MemoNav: Working Memory Model for Visual Navigation

Supplementary Material

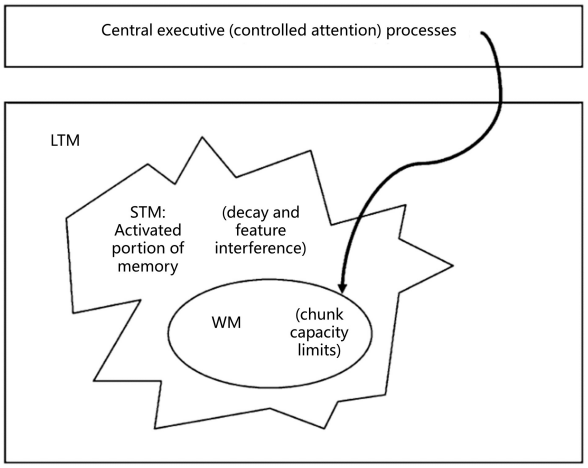

Figure 6. The memory model by Cowan et al. [15]. This figure is borrowed and adapted from its original paper.

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 [35] 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 [43] 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

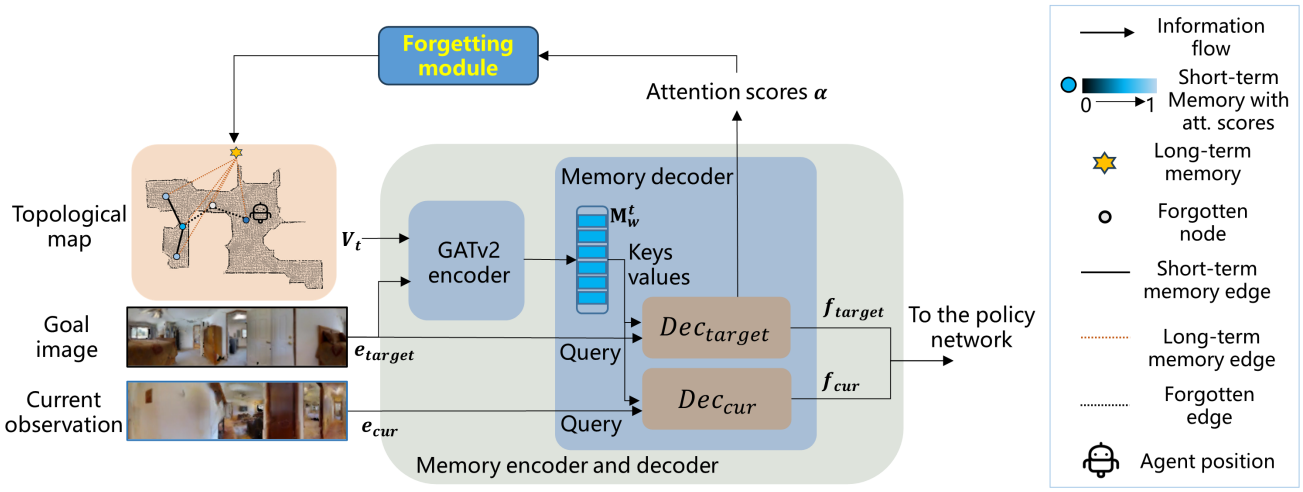

Figure 7. **The detailed structure of MemoNav.** The goal decoder $\mathcal{D}_{target}$ calculates the attention scores α for each STM feature in the topological map. Then the scores are used by the proposed forgetting module to remove redundant STM which will no longer be utilized for downstream action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

Long-term memory $\boldsymbol{n}_{global} \leftarrow \mathbf{0} \in \mathbb{R}^d$;

Attention scores for graph nodes V: $\alpha \leftarrow \mathbf{0} \in \mathbb{R}^{|V|}$;

while *not* AgentCallStop () **do**

`// Step 1: Update the topological map`

$\mathbf{l}_t \leftarrow$ GetCurrentPanorama();

G. UpdateMap($\mathbf{l}_t$);

`// Step 2: Retain the informative fraction of the STM`

Forgotten number $n \leftarrow$ Floor $(p \cdot |\mathcal{V}|)$;

Sorted indices $i \leftarrow$ Argsort(α);

Forgotten indices $i_{forgotten} \leftarrow i\,[0\colon n]$;

G. RemoveNodes($i_{forgotten}$);

$\boldsymbol{V} \in \mathbb{R}^{|\mathcal{V}| \times d} \leftarrow G$. GetNodeFeatures ();

Working memory $\boldsymbol{M}_w \leftarrow$ GATv2($\{\boldsymbol{V}, n_{global}\}$) `// Note that STM is fused before being forgotten in the next step so the features of forgotten STM have been fused into LTM.`

; $\boldsymbol{e}_{cur} \leftarrow \mathcal{F}_{enc}(\mathbf{l}_t), \boldsymbol{e}_{goal} \leftarrow \mathcal{F}_{enc}(\mathbf{l}_{goal})$;

$f_{cur} \leftarrow \mathcal{D}_{cur}\,(e_{cur}, \boldsymbol{M}_w)\,,\ f_{goal} \leftarrow \mathcal{D}_{goal}(\boldsymbol{e}_{goal}, \boldsymbol{M}_w)$;

$\alpha \leftarrow \mathcal{D}_{goal}$. GetAttScores()

`// Step 4: Action generation`

$\boldsymbol{x} \leftarrow$ LSTM(FC($[\boldsymbol{f}_{cur}, \boldsymbol{f}_{goal}, \boldsymbol{e}_{cur}]$));

$p\,(a_t \mid x) = \sigma(\text{FC}(x))$;

$a_t \leftarrow$ SampleFromDistribution($p(a_t \mid \boldsymbol{x})$);

end

VGM [23]. We also add positional embeddings to the encoded RGB-D observations to contain temporal information. Moreover, we concatenate the encoded RGB-D observations with the goal image embedding and project the

concatenated feature (1024D) to a 512D feature, so CNNL-STM 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 [43]. 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.

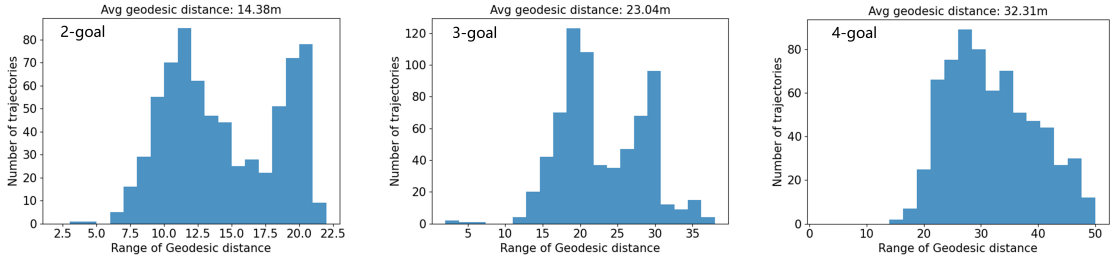

Figure 8. Histograms of geodesic distances for the multi-goal test datasets. As we set a distance limit for goals and discard invalid trajectories, a scene may own its prominent distance range, leading to the nonuniform histograms.

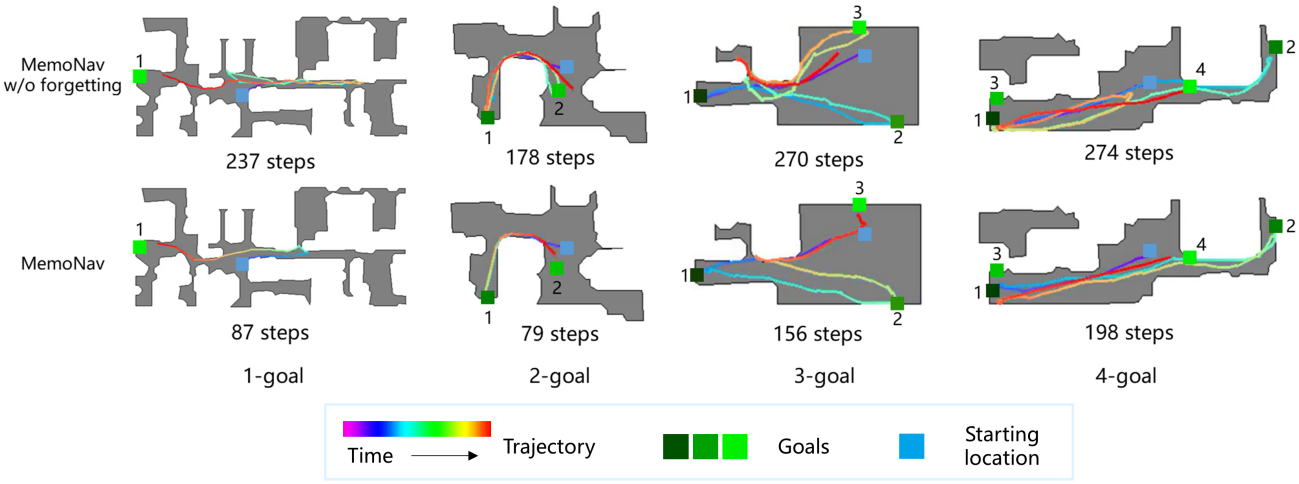

Figure 9. **Visualization comparing the MemoNav with and without the forgetting module.** We compare selected episodes at four difficulty levels in the Gibson scenes and visualize the top-down views. MemoNav without the forgetting module exhibits more sharp turns and tends to take more steps, demonstrating 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 [36] 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 incompatible 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.

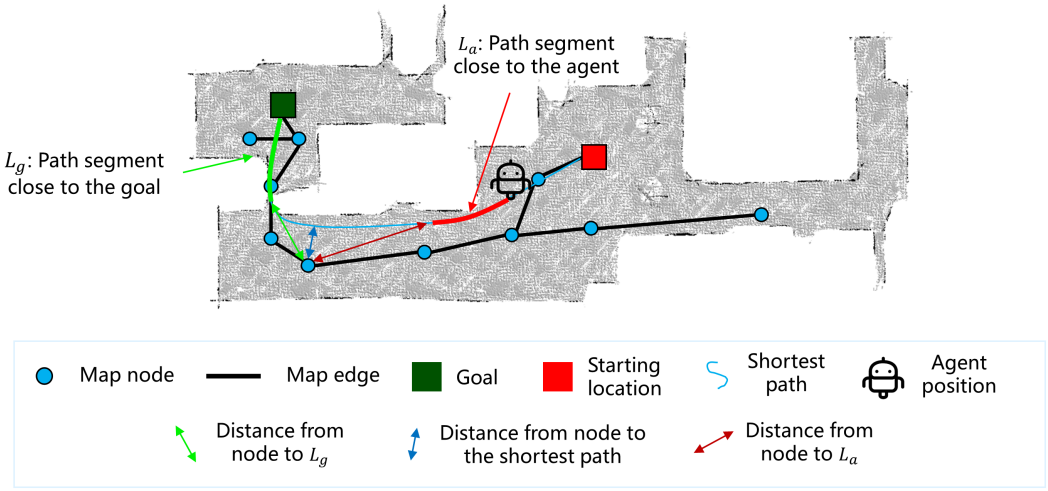

Figure 10. The visualization of distance metrics (c), (d), and (e) defined in Sec. 11.

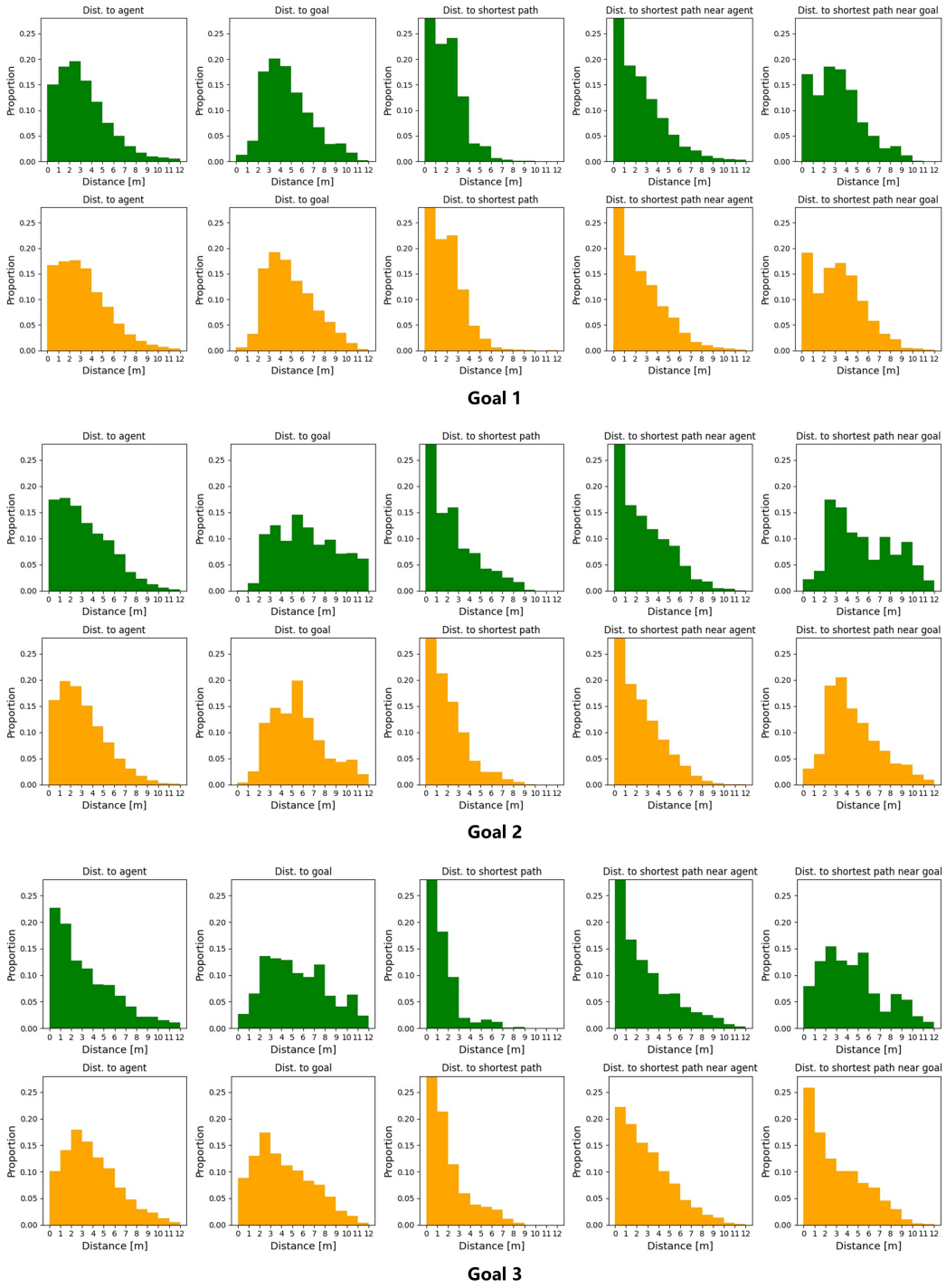

Figure 11. **Histograms of the five distance metrics defined in Sec. 11**. The data of these metrics is collected by evaluating the MemoNav on the 3-goal task in the Gibson scenes and averaged over five runs. The upper row (green) and lower row (orange) belong to the forgotten nodes and retained ones,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

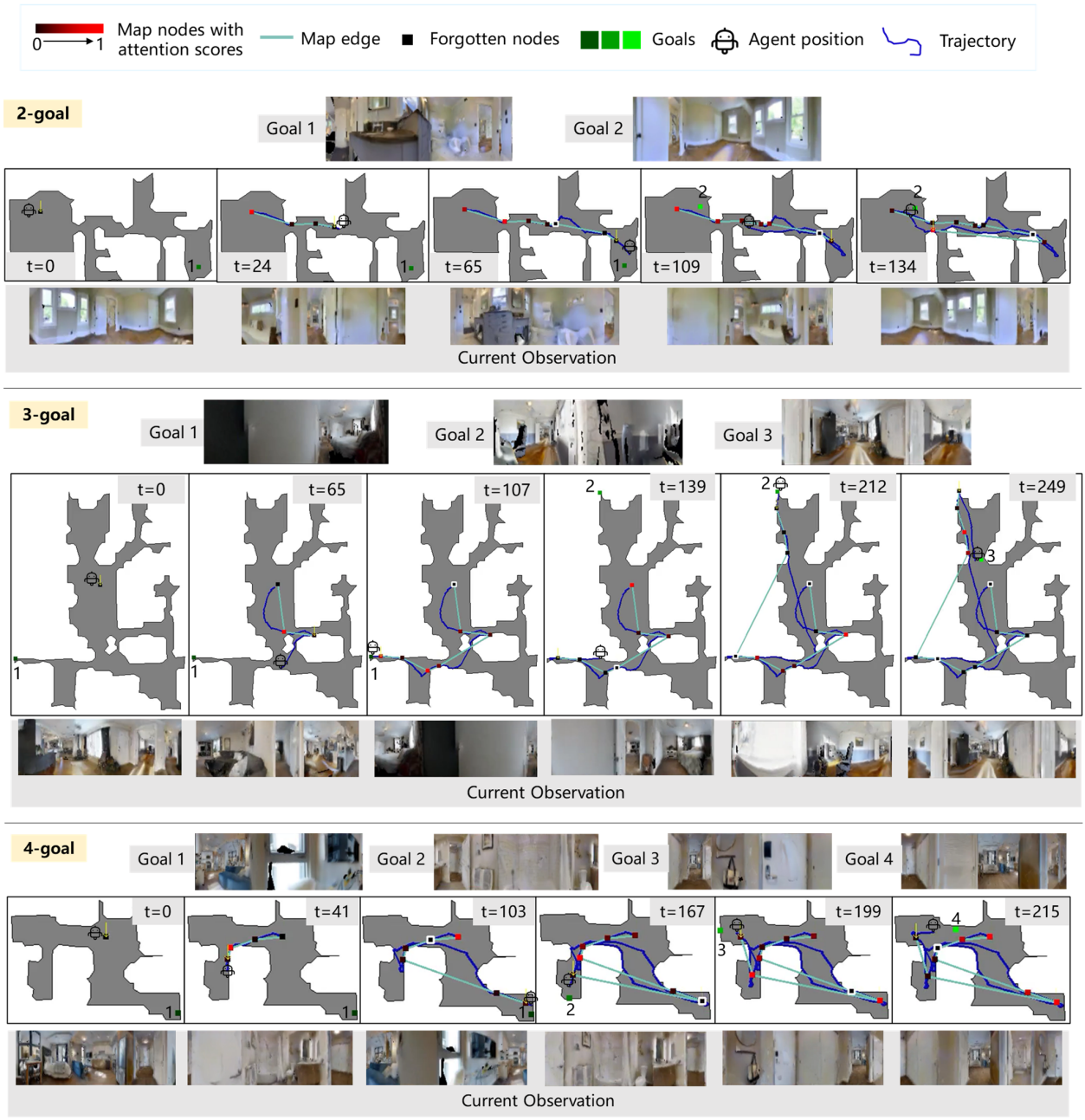

Figure 12. **Multi-goal example trajectories of MemoNav.** Each example shows both the topological map and the trajectory. The graph nodes are incrementally added to the map and selectively retained by the forgetting module in MemoNav. The examples illustrate that MemoNav flexibly neglects distant nodes. 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.

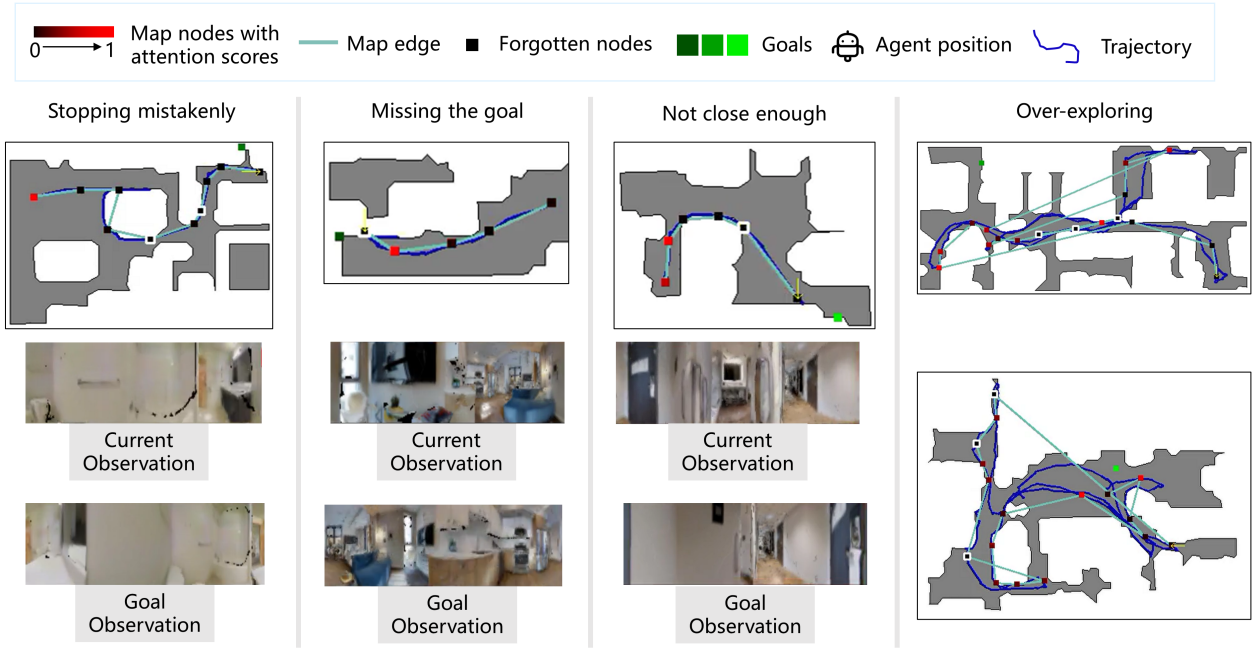

Figure 13. **Examples of failed episodes.** The agent encounters four major failure mode: (1) *Stopping mistakenly*: the agent implements stop at the wrong place. (2) *Missing the goal*: the agent has observed the goal but passes it. (3) *Not close enough*: the agent attempts to reach the goal it sees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

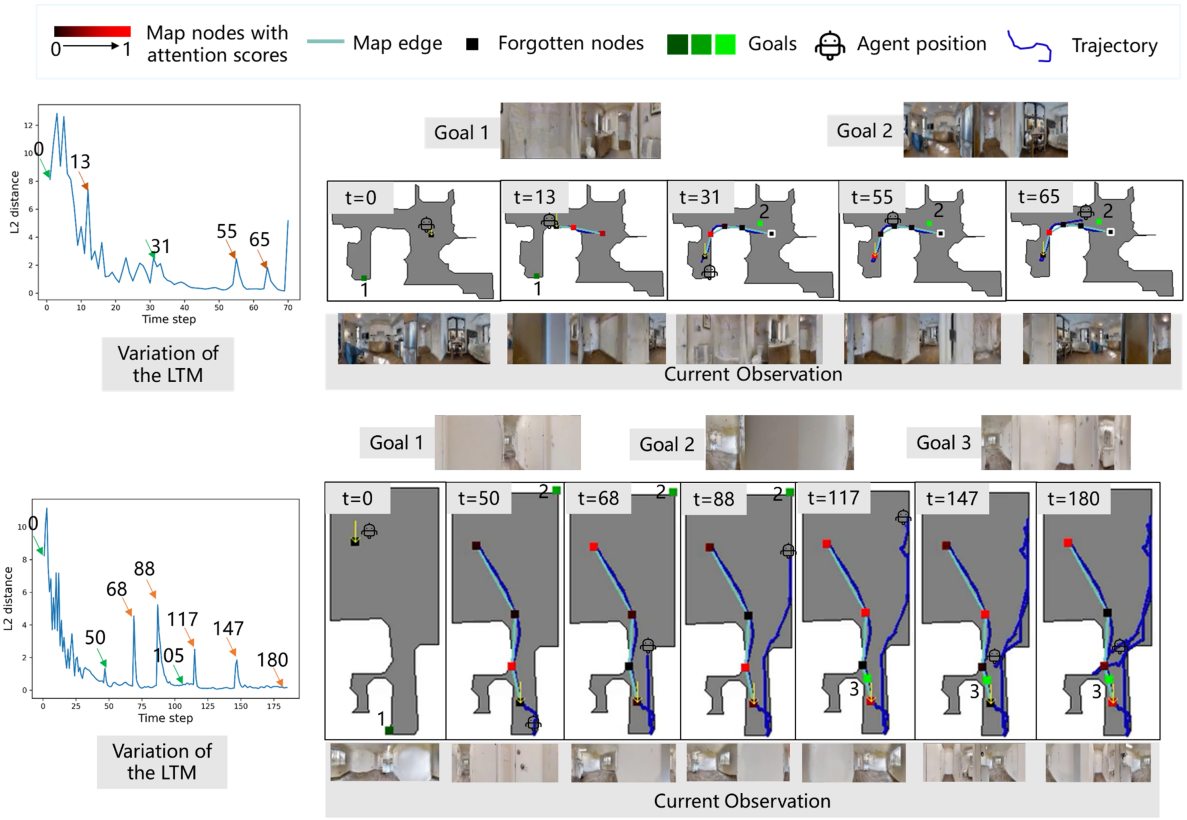

Figure 14. **Visualization of the LTM variation.** We show the agent's trajectories in two example episodes and visualize the agent's observations at the time steps when peaks appear on the LTM variation curves. The green arrows denote when the agent sets a new goal while the orange ones denote when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