Visual Graph Memory with Unsupervised Representation for Visual Navigation

Obin Kwon Nuri Kim[†] Yunho Choi[†] Hwiyeon Yoo[†] Jeongho Park[†] Songhwai Oh

Department of Electrical and Computer Engineering, ASRI, Seoul National University [*]
{firstname.lastname}@rllab.snu.ac.kr, songhwai@snu.ac.kr

Abstract

We present a novel graph-structured memory for visual navigation, called visual graph memory (VGM), which consists of unsupervised image representations obtained from navigation history. The proposed VGM is constructed incrementally based on the similarities among the unsupervised representations of observed images, and these representations are learned from an unlabeled image dataset. We also propose a navigation framework that can utilize the proposed VGM to tackle visual navigation problems. By incorporating a graph convolutional network and the attention mechanism, the proposed agent refers to the VGM to navigate the environment while simultaneously building the VGM. Using the VGM, the agent can embed its navigation history and other useful task-related information. We validate our approach on the visual navigation tasks using the Habitat simulator with the Gibson dataset, which provides a photo-realistic simulation environment. The extensive experimental results show that the proposed navigation agent with VGM surpasses the state-of-the-art approaches on image-goal navigation tasks. Project Page: https://sites.google.com/view/iccv2021vgm

1. Introduction

Visual navigation has been one of the fundamental building blocks in developing intelligent autonomous agents. To effectively navigate through a large-scale environment, an agent is required to build an internal representation of the environment from raw sensory inputs and its own actions. Using this internal representation, the agent can store useful information such as its navigation history and successfully perform tasks with additional guidance.

Numerous studies in psychology and cognitive science have shown that animals build a landmark-based topological representation about the environment during navigation [14, 16, 18, 31, 33]. Inspired by this observation, several navigation methods based on a topological map have been proposed [2, 3, 5, 15, 25, 29]. These methods use a graph, in which a vertex represents a landmark in the environment and an edge represents the relationship between two vertices, such as reachability and proximity.

Recent works in computer vision and deep learning show remarkable improvements in visual navigation tasks using topological representation [4, 8, 9, 23, 24, 27]. The existing methods autonomously build a topological map about the environment using a pretrained classifier network, which can determine whether two image observations are close or not. However, there are limitations to currently available methods. (1) To train the classifier network, elaborately designed annotation rules based on accurate geometric information are required for preparing training datasets. (2) Their navigation performance depends on the quality of geometric information (*e.g.*, odometry) for calculating distances between nodes or providing local point-goals to the navigation policy. (3) The majority of methods [4, 9, 23, 24, 27] assumes that the environment is previously observed before. These methods use a pre-built graph and the size of the graph is fixed during the navigation, demanding a pre-exploration time to build the entire topological map before an actual navigation task can begin.

In order to address the aforementioned limitations, we propose the visual graph memory (VGM), a graph-structured memory for visual navigation. VGM is constructed using an RGBD image encoder, which is trained in an unsupervised manner. This encoder can be trained using only an unlabeled image dataset, and this can release the burden of annotating the training dataset. We also present a navigation framework which is designed to utilize the VGM to tackle visual navigation tasks. The proposed navigation framework consists of two components: a memory update module and a navigation module, as shown in Figure 1a. The memory update module selectively stores image representations from the navigation history and constructs a VGM using their spatio-temporal relationships. The navigation module uses the VGM to select an ideal action for a given navigation task. Specifically, this module encodes the VGM using a graph convolutional network and extracts the attention-guided context information from the encoded VGM using the current and target images.

The proposed navigation framework does not require robot pose information unlike previous methods, thus the navigation performance is independent of the environmental errors. Furthermore, the proposed navigation framework incrementally builds the VGM during navigation. Hence, the proposed method is able to conduct a given navigation

[*]This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01190, [SW Star Lab] Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning). † These authors contributed equally to this work.

task in an unfamiliar environment without pre-exploration, unlike existing methods.

We evaluate the proposed method for the image-goal navigation task, which requires an explorative behavior and memory to find a target location based on an image. The proposed navigation method finds the target location in 76% of the test episodes, which is 11.8% better than the best performing baseline among the state-of-the-art approaches using visual memories. In terms of success weighted by path length (SPL), the proposed method achieves 0.64, which is a 14.3% improvement over the baseline. The main contributions of this paper are as follows:

- We present the visual graph memory (VGM), which can be built without any geometric information-based annotations. The VGM enables unsupervised topological simultaneous localization and mapping (SLAM) in an unseen environment.
- We also present a navigation framework that can utilize the VGM for visual navigation tasks. The proposed navigation framework is able to conduct a given visual navigation task in a completely unseen environment. Furthermore, the proposed method is highly robust against pose information errors compared to existing methods.
- The extensive experimental results show that the proposed navigation framework with VGM outperforms the state-of-the-art visual navigation methods based on visual memories.

2. Related Work

From the classical robot navigation methods [12, 30] to the recent learning-based methods [6, 7, 10], explicit 2D-grid metric maps have been widely used as a method to represent the environment for navigation. A metric map can provide accurate information about the environment as well as the state of the robot if mapping and localization are accurate. However, building and maintaining an accurate map from raw sensory signals is computationally expensive.

Numerous studies in classical robot navigation [2, 3, 5, 15, 25, 29] build a topological map to represent the environment. A topological map is a graph, in which a node contains topological information and an edge represents direct reachability between a pair of nodes. While the topological representation can be less accurate than metric map-based approaches, it is more convenient to build and maintain due to its concise and sparse representation.

One of the main difficulties that arise using topological representations is place recognition using raw sensory inputs. Thus, earlier topological approaches focused on mapping and localization rather than on navigation tasks. Furthermore, they all used handcrafted features, which require relatively accurate geometric information under static environments.

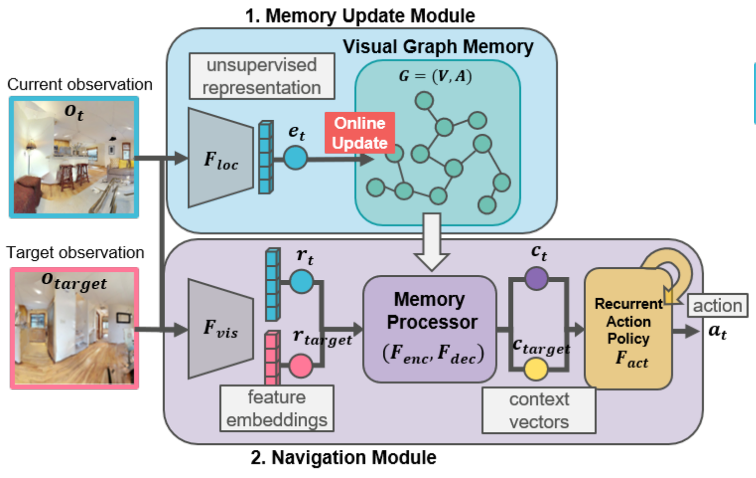
In recent years, learning-based methods have been proposed to construct a navigation memory in a form similar to the classical topological map [4, 8, 9, 11, 20, 23, 24, 27]. A number of approaches use a pretrained classifier network to build a graph autonomously without human intervention [4, 8, 20, 23, 24, 27]. The classifier is trained to determine whether two observations are from the same area. While these methods can autonomously build a graph representation from the experience, they need a sufficient amount of data collected from robot demonstration and manually-specified annotation rules to prepare training data. For example, [20, 24, 27] collected random exploration data using the agent to train the classifier and a temporal distance between nodes is specified to teach the network how to detect novel places. [8] uses the depth information in a 3D mesh of the environment to calculate the distance and visibility between two locations. [4] uses a 2D occupancy map to determine whether two locations are visible from each other. Meanwhile, [23] trained a network that predicts the navigability of a local locomotion policy using robot demonstrations. The estimated navigability is used as a metric for determining node boundaries. In contrast, the proposed VGM uses an image encoder trained with an unlabeled image dataset, without the need for the elaborately designed annotation rules and accurate geometric information.

Another main limitation of existing methods is that the majority of them can only be executed on a pre-built graph [9, 11, 20, 23, 24, 27]. [9, 11] assume that the environment has a predefined graph structure, so the navigation agent does not have to determine appropriate landmarks to build a graph. [20, 23, 24, 27] build a graph memory for navigation in an unstructured environment, but they need sufficient exploration time before conducting actual navigation tasks. Because of this characteristic, these methods are not appropriate to be implemented on a target-searching task in unseen environments. These methods are unable to search the target location when the target is not in the pre-built graph. However, if the target location is in the pre-built graph, it means that the target has been already discovered during the exploration time even without the actual navigation policy.

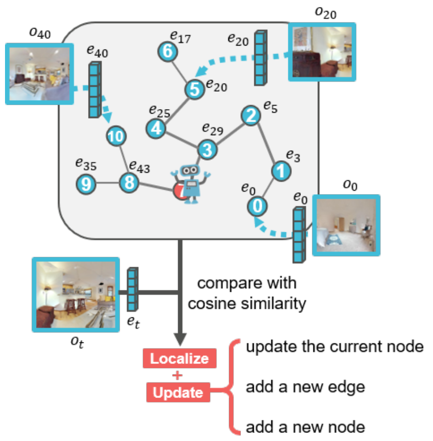
The topological map-based methods [4, 8] that can conduct image-goal navigation in unseen environments are recently proposed. They use the pose information of a robot to build a graph and navigate the environment. [4] stores the position of each node in the graph to calculate the distances between nodes. [8] uses a pose sensor to store the relative pose between nodes and draw a local metric map for the local point-goal navigation policy. In contrast, our navigation method does not use any pose information from the environment or sensors. Thus, our method can be robust against environmental errors. Also, our method can be easily implemented in an environment where an accurate pose information is unavailable.

Different types of navigation memories are also exploited in various ways. In [26, 34, 36], a knowledge graph, which is constructed using object relationships or room layouts, is studied. They use a learned (or predefined) knowledge graph to conduct given navigation tasks instead of building an internal representation of the environment like the methods mentioned above.

[13, 19] proposed a navigation system which accumulates all past observation features and compares them to the present one during navigation. Since all past observations are stored in the memory, the required computational complexity increases as more observations are collected. In contrast, the proposed VGM selectively stores observations



(a) Overview



(b) Memory Update

Figure 1: **(a) Overview of the proposed method**. The proposed method consists of two modules as shown in the figure. The memory update module builds a visual graph memory (VGM) using e_t, which is the unsupervised representation of the current image o_t encoded by the encoder F_{loc}. In the navigation module, F_{vis} encodes the current and target images (o_t, o_{target}) into feature embeddings (r_t, r_{target}). Then, the memory processor associates them with the VGM and outputs context vectors (c_t, c_{target}). Finally, c_t, c_{target} and r_t are concatenated and fed into a recurrent action policy network F_{act}. The action policy network F_{act} returns an action distribution for the agent to take. **(b) Memory update process.** In a VGM, each node embedding is an image representation e_t. At every time step t, the memory update module compares e_t with the node embeddings by calculating similarities. Based on the results, the memory update module localizes the agent and determines how to update the graph structure of the VGM.

and contains a significantly smaller number of memory elements in the memory. In addition, our navigation agent is able to conduct a given navigation task even with a limited size of memory.

3. Method

3.1. Problem Formulation

We first formulate the image-goal navigation task before explaining the proposed navigation method. The objective of image-goal navigation is to successfully arrive at the target location using the image of the target location, o_{target}, and a sequence of image observations, $\{o_t\}$, collected during navigation. At each time step t, the agent receives an RGBD panoramic image, o_t, of the current location. Any additional guidance, *e.g.*, GPS, demonstration images, and language instructions, are not provided.

The target location may not be visible from the start position, so the agent is required to explore an unfamiliar environment to find the target. Remembering the navigation history can be helpful to the agent for an efficient target search. Also, it is important to explore places where the target is more likely to be present, rather than exploring all areas.

The action space of an agent is discrete and has four options: {Stop, Move Forward, Turn Left, Turn Right}. When the agent performs a forward action, the agent moves 0.25m forward. A turning action rotates the agent for 10 degrees in the selected direction. The navigation becomes a success when the agent takes the stop action within 1m from the target location.

3.2. Method Overview

Figure 1a shows an overview of the proposed navigation framework with a visual graph memory (VGM). The navigation framework consists of two modules: the memory update module and the navigation module. The memory update module contains a pretrained image encoder, F_{loc}, which encodes an image o_t into a representation vector e_t. The memory update module gradually builds a VGM using e_t during the navigation. The navigation module has another image encoder F_{vis} which encodes the observations (o_t, o_{target}) into the feature embeddings (r_t, r_{target}). Using r_t, r_{target} and the VGM, the memory processor produces context vectors (c_t, c_{target}). The memory processor consists of a graph convolutional network, F_{enc}, and the attention networks, F_{dec1} and F_{dec2}. Finally, c_t, c_{target} and r_t are concatenated and passed to a long short-term memory (LSTM) network which returns an action a_t for the agent to take. Based on the VGM, the agent can efficiently search for the target location in an unseen environment. In the remainder of this section, the components of the proposed navigation system are described in detail.

3.3. Visual Graph Memory

The memory update module contains a VGM $= (V, E)$, where V and E represent nodes and edges, respectively. An adjacency matrix A can be constructed from a VGM. A pretrained image encoder F_{loc} encodes the current image observation o_t into e_t. F_{loc} produces similar representations if the observations are from similar locations. The node embeddings of V are composed of image representations from past image observations. Based on similarities between the current observation and the node embeddings of the VGM, the memory update module localizes the agent and updates the VGM.

3.3.1 Representation Learning

Prior studies based on topological maps [4, 8, 20, 23, 24, 27] utilize neural networks similar to F_{loc}, which classify whether a given pair of observations have a smaller distance than a specified temporal or spatial distance. Some of them used a 3D or 2D-grid map of the environment to calculate the visibility between a pair of locations. Humans, on the other hand, remember novel landmarks instead of equally-spaced distances. In this regard, we hypothesize that unsupervised learning of image representation is sufficient to detect the novel observations and build a topological map.

We have adapted the prototypical contrastive learning (PCL) [21], a recently proposed unsupervised representation learning method. This contrastive learning method first clusters images in the dataset. Then, the encoder F_{loc} is trained to encode images of the same cluster closer while images of different clusters are placed further. The closer the distance between the locations of the observations, the more likely they belong to the same cluster because they have similar appearances.

The training dataset is only composed of randomly sampled observation images from the training environment. Once F_{loc} is trained, the parameters of F_{loc} are frozen while the navigation module is trained. In Section 5, we show that this unsupervised representation learning is sufficient for building a compact graph representation of the environment and this graph is highly effective for navigation. The implementation details of PCL for our work are provided in the supplementary material.

3.3.2 Memory Update Module

As shown in Figure 1b, the VGM is updated in two steps: localization and graph update. In the localization step, the memory update module looks for a node similar to the current observation to determine where the agent is in the VGM. Next, in the graph update stage, the memory update module determines how to update the VGM.

Localization. The position of an agent can be localized using node embeddings in the VGM and the current observation embedding. Assume that the last localized node is $v_n \in \mathbb{R}^d$, and the number of nodes in the VGM is N_t at time t. F_{loc} encodes a new observation o_t into $e_t \in \mathbb{R}^d$, where d is the dimension of the embedding vector. Then, the memory update module calculates $S = \{s_i \mid s_i = \frac{v_i \cdot e_t}{\|v_i\|\|e_t\|}, i = 1, ..., N_t\}$, a set of cosine similarities between e_t and nodes in $V = \{v_1, v_2, ..., v_{N_t}\}$. If there is a node v_i whose cosine similarity s_i is higher than a threshold s_{th}, the module decides that the agent is near v_i in the VGM. There can be multiple nodes whose cosine similarities are higher than the threshold. In this case, the module selects the most similar node among them.

We set s_{th} considering the number of nodes and sparsity of the generated graph. Note that the value of s_{th} does not affect the training of F_{loc}; hence we can adjust the sparsity of a graph by changing s_{th} without re-training F_{loc}. In Section 5, we provide the comparison of generated graphs and the performances of the agent across various values of s_{th}.

The localization process of our method is simpler than other methods. Previous methods use an additional network which takes two images as input to determine whether the two observations are close. In this case, every pair of an image of a node in the graph and the current observation image has to be processed by the neural network for localization at every step. Ours can just calculate cosine similarities using previously stored visual features of all nodes without the need for additional computations.

Graph Update. Suppose that the current location is localized as v_i. If the localized node v_i is the same as the last localized node v_n, i.e., $i = n$, the VGM is not updated. If they are different, a new edge between v_i and v_n is added. The embedding of v_i is replaced with the current feature e_t. If the current location cannot be localized in the VGM, a new node v_{N_t+1} with embedding e_t and an edge between the new node and v_n are added to the VGM. We also store the time step t with the node embedding when the node is added or updated. These time steps are used to order nodes in the navigation module as described below. With this update process, the memory update module stores the summarized snapshots of the navigation history and their spatio-temporal relationships.

3.4. Navigation Module

The navigation module processes the updated VGM and estimates the most appropriate action for the navigation task. In the navigation module, an image encoder F_{vis} encodes the observations, o_t and o_{target}, into feature embeddings, r_t and $r_{target} \in \mathbb{R}^d$, respectively. We set two image encoders F_{loc} and F_{vis} separately, as we want F_{vis} to learn additional information related to navigation skills such as obstacle avoidance. In the remainder of this section, we explain how the feature embeddings, r_t and r_{target}, are used with the VGM in the navigation module.

3.4.1 Memory Processor

Encoder. The memory processor has an encoder-decoder structure as shown in Figure 2. We use a graph convolutional network (GCN) [17], which is a powerful tool for processing graph-structured data, to encode the VGM $= (V, E)$. In a single graph convolutional layer, the feature embedding of each node $H \in \mathbb{R}^{N_t \times d}$ is projected by a parameter matrix $W \in \mathbb{R}^{d \times d}$ and updated with the aggregated information from its neighbors. An adjacency matrix A, which is constructed based on edges E, is used for aggregation. Through the multiple (K) graph layers in GCN, the node embedding is updated with the information from its K-hop distant neighbors. The encoding process F_{enc} is formulated as follows:

$$\begin{aligned} H^{(0)} &= \text{FC}([V; r_{target}]), \\ H^{(l+1)} &= \text{ReLU}(\tilde{A} H^{(l)} W^{(l)}), \\ F_{enc}(V, A; \psi) = M &= H^{(K)}, \end{aligned} \tag{1}$$

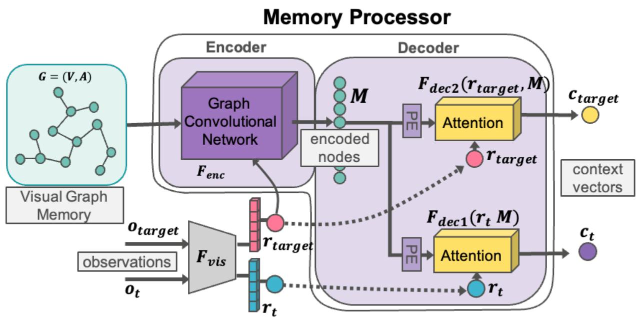


Figure 2: Detailed structure of the memory processor in the navigation module. Each node in VGM is encoded with the encoder F_{enc}. The feature embedding r_t and r_{target} from F_{vis} are fed into the decoder with the encoded nodes M. The context vectors, c_t and c_{target} are calculated by attentioning r_t and r_{target} over M. PE attaches positional encodings based on node visitation times.

where $[\cdot;\cdot]$ denotes concatenation, and $\tilde{A} \in \mathbb{R}^{N_t \times N_t}$ is a normalized adjacency matrix from A with self-loops, FC is a fully connected layer, and ReLU is a rectified linear unit.

Before the initial graph convolutional layer, we encode each embedding of a node with r_{target} using a single linear layer FC to fuse the information about the target observation. Each node embedding $v_i \in \mathbb{R}^d$ is concatenated with $r_{target} \in \mathbb{R}^d$, after which FC projects $[V; r_{target}] \in \mathbb{R}^{N_t \times (d+d)}$ into $H^{(0)} \in \mathbb{R}^{N_t \times d}$. Following K graph convolutional layers, we obtain a memory $M \in \mathbb{R}^{N_t \times d}$ which contains the spatial knowledge about the environment. The proposed navigation agent can take the whole graph as an input using the GCN, and thus it can be trained in an end-to-end manner. This allows the agent to learn how to use the graph structured VGM effectively, without any manually-specified protocol to decide which node to navigate. Also, the proposed method can be flexible in dealing with localization errors, as the agent can see all the nodes at a glance.

Decoder. We designed the decoder F_{dec} to extract information useful for navigation from the encoded memory M in the context of the current observation and the target observation. We apply the multi-head attention mechanism [32] to aggregate information from the encoded memory $M = \{m_1, ...m_{N_t}\}$, where $m_i \in \mathbb{R}^d$, in the context of o_t and o_{target}. The feature vectors $r_t, r_{target} \in \mathbb{R}^d$ are used as queries and the encoded memory $M \in \mathbb{R}^{N_t \times d}$ is the key and value. Using multi-head attention, an agent can get diverse insights about the VGM from the different decoded outputs of the heads in the attention network.

The decoder utilizes the time step information, which is stored with the node embeddings. Inspired by the positional encoding in the transformer network [32], we add sinusoidal positional encoding to each element in M according to its relative temporal distance from the current time step t. We denote $\tilde{M}$ as the encoded memory of M with positional encodings.

The attention function, $\text{Att}(\cdot,\cdot)$, and its multi-head version, $\text{mhAtt}(\cdot,\cdot)$, with J heads can be formulated as

$$\text{Att}_j(r_t, M) = \sigma\left(\frac{(W_j^q r_t)(W_j^k \tilde{M}^T)}{\sqrt{d}}\right)(W_j^v \tilde{M}^T)^T, \qquad (2)$$
$$\text{mhAtt}(r_t, M) = W_m([\text{Att}_j(r_t, M)]_{j=1}^J),$$

where $W_j^q, W_j^k, W_j^v \in \mathbb{R}^{d \times d}$ are the parameter matrices in the j-th attention head and σ is the softmax function. $[\text{Att}_j]_{j=1}^J$ means that we concatenated the outputs of attention networks from j=1 to j=J. The parameter matrix $W_m \in \mathbb{R}^{d \times (Jd)}$ projects them onto a d-dimension vector.

The decoder F_{dec} on r_t is formulated as:

$$F_{dec}(r_t, M) = \text{LN}(\text{FC}(\bar{c}_t) + \bar{c}_t), \qquad (3)$$
$$\text{where } \bar{c}_t = \text{LN}(\text{mhAtt}(r_t, M) + r_t),$$

where LN is a layer normalization and FC is a fully conntected layer. We also conduct F_{dec} on r_{target} to get the context information related to the target from the memory. The decoder outputs two context vectors, $c_t = F_{dec1}(r_t, M)$ and $c_{target} = F_{dec2}(r_{target}, M)$, where F_{dec1} and F_{dec2} are separate networks with the same structure.

3.4.2 Navigation Policy

The context vectors c_t, c_{target}, and the visual feature of the current observation r_t are concatenated as a single vector and fed into an LSTM network. This network outputs the distribution of actions based on the concatenated features. The navigation agent outputs a stochastic policy as follows:

$$x_t, h_t = \text{LSTM}(FC([c_t, c_{target}, r_t]), h_{t-1}) \qquad (4)$$
$$p(a_t|x_t) = \sigma(\text{FC}(x_t)),$$

where h_t is the hidden state vector of LSTM.

4. Learning

IL and RL. In contrast to previous topological navigation methods, the proposed method can be trained for various tasks, using both imitation learning (IL) and reinforcement learning (RL). We first train the agent using IL to initialize networks and finetune with RL.

The agent is supervised to minimize the negative log-likelihood of the ground-truth actions using the cross-entropy loss. The loss function for the IL stage is

$$\text{Loss}_{IL} = E_{\tau \sim \mathbb{D}}\left[\sum_{t=0}^{T_\tau} -a_t^* \log p(a_t|x_t)\right], \qquad (5)$$

where T_τ is the total length of a sampled demonstration, $\tau = (o_t, a_t^*)_{t=0,\ldots,T_\tau}$, from the training dataset $\mathbb{D}$ and a_t^* is the ground-truth action from the oracle agent.

We further finetune the agent's policy with RL, namely proximal policy optimization (PPO) [28], to encourage the exploratory behavior of the agent. We set a dense reward function proportional to the progress of the distance to the target location as $r_t(s, a) = \lambda(d_{t-1} - d_t)$, where d_t is the distance to the target and λ is a positive scalar constant for adjusting the reward scale. A small penalty (-0.01) is added at every time step to encourage a faster search. When the agent reaches the target location and chooses a stop action, a large success reward ($+10$) is given.

Auxiliary losses. We have added a few auxiliary tasks to encourage the memory processor to extract the useful information from the memory. The first one is to classify whether the agent has been in the current location before. If the agent has passed the current area before, the label u_t^* is set to 1. The second task is to predict the distance score s_t from the current location to the target location. This task is to assist the agent to learn when to apply the stop action near the target location. Simple two-layer MLP networks, FC$_{aux1}$ and FC$_{aux2}$, are attached to the memory processor for these two auxiliary tasks. They predict the proper value for each task using the context vectors (c_t, c_{target}) from the decoder. The auxiliary loss term is

$$\tilde{u}_t, \tilde{s}_t = FC_{aux1}(c_t), FC_{aux2}([c_t; c_{target}])$$
$$\text{loss}_{\text{aux}} = E_{\tau \sim \mathbb{D}}\left[\sum_{t=0}^{T_\tau} -u_t^* \log \tilde{u}_t + (s_t^* - \tilde{s}_t)^2\right], \quad (6)$$

where s_t^* denotes the ground-truth distance score to the target. These auxiliary tasks are simultaneously trained during IL and RL. The auxiliary loss term in (6) is added to each action loss term of IL and RL. For example, in the IL stage, total loss becomes

$$\text{Loss}_{total} = \text{loss}_{IL} + \text{loss}_{aux}. \quad (7)$$

5. Experiments

5.1. Baseline Methods

We compare the proposed method with a number of baselines that use various types of memory. The considered baselines are as follows:

- **CNN + LSTM** [37]. This baseline is an LSTM model with CNN, which is adapted from [37].
- **ANS + pred. target pose** [7]. We implemented this baseline in a similar manner as [8]. This model builds a metric map using depth information. Based on the metric map, a global policy chooses a location to explore. Then, the local policy navigates to the point chosen by the global policy. Using a pretrained target pose estimator, we set the output of the global policy to be the relative position of the target when the target is detected.
- **Exp4nav** [10]. We adapted this model to the image-goal navigation task. This model has multiple CNN networks to process the current observation and the multi-scale metric map, as well as the target observation. A recurrent policy takes the features from the CNN networks and outputs an action.
- **SMT** [13]. This model stacks all the visual features of the past observations and the pose information as a navigation memory. It uses a transformer network to process this memory in the context of the current and target observations. If VGM stores all the observations and fully connects all the nodes, the network architecture would be similar to this baseline.
- **Neural Planner** [4]. This method first collects a certain amount of rollouts from an exploration policy to build a topological map. Then, a pretrained neural planner calculates the path to the node which is most similar to the target image. A local point-goal navigaiton policy follows the path. We adapted this model to use the target estimator of the ANS baseline to determine whether the target location is near. If the target is not detected after the local policy arrived at a node similar to the target, the exploration policy starts the exploration again to enlarge the graph. This model uses a pose sensor to store the position of each node.
- **Exploration + SPTM [27]**. Despite [27] can only navigate through the prebuilt graph, this model can be adapted in similar way as Neural Planner. This baseline does not require pose information because it uses image-based local navigation policy and does not calculate explicit distances between nodes. In comparison to the Neural Planner, the Dijkstra algorithm is used for planning.
- **NTS** [8]. This model builds a topological map using several pretrained networks which provide estimated geometric and semantic information. This baseline uses a pose sensor to store relative pose between nodes and provides the local metric map to the local point-goal navigation policy. We report scores from the paper [8] as the experimental setting are the same.

All the end-to-end policy baselines (CNN+LSTM, Exp4nav and SMT) and our proposed model are first trained using imitation learning, with 200 demonstration trajectories from each training environment. After imitation learning, they are trained using reinforcement learning with 10M frames. ANS model is also trained with 10M frames. Further implementation details of the proposed method and the baseline models are provided in the supplementary material.

5.2. Experimental Settings

We conducted experiments on the Habitat simulator [22] with the Gibson dataset [35]. All models are trained with 72 scenes, and evaluated on 14 unseen scenes according to the split used in [22]. We evaluated each method in various difficulty settings. The difficulty of an episode is determined by the geodesic distance to the target location (easy: 1.5m$\sim$3m, medium: 3m$\sim$5m, hard: 5m$\sim$10m). We tested 1,007 sampled episodes for each difficulty level. The maximum time step of an episode is set to 500. We used an actuation noise model from [7]. The baselines requiring a pose sensor are given a noisy sensor, following the practice used in [7].

5.3. Evaluation Metrics

Two evaluation metrics are used: the success rate and success weighted by path length (SPL) [1]. The navigation becomes a success when the agent takes the stop action within 1m from the target location. SPL represents the efficiency of a navigation path and it is calculated as SPL $= \frac{1}{E}\sum_{i=1}^{E} S_i \frac{l_i}{max(p_i, l_i)}$, where E is the total number of evaluation episodes, $S_i \in \{0, 1\}$ represents whether the agent succeeded ($S_i = 1$) the i-th episode or not ($S_i = 0$),

Methods	Memory Type	Need Pose info.	Easy SR	Easy SPL	Medium SR	Medium SPL	Hard SR	Hard SPL	Overall SR	Overall SPL
CNN + LSTM [37]	hidden vector	yes	0.73	0.70	0.53	0.49	0.22	0.18	0.49	0.45
ANS + pred. target pose[7]	metric map	yes	0.74	0.21	0.68	0.23	0.30	0.11	0.58	0.18
Exp4nav [10]	metric map	yes	0.70	0.62	0.61	0.52	0.47	0.39	0.59	0.51
SMT [13]	stack	yes	0.82	0.77	0.66	0.52	0.56	0.40	0.68	0.56
Neural Planner [4]	graph	yes	0.72	0.41	0.65	0.39	0.42	0.27	0.60	0.36
Exploration + SPTM [27]	graph	**no**	0.67	0.41	0.64	0.39	0.42	0.25	0.58	0.35
NTS [8]	graph	yes	**0.87**	0.65	0.58	0.38	0.43	0.26	0.63	0.43
VGM (ours)	graph	**no**	0.86	**0.80**	**0.81**	**0.68**	**0.61**	**0.46**	**0.76**	**0.64**

Table 1: Comparison and evaluation results of the baselines and our model. (**SR**: success rate, **SPL**: success weighted by path length)

and l_i and p_i are the shortest path distance to the target location and the actual path length taken by the agent, respectively.

5.4. Quantitative Result

Comparison with the baselines. The overall results of the baselines and the proposed method are shown in Table 1. The proposed navigation framework brings performance improvement over other types of memory models. The CNN+LSTM model shows the poorest performance because of its limited size of the implicit memory. Compared to the metric map-based model (Exp4nav), VGM shows an improvement of the success rate by 28.8% (from 0.59 to 0.76) and SPL by 25.5% (from 0.51 to 0.64). The metric map can contain the overall structure of the environment, but it cannot represent possible target regions. On the contrary, VGM can provide the information about what nodes are similar to the target observation.

Comparing the results of SMT and VGM, we can see that the graph structure of the memory significantly helps the agent to find a target with a smaller number of memory elements. On average, VGM uses 96% fewer memory elements than SMT (for more information, see the supplementary material). The success rate is increased by 11.7% (from 0.68 to 0.76) and SPL is increased by 14.3% (from 0.56 to 0.64) from the SMT model.

The handcrafted models with an exploration policy (ANS, Neural Planner and SPTM) shows relatively lower SPL. The graph-based methods (Neural Planner, SPTM) can robustly navigate between the nodes in the noisy environment, but they can not actively search for the target beyond the given graph. Consequently, these models depend highly on the performance of the exploration policy and the target estimator. However, the exploration policy only looks for novel places rather than the target-like places. In contrast, our method looks for the places that the target might present by expanding the graph memory, so it can find the target much faster.

Our method also benefits from end-to-end structure that can utilize much larger dataset from RL. Note that NTS [8] uses smaller dataset (300 demonstrations per env.) than ours (200 demonstrations per env. + 10M RL frames). We provide experiment results with limited sizes of dataset in the supplementary material.

Importantly, the noise of the pose sensor affects the performance of the other models. Figure 3 shows the model performances at different noise levels tested on scenarios with hard difficulty. The average absolute error $(x, y, orientation)$ of Level 1 is $(0.22cm, 1.37cm, 0.089\,\text{deg})$ and Level 5 is $(1.9cm, 12.31cm, 0.81\,\text{deg})$ with equal intervals from Level 1 to Level 5. Level 0 is when the ground-truth robot pose information is given. We can see that the metric-map based models (ANS, Exp4nav) are highly dependent on the pose noise levels. The Neural Planner also shows a performance drop because it uses the positions of nodes to calculates the distance between them. These distances can be inaccurate due to noise and affect negatively to path planning. Also, using a target pose estimator rather than the ground-truth target position lowers the performances of ANS and Neural Planner as shown in Figure 3 (at Level 0). Our model is highly robust against the pose sensor noise and the pose estimation error since the pose information is not used during navigation.

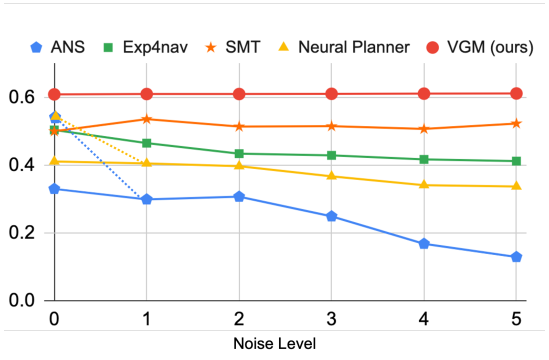


Figure 3: Experiments with various noise levels of pose sensor. Level 0 is when the ground-truth position of the agent is given. The dotted lines of the ANS and Neural Planner denote when the ground-truth position of target location is also given.

Localization. We measured the localization accuracy of the memory update module across various s_{th}. Localization succeeds when the memory update module correctly selects the closest node from the current location. All episodes are sampled from episodes with hard difficulty. The results are shown in Table 2. The proposed method shows robust localization even though the image encoder is trained using unsupervised representation learning.

We also measured the average distance between nodes and the performance of the agent across various s_{th}. The

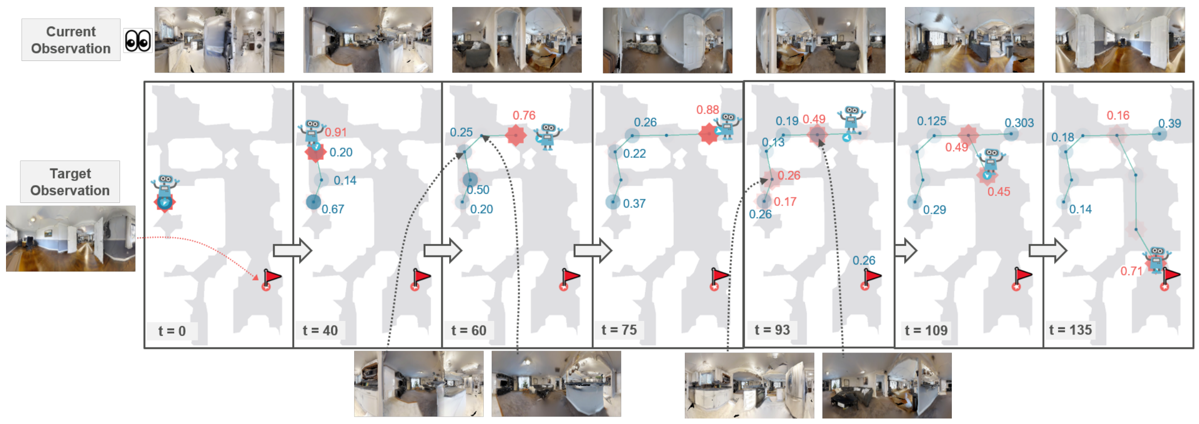


Figure 4: Example visualization of a target searching episode. To visualize the behavior of the agent, we plot the VGM in the top-down map of the environment. Each node in the map denotes the node elements which is added throughout the path of the agent. Additionally, we visualized the attention scores of nodes as blue circles and red stars. The blue circle is the attention score in the context of current observation in F_{dec1}. The red star is the attention score in the context of the goal observation in F_{dec2}. More vivid the color, the higher the attention score. We wrote the attention score next to the node in each corresponding color if the value is over 0.1. Best shown in color.

s_{th}	SR	SPL	Avg. dists	Loc. acc.
0.30	0.60	0.41	4.21	0.97
0.45	0.58	0.42	3.41	0.94
0.60	0.60	0.44	2.55	0.93
0.75	0.61	0.46	1.69	0.94
0.90	0.59	0.44	0.86	0.93

Table 2: Success rates and SPLs at different s_{th}. The average distances between nodes and localization accuracies are also shown in the table.

average distance increases as s_{th} becomes smaller since the memory update module acknowledges a larger area as a neighborhood of a node. The low value of s_{th} does not affect the success rate significantly, but it slightly decreases SPL. We believe that this difference arises from the different amounts of information in the VGM. When we use small values, there is less information available to the agent because fewer nodes are added to the VGM.

5.5. Qualitative Result

We show an example of episodes from target search in Figure 4. The agent has found the target location after heading to some wrong places. Attention scores for c_t are usually distributed to the nodes far from the agent. In contrast, attention scores for c_{target} are usually concentrated on the nodes near the agent. An interesting point is that the attention scores for c_{target} are spreaded when the agent returns to a previously visited node. For example, the agent headed to a room at $t = 60$. However, it has found that the room is different from the target location and comes out from the room at $t = 93$. At $t = 93$, the attention scores are distributed to the previously visited nodes. It appears that the agent is trying to find a potential target region using the VGM. Usually, the agent conducts an exploration policy, but when the agent returns to the place where it has been before, it looks for the next explorable nodes by attending to nodes in the VGM.

In supplementary material, we provide additional analyses on the proposed method. (1) We ablated each element in the navigation module, such as F_{vis}, F_{enc} and F_{dec}. We found that every element is helpful to improve the navigation performance. Especially, F_{vis} plays an essential role for the agent to learn navigation skills, such as obstacle avoidance. (2) Considering the scalability issue, we tested the proposed navigation method with limited memory budgets and found that the proposed method can conduct a given navigation task with a significantly smaller memory budget. (3) We also ablated auxiliary losses and unsupervised representation. The results show that auxiliary tasks improve the performance; however, the proposed method still outperforms other baselines even without the auxiliary tasks. We also found that training F_{loc} with supervised learning further brings the performance improvement of the proposed method (about 8% improvement). (4) More examples of navigation episodes and generated graphs are also provided in the supplementary material. (5) We have also tested the proposed navigation framework on the coverage task. Our method shows competitive performance with other baselines. Detailed experiment results and discussions are preseneted in the supplementary material.

6. Conclusion

This paper proposes a new type of navigation memory, visual graph memory (VGM). The VGM is built online during the navigation with an image encoder, which is trained using an effective unsupervised representation learning method. Also, the proposed navigation framework with the VGM has shown to learn a goal-oriented policy directly from the graph representation. The experiment results have shown that the proposed method has achieved 11.8% improvement in success rate, and 14.3% improvement in SPL over the state-of-the-art methods with various types of navigation memories.

References

[1] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 6

[2] Henrik Andreasson and Tom Duckett. Incremental Robot Mapping with Fingerprints of Places. *IFAC Proceedings Volumes*, 2004. 1, 2

[3] Adrien Angeli, Stéphane Doncieux, Jean-Arcady Meyer, and David Filliat. Visual topological SLAM and global localization. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009. 1, 2

[4] Edward Beeching, Jilles Dibangoye, Olivier Simonin, and Christian Wolf. Learning to plan with uncertain topological maps. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 4, 6, 7

[5] David M Bradley, Rashmi Patel, Nicolas Vandapel, and Scott M Thayer. Real-Time Image-Based Topological Localization in Large Outdoor Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005. 1, 2

[6] Devendra Singh Chaplot, Abhinav Gandhi, Dhiraj Gupta, and Ruslan Salakhutdinov. Object Goal Navigation using Goal-Oriented Semantic Exploration. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2

[7] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning To Explore Using Active Neural SLAM. In *International Conference on Learning Representations (ICLR)*, 2020. 2, 6, 7

[8] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural Topological SLAM for Visual Navigation. In *IEEE Conference on Computure Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 4, 6, 7

[9] Kevin Chen, Juan Pablo de Vicente, Gabriel Sepulveda, Fei Xia, Alvaro Soto, Marynel Vázquez, and Silvio Savarese. A Behavioral Approach to Visual Navigation with Graph Localization Networks. In *Robotics: Science and Systems*, 2019. 1, 2

[10] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning Exploration Policies for Navigation. In *International Conference on Learning Representations (ICLR)*, 2019. 2, 6, 7

[11] Zhiwei Deng, Karthik Narasimhan, and Olga Russakovsky. Evolving Graphical Planner: Contextual Global Planning for Vision-and-Language Navigation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2

[12] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 1989. 2

[13] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene Memory Transformer for Embodied Agents in Long-Horizon Tasks. *IEEE Conference on Computure Vision and Pattern Recognition (CVPR)*, 2019. 2, 6, 7

[14] Patrick Foo, William H Warren, Andrew Duchon, and Michael J Tarr. Do humans integrate routes into a cognitive map? Map-versus landmark-based navigation of novel shortcuts. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 2005. 1

[15] Friedrich Fraundorfer, Christopher Engels, and David Nistér. Topological mapping, localization and navigation using image collections. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007. 1, 2

[16] Sabine Gillner and Hanspeter A Mallot. Navigation and acquisition of spatial knowledge in a virtual maze. *Journal of cognitive neuroscience*, 1998. 1

[17] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*, 2017. 4

[18] Benjamin Kuipers. The "map in the head" metaphor. *Environment and Behavior*, 1982. 1

[19] Ashish Kumar, Saurabh Gupta, David Fouhey, Sergey Levine, and Jitendra Malik. Visual Memory for Robust Path Following. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018. 2

[20] Dong Li, Qichao Zhang, Dongbin Zhao, Yuzheng Zhuang, Bin Wang, Wulong Liu, Rasul Tutunov, and Jun Wang. Graph Attention Memory for Visual Navigation. *arXiv preprint arXiv:1905.13315*, 2019. 2, 4

[21] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven C.H. Hoi. Prototypical Contrastive Learning of Unsupervised Representations. In *International Conference on Learning Representations (ICLR)*, 2021. 4

[22] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 6

[23] Xiangyun Meng, Nathan Ratliff, Yu Xiang, and Dieter Fox. Scaling Local Control to Large-Scale TopologicalNavigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 1, 2, 4

[24] Lina Mezghani, Sainbayar Sukhbaatar, Arthur Szlam, Armand Joulin, and Piotr Bojanowski. Learning to Visually Navigate in Photorealistic Environments Without any Supervision. *arXiv preprint arXiv:2004.04954*, 2020. 1, 2, 4

[25] Michael J Milford, Gordon F Wyeth, and David Prasser. RatSLAM: A Hippocampal Model for Simultaneous Localization and Mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2004. 1, 2

[26] Yiding Qiu, Anwesan Pal, and Henrik I. Christensen. Learning Object Relation Graph and Tentative Policy for Visual Navigation. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[27] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric Topological Memory for Navigation. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2, 4, 6, 7

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5

[29] Adriana Tapus and Roland Siegwart. Incremental Robot Mapping with Fingerprints of Places. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005. 1, 2

[30] Sebastian Thrun. Probabilistic Robotics. *Communications of the ACM*, 2002. 2

[31] Edward C Tolman. Cognitive maps in rats and men. *Psychological Review*, 1948. 1

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 5

[33] Ranxiao Frances Wang and Elizabeth S Spelke. Human spatial representation: Insights from animals. *Trends in cognitive sciences*, 2002. 1

[34] Yi Wu, Yuxin Wu, Aviv Tamar, Stuart Russell, Georgia Gkioxari, and Yuandong Tian. Bayesian Relational Memory for Semantic Visual Navigation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[35] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: Real-World Perception for Embodied Agents. In *IEEE Conference on Computure Vision and Pattern Recognition (CVPR)*, 2018. 6

[36] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual Semantic Navigation using Scene Priors. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[37] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 6, 7