2020 20th International Conference on Control, Automation and Systems (ICCAS 2020)
Oct. 13∼16, 2020; BEXCO, Busan, Korea

Learning to use Topological Memory for Visual Navigation

Obin Kwon[1] and Songhwai Oh[1]

[1]Department of Electrical and Computer Engineering and ASRI,
Seoul National University, Seoul, Korea
{obin.kwon, songhwai.oh}@rllab.snu.ac.kr

Abstract: We propose a new method of using topological memory in visual navigation problem. In our method, an agent builds a topological map during the navigation and simultaneously reasoning on the built map. The proposed method can efficiently use the elements of the graph memory using GCN and Transformer network. We evaluated our method on the visual target navigation problem which need some exploration strategies. The agent is trained using Deep Reinforcement Learning in photo-realistic Habitat simulator with Matterport 3d dataset. Using much smaller memory than the baseline, our method achieved competitive performance in visual target navigation problem.

Keywords: Visual Navigation, Topological Map, Deep Reinforcement Learning

1. INTRODUCTION

An episodic memory which contains information about the trajectory and the overall environment is a crucial element for an efficient robot navigation task. Topological memory has been a popular option as a type of external memory [1], [2], [3]. It resembles how the human remember about the environment and also, it requires much smaller computation then building a metric map. However, many of the topological map based approaches are using fixed-size pre-built graph. This limitation makes them hard to be implemented on unseen envrionment as they need certain amount of experiences about the environment.

We propose a new method to learn a navigation policy with efficient and effective usage of topological memory. While building the topological memory online, our navigation policy can reasoning on the whole memory and determine which part is important. The overall system structure of our proposed method is shown in Fig. 1. As the memory itself can be a state and the action policy network can takes the memory state as an input, our navigation policy can be trained using Reinforcement Learning.

The proposed method consists of 3 parts. 1.Memory module which build a topological memory, 2.Embedding module which embeds the memory using GCN and Transformer Network, 3.Action policy which decides the action based on the embedded memory.

We evaluated our method on image target navigation. An agent needs to navigate to the target point given the observation image at the target point. As neither the path nor the position information is not given, the agent needs to explore the environment to find appropriate region. Additionally, it is important to remember its trajectory for efficient search. Experiment results show that the proposed method is more effective and efficient than other baselines.

This work was supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2019-0-01309, Development of AI Technology for Guidance of a Mobile Robot to its Goal with Uncertain Maps in Indoor/Outdoor Environments)

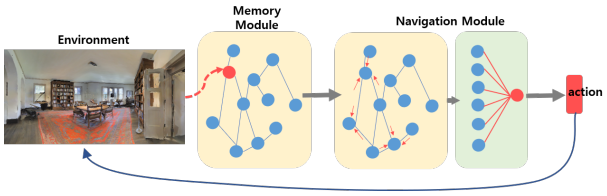

Fig. 1 Overall system architecture of the proposed navigation method.

2. TASK SETUP

This paper focuses on the image-target search task among various visual navigation tasks. This task requires high-level semantic reasoning with long-horizon navigation. We consider an episodic environment where an robot agent is randomly spawned in random environment. Every start of the episode, the agent is given the target observation image from the random location. The objective of this task is to successfully navigate to the target position. RGBD panoramic image is the only observation that agent can see, and no geometric information is provided such as robot position. There is no guarantee that the target location can be seen from the start position. Unlike simple navigation tasks like obstacle avoiding or path following, the agent needs to remember the navigation history for efficient exploration in the target search task.

3. METHOD

In this section, we first explain how the topological memory is built online in Memory Module. Then, we introduce our proposed method dividing into three parts;Encoder, Decoder and Actor.

3.1 Memory module

The visual encoder F_{CNN} and the localization network F_{Loc} are the main components of this module. These networks are used as an siamese network[4] which determines the similarity of the two image observations. Given the observation $o_t \in R^{4\times h\times w}$, F_{CNN} encodes the

image into a embedding vector $e_t \in R^d$. This embedding vector will be the feature embedding of the node in the topological memory $G \in R^{N \times d}$. Every start of the episode, the memory module initialize G with the embedding of initial observation e_0. As the agent moves, the memory module receives the flow of the observations and gradually constructs the topological memory as well as localize the agent.

Memory update process is done as follows. When a new embedding vector e_t is given from the F_{CNN}, the localization network F_{Loc} compares e_t with the nodes in the memory. F_{Loc} takes concatenation of two embedding vectors and outputs the similarity of them. If the similarity is larger than threshold, the two observations are considered to be close to each other. If any similar node is found, the memory module updates the localized node with e_t and records the visited time. Additionally, new edge between previously localized node and current localized node is added.

$$close = F_{Loc}([e, e_t]) > \epsilon \tag{1}$$

If none of the nodes in memory are similar with e_t, a new node is added to memory.

After the memory update process, memory module returns the updated topological memory G and index of the localized node where the agent is.

3.2 Navigation Module

Navigation Module has an Encoder-Decoder structure. First, the Encoder embeds the nodes in the memory with regards to each neighbors of them. Then, the Decoder further embeds the memory with the current observation.

3.2.1 Encoder

As the built memory has a graph structure, we use Graph Convolutional Network (GCN)[5] to encode the memory. Single graph convolutional layer can be represented as eq2.

$$GCN(V, A; \theta) = \sigma(\tilde{A} V W_\theta) \tag{2}$$

$\tilde{A} \in R^{N \times N}$ is a normalized adjacency matrix with self-loop, and $W_\theta \in R^{d \times d_m}$ represents the parameters of GCN. The feature embedding of each nodes are updated with θ and aggregated with the features of its neighbors. Through the multiple graph convolutional layers, the node embedding contains the information about its own observation and its multi-hop distant neighbors. As we handle the visual target navigation, we first encode each embedding of nodes with target embedding using a single Fully-connected layer. Therefore, the Encoding process can be formulated as eq 3.

$$Enc(V, A; \theta) = GCN^n(FC([V; e_{target}], A)) \tag{3}$$

3.2.2 Decoder

Decoder receives the output of Encoder, and embeds it with regard to the current observation. We use the Transformer network[6] to obtain related information about the current observation from the encoded memory. Let $W^q \in R^{d \times d_q}$, $W^k \in R^{d_m \times d_k}$, $W^v \in R^{d_m \times d_k}$ are each parameter matrices for query, key and value. F_{vis} encodes the current observations to $s_t = F_{vis}(o_t)$. (Note that we use 2 different visual embedding networks F_{vis} and F_{CNN} as each network conducts different task.) The current observation $s_t \in R^d$ is a query and the the encoded memory $M \in R^{N \times d_m}$ is the key and value. Eq4 shows the Scaled dot-product attention using them.

$$Attention(e_t, M) = softmax(\frac{e_t W^q (M W^k)^T}{\sqrt{d_k}}) \tag{4}$$

We use Multi-head attention to diversify the representations. The output of multi-head attention further processed with residual connection and layer normalization.

$$\begin{aligned} M' &= MultiHeadAttention(e_t, M) \\ C &= LayerNorm(M' + e_t) \\ C &= FC(C) \end{aligned} \tag{5}$$

Additionally, time information is added to each memory elements, which is equivalent to positional encoding in Transformer[6] network. Every time when the node in memory is updated, visited time step of its node is recorded. Using this visited time as the order of the elements in the memory, we add sinusoidal positional encoding to M.

3.2.3 Actor

Actor takes the decoded memory C from the decoder, and visual features of the current observation and the target observation s_t, s_{target}. Actor consists of two Fully Connected layers and single LSTM layer. This network outputs the distribution of action a_t based on the concatenated features.

$$\begin{aligned} v_t &= FC(ReLU(FC([C, s_t, s_{target}]))) \\ h_t &= LSTM(v_t, h_{t-1}) \\ p(a_t|h_t) &= FC(h_t) \end{aligned} \tag{6}$$

3.3 Implementation Details

F_{vis} and F_{CNN} is Resnet18, and F_{loc} is 2 fully connected layers. F_{CNN} and F_{loc} in the memory module are pretrained as a siamese network with the random sampled observations from the training environments. Let $\{e_t, e_{pos}, e_{neg}\}$ are the outputs of F_{CNN} when taking the observations triplet pair $\{x_t, x_{pos}, x_{neg}\}$

$$L_{triplet}(\theta) = max(0, m + d(e, e_{pos})) - d(e, e_{neg}))$$

$$L_{classify}(\theta) = \log(F_{loc}([e, e_{pos}])) + \log(1 - F_{loc}([e, e_{neg}]))$$

$$L_{total} = L_{triplet} + L_{classify}$$

$L_{triplet}$ is triplet loss and $L_{classify}$ is classification loss. With these loss functions, the two networks are trained to classify whether the two observations is close or not while satisfying the feature distance condition. These networks are freezed during navigation policy training.

The Encoder Network process the graph memory using 3 GCN layers. The hidden feature dimension in GCN as well as queue, key, value dimension in Transformer are all set to 512.

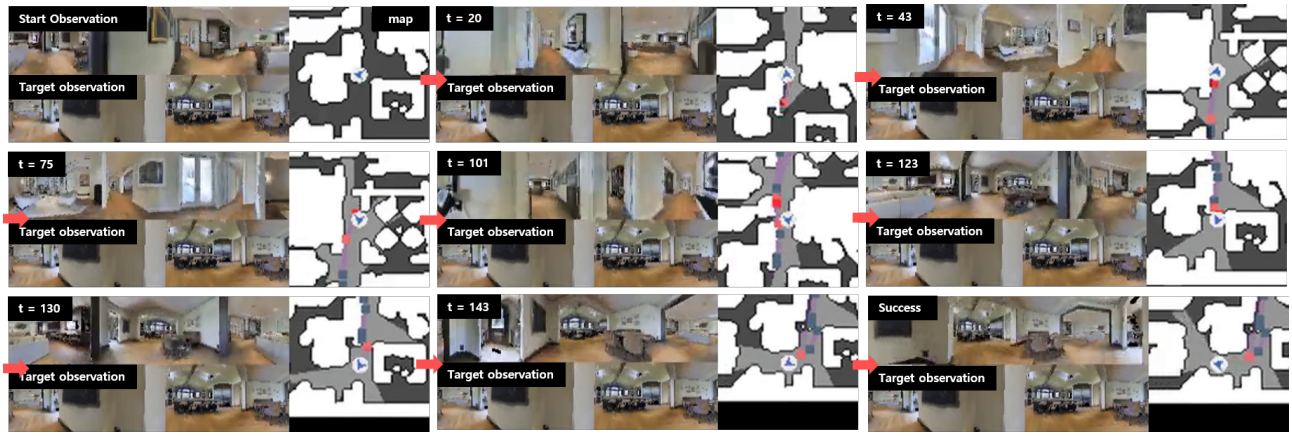

Fig. 2 Visualization of single evaluation episode. We sampled some time steps that well represent the situation. In each time step block, the image in the left top is the image observation shown to the agent. Target observation is in the left bottom. In the map, each square represent the node in the graph memory. Purple line represent the edge between the nodes, and the red nodes are localized node from the memory module.

We train the navigation module with Proximal Policy Optimization(PPO)[8] algorithm. Reward function is set as dense reward proportional to the progress distance to the target location. λ is a scalar constant to adjust the reward scale.

$$r_t(s,a) = \lambda(dist_{t-1} - dist_t)$$

Small penalty (-0.01) is added every time step to encourage faster search. Action space of the agent is $\{\text{move\_forward}, \text{turn\_left}, \text{turn\_right}\}$.

4. EXPERIMENT

4.1 Experiment setup

We conducted experiments on the Habitat simulator[9] with Matterport dataset[10] Two baselines are compared with the propose method. The first is a simple LSTM policy without memory. The second is [7] which is similar with our method in that it does not uses any metric map or estimated geometric information. All the baselines are trained in 61 scenes from Matterport dataset, and evaluated on 11 unseen scenes. We evaluated each methods using 1000 sampled episodes in 11 scenes. We sampled the episode that the target distance is between 3m to 5m. An agent need to find the target location based on the given observation. The episode is considered as success if the agent reach the target position within 1m before the episode ends. We set the memory size of SMT to 500 which is equivalent to the maximum time step of each episode. As our method selectively stores the observations, maximum number of nodes of ours is set to 100. Evaluation metrics are success rate and SPL(Success weighted by Path Length), which is proposed in [11]. SPL is a measure of navigation efficiency and it can be calculated as eq 7. N is a total number of evaluation trajectory, and l_i and p_i are shortest path distance and the actual path length taken by the agent.

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(p_i, l_i)} \tag{7}$$

4.2 Experiment Result

Figure 2 shows an example of single evaluation episode. An agent is spawned in unseen environment with the target observation. The agent navigates to the target position by building a topological memory. Map in the figure is only for the visualization, the agent can not see the map of the environment. We can see that the agent initially headed to opposite direction from the target, but it returned to the first location and succeeded to find the target location. Tables 1 shows the results of the experiments.

Table 1 Success Rate(SR) and SPL of each methods.

Metrics	LSTM	SMT[7]	**Ours**
SR	0.448	0.583	**0.588**
SPL	0.293	0.335	0.316

The results shows that the use of the memory yields higher success rate. Further, we can see that the proposed method achieves similar performance while using a much smaller amount of memory than the baseline.

5. CONCLUSION

This paper introduced a new navigation method which can build and utilize a topological memory. Using a simple siamese network, an agent can store the observation selectively, and the required memory capacity and computational cost can be reduced. The experiment results show that, even with much smaller memory size, the propose method can achieve comparable results with the baseline which uses a lot of memory.

REFERENCES

[1] Nikolay Savinov, Alexey Dosovitskiy and Vladlen Koltun, “Semi-parametric topological memory for navigation,” In Proc. of *International Conference on Learning Representations (ICLR)*, 2018.

[2] Kevin Chen, Juan Pablo de Vicente, Gabriel Sepulveda, Fei Xia, Alvaro Soto, Marynel VÃ¡zquez,

Silvio Savares"A Behavioral Approach to Visual Navigation with Graph Localization Networks", In Proc. of *Robotics: Science and Systems*, 2019.

[3] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta, "Neural Topological SLAM for Visual Navigation", In Proc. of *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020

[4] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov, "Siamese neural networks for one-shot image recognition," In Proc. of *International Conference on Machine Learning (ICML)*, 2015. siam

[5] Thomas Kipf, Max Welling, "Semi-Supervised Classification with Graph Convolutional Networks", In Proc. of *International Conference on Learning Representations (ICLR)*, 2017.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is All you Need" In Proc. of *Advances in neural information processing systems*, pp. 5998-6008, 2017.

[7] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese, "Scene Memory Transformer for Embodied Agents in Long-Horizon Tasks", In Proc. of *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 538-547, 2019.

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

[9] 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 Proc. of *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[10] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, Yinda Zhang, "Matterport3D: Learning from RGB-D Data in Indoor Environments", In *International Conference on 3D Vision (3DV)*, 2017.

[11] 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.