, as it obtains the attention scores of the decoder before deciding which nodes are to be forgotten. Future work can explore trainable forgetting modules. The second limitation is that our forgetting module does not reduce memory footprint, since the features of the forgotten nodes still exist in the map for localization. Moreover, the forgetting threshold in our experiments is fixed. Future work can merge our idea with Expire-span [37] to learn an adaptive forgetting threshold.

14. Potential Impact

The notable potential of negative societal impact from this work: our model is trained on 3D scans of the Gibson scenes which only contain western styles. This inadequacy of diverse scene styles may render our model biased and in-

compatible with indoor environments in unseen styles. As a result, our model may be only available in a small fraction of real-life scenes. If our model is transferred to out-of-distribution scenes, the agent may take more steps and even bump on walls frequently.

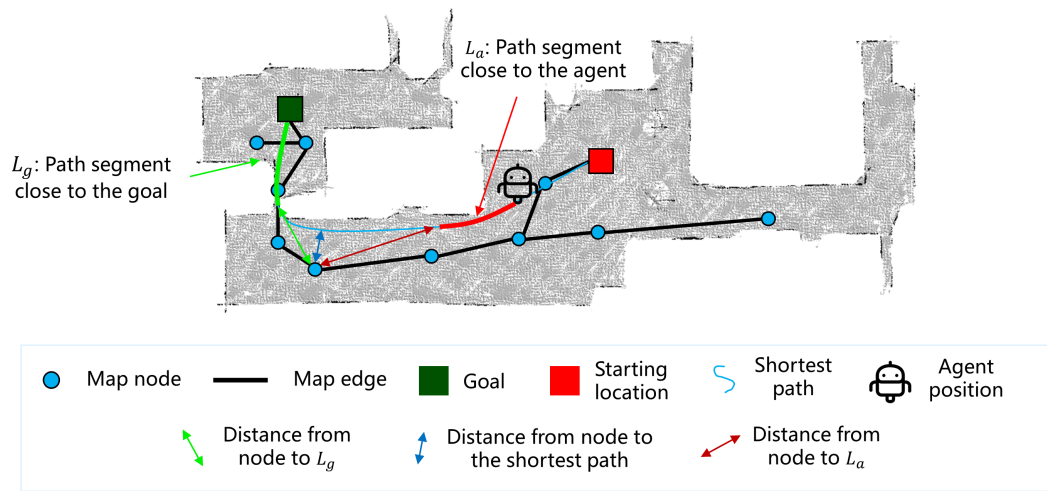


Figure 10. The visualization of distance metrics (c), (d), and (e) defined in Sec. 11.

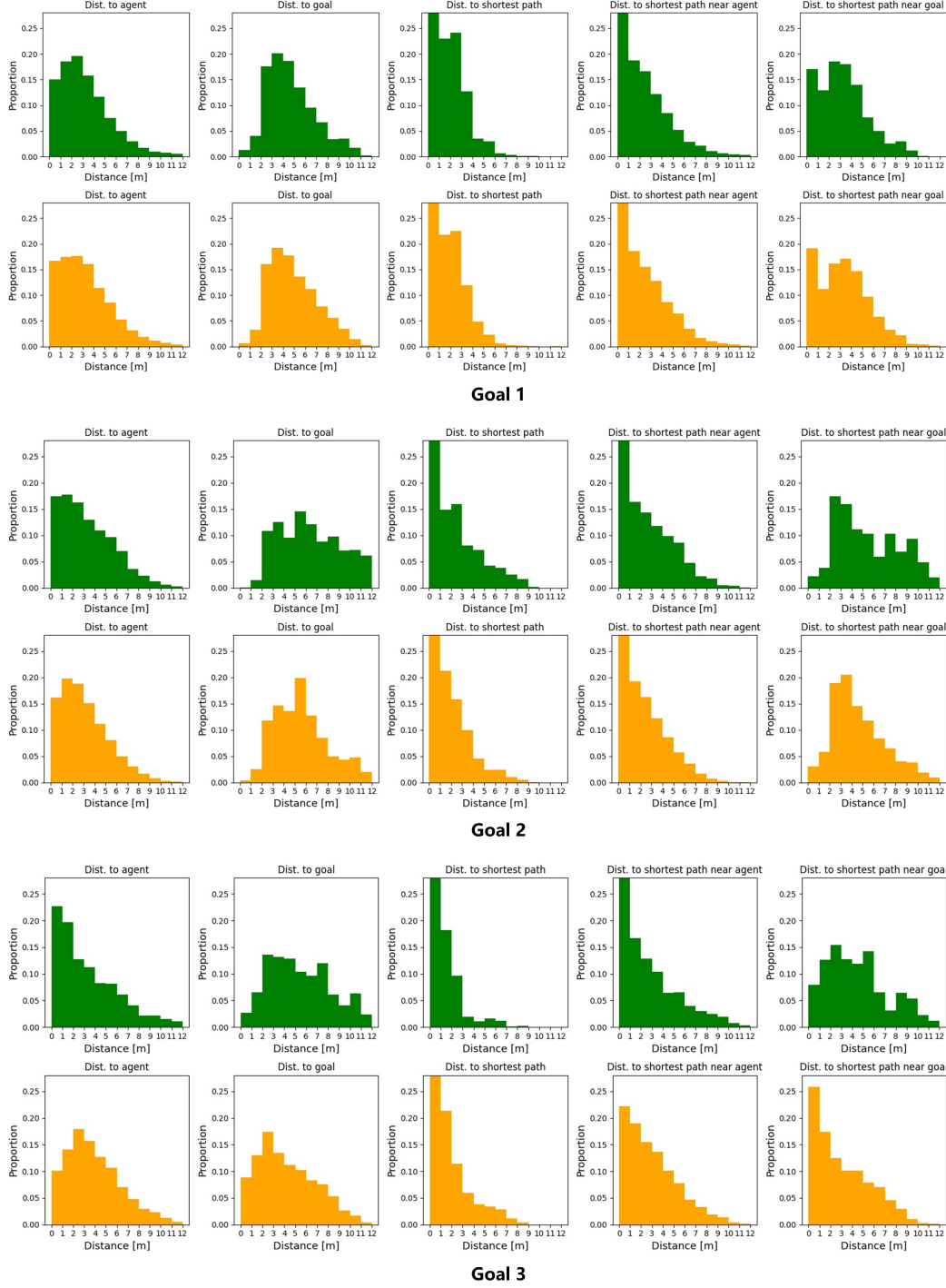


Figure 11. **Histograms of the five distance metrics defined in Sec. 11.** The data of these metrics is collected by evaluating the MemoNav on the 3-goal task in the Gibson scenes and averaged over five runs. The upper row (green) and lower row (orange) belong to the forgotten nodes and retained ones, respectively.

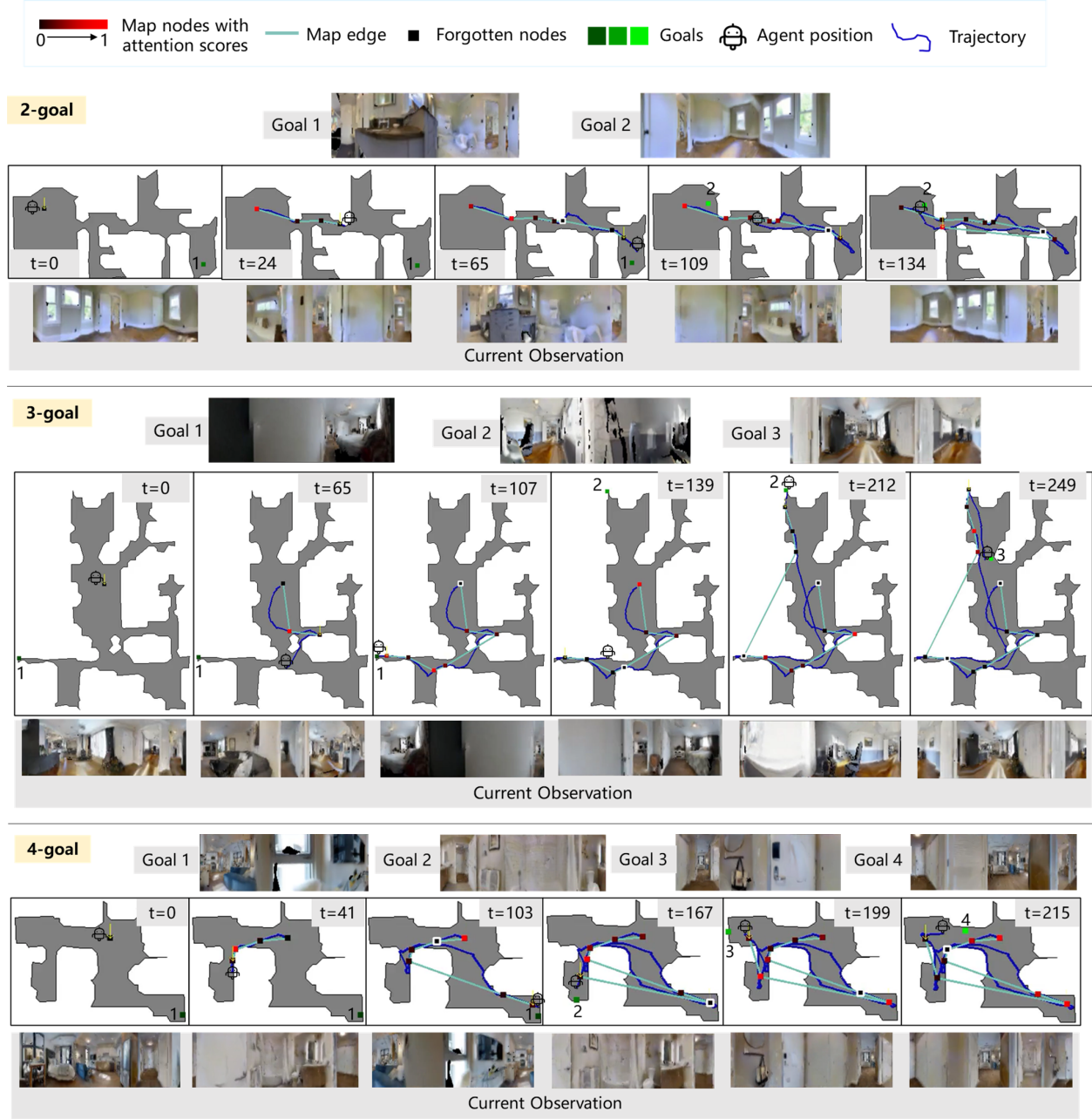


Figure 12. **Multi-goal example trajectories of MemoNav.** Each example shows both the topological map and the trajectory. The graph nodes are incrementally added to the map and selectively retained by the forgetting module in MemoNav. The examples illustrate that MemoNav flexibly neglects distant nodes. The yellow downward arrow denotes the current localized node of the agent. The comparison with VGM in these example tasks is recorded in the supplementary videos.

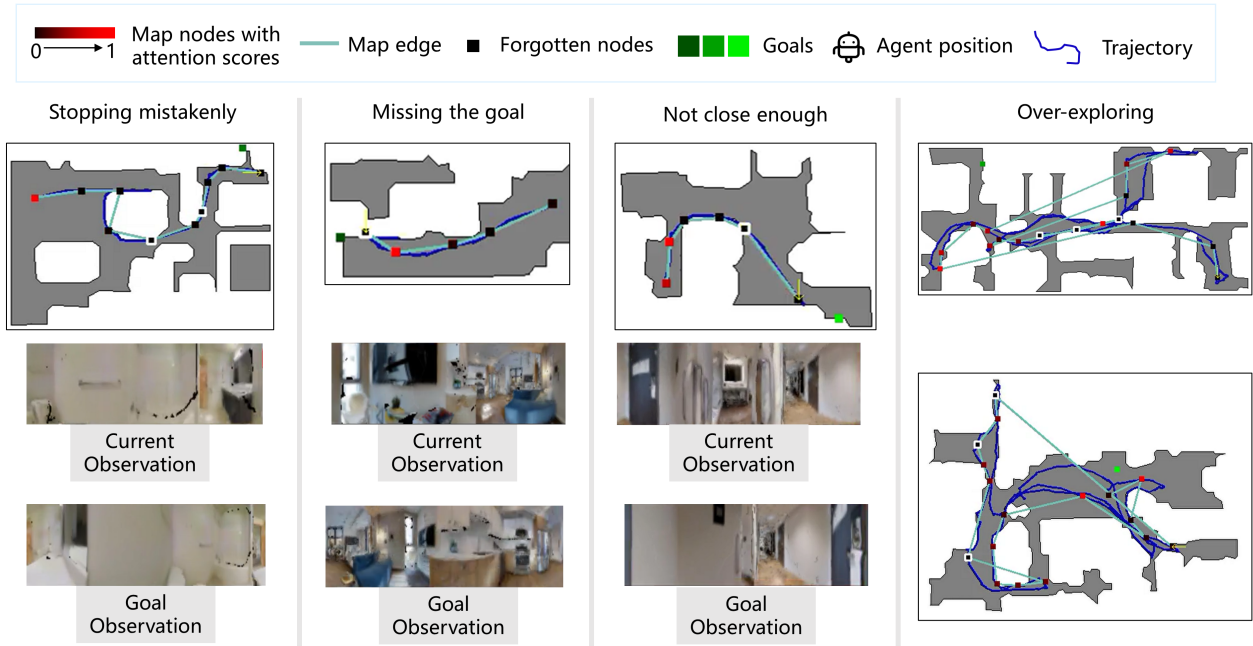


Figure 13. **Examples of failed episodes.** The agent encounters four major failure mode: (1) *Stopping mistakenly*: the agent implements stop at the wrong place. (2) *Missing the goal*: the agent has observed the goal but passes it. (3) *Not close enough*: the agent attempts to reach the goal it sees but implements stop outside the successful range. (4) *Over-exploring*: the agent spends too much time exploring open areas without any goals.



Figure 14. **Visualization of the LTM variation.** We show the agent’s trajectories in two example episodes and visualize the agent’s observations at the time steps when peaks appear on the LTM variation curves. The green arrows denote when the agent sets a new goal while the orange ones denote when peaks appear.