FACTORING METRICS INTO PANORAMIC TOPOLOGICAL MEMORY FOR NAVIGATION

Siyuan Zhang
Department of Computer Science
University of Illinois at Urbana-Champaign
siyuan3@illinois.edu

ABSTRACT

The usage of a topological map as memory structure for the task of indoor visual navigation has been restudied recently thanks to the advancement of deep learning. However, previous approaches formulated the problem more as a path memorization problem from sequence of observations. This resulted in limited representation of the rich indoor environment and lack of connectivity while being prone to problems such as scalability and aliasing. We introduce a new form of topological memory architecture where each vertex is a panoramic view of the surroundings and each edge contains relative pose changes between two close-by vertices. We show that panoramic views enables better connectivity while alleviating aliasing. At the same time, edge metrics strengthens the capability of local control and enriches the memorized representations. The combined navigation agent outperforms previous solutions in many aspects.

1 INTRODUCTION

Recently, there has been a surge of work in using learning based methods for the task of visual navigation thanks to the advancement of deep learning. Unlike the traditional methods such as SLAM Mur-Artal et al. (2015) which are sensitive to sensor calibration and image qualities, learning based methods can leverage the rich semantic information from RGB inputs and construct maps beyond pure geometric reasoning. Recent work has shown that embodied navigation agents with dedicated memory structure for storing learned internal map representation of the environments outperforms agents that are purely reactive or only using short term memories such as LSTM. This phenomenon is more observed in larger scale environments and with more complicated tasks beyond just obstacle avoidance.

Most of the approaches try to learn a metric map representation, which has the advantage of precise local metrics of nearby surrounding. However, such solutions often assumes ground truth coordinate or ego-motion information with prefect actuation which becomes impractical in the real-world. Strong psychological evidence Thrun (1998) suggest that human and animals have a topological graph representation of the environment and uses landmarks as reference to accurately executes planned paths.

Savinov et al. (2018) introduced semi-parametric topological memory which memorizes sequence of observations as a topological graph and learns to navigate from observations to observations. While this work has shown success in robust path following in well explored environments, there are still many significant limitation of this framework: i) The input view angle is narrow resulting in short-cuts between paths to be detectable only when they are in the same direction ii) It suffers from aliasing observations when localizing iii) it scales linearly with the size of experience which become problematic in larger environments. Moreover, metric information are completely ignored when constructing the memory graph which we show can be quite useful in several ways.

Motivated from the representation used by human when showing indoor environments, in this work, we propose a modified formulation of learning topological navigation that resolves many of the limitation previous method faces while increase scalability and enables richer representation. One the key modification we propose is to replace the narrow view angle image at each vertex with a panoramic view observation. The main advantage we believe this change would bring is the

improved connectivity of the learned graph. Previously used narrow angle view observation as a vertex can only become a point in a path that travels in the same direction as the observation faces. This significantly limited the capacity of the topological memory to memorization of paths that was traveled during the map construction. By replacing with panoramic views, each vertex can effectively connect paths in all directions resulting in improved connectivity, higher vertex utilization and optimized path planning.

Another main change we propose is to factoring metric information into the topological representation. We believe even just noise ego-motion information from adjacent observations in the trajectories can greatly improve the performance of local controller. Without any metric information, previous approach relies on close-by observations to learn the transition dynamics and pose change from feature shifts in order to select actions. However, such noisy estimation can the very easily collected from sensors such as inertia measurement unit(IMU) which is very common on modern robot platforms. With the rough estimation of pose change required to reach the next point on the path, local controller can reach locations farther away from current observation with reliability. This advantage allows to learned memory to be further sparsified thus increase scalability.

Less visual aliasing was observed thanks to more information contained in panoramic views and spasified vertices resulting less similar looking observations. We evaluate our approach in the Gibson simulation environment which contains realistic 3D indoor environments of houses. We show that our approach outperforms the baselines in may aspects and the learned topological memory solves many previous limitations.

2 RELATED WORK

There has been a concentration of interesting recently on deploying learning based method for indoor visual navigation. Some are purely reactive Dosovitskiy & Koltun (2016); Zhu et al. (2017) while some other methods deploy generic memory structures Mnih et al. (2015); Mirowski et al. (2016); Pathak et al. (2017). These methods lacks memory structures with great capacity thus limiting their ability for planning which is important for navigation in large and complex environments. Later research leveraged dedicated memory structure to learn a metric internal representation of the environment. Bhatti et al. (2016) apply DL algorithm onto a map generated by traditional SLAM; Gupta et al. (2017a) learns a internal ego-centric spatial reasoning through end-to-end learning while Parisotto & Salakhutdinov (2017) learns a global map; Fang et al. (2019) uses attention transformer. Most of the metric representation work assumed some level of ground truth coordinate information which is not practical.

Another line of work Savinov et al. (2018) focused on building a topological memory that does not rely on accurate metric readings. Such methods build a graph to represent the environment with vertices being locations and learns to navigate from vertex to vertex. Kumar et al. (2018) studied path following and Gupta et al. (2017b)uses sparse observation and pose information for navigation. Chen et al. (2019) formulated the transition between vertices as a behavior problem. Meng et al. (2019) studied the scalabilty problem by learning the capability of local controller. There has also been an interest in using panoramic views for navigation Hirose et al. (2019).

3 TASK SETUP

The problem we study here is to learn an internal map representation through exploration of the environment and later reuse the learned memory for path planning and execution. During the exploration phase, the agent receives discrete time dense observations from expert guided traverse trajectory of the environment. This sequence of observation $\{o_1, ..., o_N\}$ is only presented once and the agent must learn its internal map solely from it. In addition to RGB input at each time step, a noisy sensor reading of the relative pose change from last to current observation is given to the agent. The ground truth pose change can be formulated as $\Delta p = (x, y, \theta)$ where x, y are the coordinate translation and θ is the rotation. The agent is presented with $\Delta\hat{p} = (x + \epsilon_x, y + \epsilon_y, \theta + \epsilon_\theta)$ where $\epsilon_x, \epsilon_y, \epsilon_\theta$ are random Gaussian noises with 0 means, ϵ_x, ϵ_y has a variance of 0.03m and ϵ_θ has a variance of 3 degree. During task time, the agent is placed at a starting point in the environment and is given a panoramic goal observation. The agent needs to localize itself and the goal in the learned environment, plan a

Figure 1: An illustration of the learned panoramic topological memory(PTM) after exploring the environment. Each vertex is a recorded panoramic view observation with center to be the front face direction of the agent. Each edge records the relative pose difference required to travel from one to another via local controller.

path and then navigate itself to the goal in limited time frame. Only reaching the goal location under the given time limit is counted as a success episode.

We try to use realistic simulation environments to improve generalization when deploying such model in real world robots in the future. Due to limited resource, our agent is trained, validated and tested in the Gibson environment with the tiny split from the Gibson data-set of indoor environment. The local controller has action space of $\{forward, left, right\}$ referring to going forward by 25cm, rotate counter-clockwise by 15 degrees and rotate clockwise by 15 degrees. The movement executed by the local control thus can be formulated as $\Delta u_a = (x_a, y_a, \theta_a)$ where $\Delta u_{forward} = (0.25, 0.0, 0.0)$, $\Delta u_{left} = (0.0, 0.0, 15 \times \pi/180)$, $\Delta u_{right} = (0.0, 0.0, -15 \times \pi/180)$. Since we assume noisy actuation of movements by the agent, we thus add Gaussian noises $\{\epsilon_x, \epsilon_y, \epsilon_\theta\}$ to movements as well similar to the sensory reading. Recent work from Chaplot et al. uses GMM to model both actuation and sensor noise from real world data, but in our context, such accurate pose estimation is not necessary as we leverage noisy ego information to provide heuristics to the local controller.

4 METHODS

4.1 PANORAMIC TOPOLOGICAL MEMORY

We propose a new form of memory for storing an internal representation of the learned environment, we refer as panoramic topological memory(PTM). Figure 1. shows an example of a learned PTM after exploration. A panoramic view of surrounding environment is stored at each vertex to serve as a connecting point among locations from different directions. The middle line of the panoramic view represents the direction in which the agent is facing when recording the observation. An edge $E_{1,2}$ is formed between two vertices(V_1, V_2) only when the local controller can guide the agent from one vertex to another in both directions. To increase the capability of the local controller, each edge also stores a rough estimate of the metric relationship (θ_1, θ_2, D) between two observations represented by the two vertices. For instance, if the agent is observing similar view as V_1, it can reach V_2's location by rotating θ_1 then move forward D given no obstacle in between. This provides an effective heuristic for the local controller, increase the distance between locations where it can reliably guide to.

4.1.1 LOCAL CONTROL NETWORK

The network C is trained to navigate towards close-by target observations. It takes input of form $(O_{current}, O_{target}, \Delta p)$, consists of the current observation, the goal observation and a rough estimate of the pose difference needed to reach target observation's. The current observations helps

Figure 2: (a) Architecture of the local controller C, localization network L is similar to it. (b) One iteration of rotation estimation by view angle estimator R, this process is repeated to obtain accurate estimation

the controller for obstacle avoidance while the goal observation and relative pose change serves as heuristics for planning. The output is a probability distribution over the action space and the highest probability action is executed.

The network is trained using reinforcement learning algorithm actor-critic. We pick random points along training trajectories to serve as staring and goal observations and reward at each step would be the reduction of Euclidean distance to the goal observation's coordinate. For each panoramic observation we generate 4 120^o FOV images equally spaced by 90 degrees to restore edge features, then pass through 5 layers of convolution layers to obtain an 1024 dimension embedding. The 8 embedding from the 2 panoramic view inputs are then concatenated with pose information and passed through 4 layers of fully connected network with ReLU activation and batch normalization ending with a 3-way softmax. The architecture is illustrated in Fig. 2(a).

4.1.2 Localization network

The localization network $L(O_1, O_2)$ estimate the probability that the local controller can successfully navigate between O_1 and O_2. This network serves couple purposes: i) localization of the agent during task time ii) connecting vertices during map construction iii) by measuring the capability of the local controller, we can sparsify the memory by removing trivial edges. Since this network measures the ability of the local control network, it is trained based on a trained version of the local controller.

The architecture of L is very similar to the local control network but without the pose estimation information and ends with a single probability output. The training data for L is directly generated from trained C by random sampling observations from space and label as 1 if C successfully navigates otherwise 0.

4.1.3 View angle estimator

The view angle estimator $R(O_1, O_2)$ estimates the view angle difference between two panoramic views. Since we are using panoramic view as vertices and pose metric for control, it is import to learn the rotation needed to align with observations. We formulate this as an iterative process of narrowing down the possible rotation range: at each iteration, 4 equally spaced 120^o view of one panoramic and 1 from the other are encoded, stacked then through FC ended with 4-way softmax. This effectively narrows down the possible rotation angle by four times and we repeat this process 3 times to narrow down the view angle to an accuracy of 3^o. To make the method more reliable, 2-way and 4-way cross matching can also be performed at the cost of efficiency. This cross validation can also be used to verify the localization as correctly localized observation should align in all directions. Training of this estimator is through direct supervision with randomly sampled close by panoramic observations with ground truth rotation. The workflow is illustrated in Fig. 2(b).

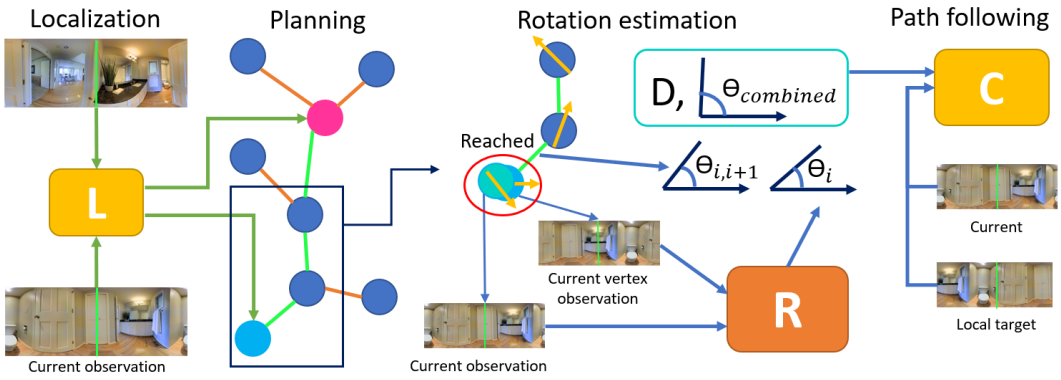

Figure 3: The navigation pipeline of the proposed method. After localization and planning the path is then executed. Upon reaching an anchor, the belief of relative poss change to the next anchor is updated.

4.2 Map construction

The map construction process during the exploration phase is incremental: given a new observation O_t, a vertex V_t is generated and an edge between O_t and O_{t-1} is automatically formed. The poss change estimation for this edge is calculated using noisy IMU input. In cluttered indoor environments, exploration sequence intersects and comes close by previous observations a lot, leaving potential for short cuts between paths to increase map connectivity. When new observation O_t is received, localization $L(O_t, O_i)\ \forall i < t - k$ are performed and an edge is formed when $L(O_t, O_i) > \Theta_{shortcut}$ where $\Theta_{shortcut}$ is some shortcut threshold. The pose estimation for shortcuts are formed using R for rotation and 0 for displacement. Each edge also stores the localization result between its two vertices.

In order to increase efficiency of the whole solution and build more compact representation, we would want to remove some vertices that are unnecessary. When a new vertex is introduced and shortcuts are formed from it, we applied the following vertex elimination decision to this new vertex and all vertices that connect to it directly: A vertex V_i will be removed when $L(O_j, O_k) > \Theta_{sparsify}$ $\forall i, j \in Adj_i\ s.t.\ i \neq j$. This will allow the number of vertices in a limited area environment to converge as the length of exploration experience growth as there is a maximum vertex density.

4.3 Navigation

At the beginning of a task episode, the agent localize itself to vertex $V_c = argmax_{V i \in V_1 ... V_N} L(O_i, O_{current})$ and similarly O_{goal} to V_{goal}. A path is planned through the topological graph using greedy Dijkstra's algorithm. Here the edge weight can the Euclidean distance if the goal is to minimize travel distance, or $-log(L(O_i, O_j))$ if the purpose is to increase navigation reliability, or a linear function of both. After the path is chosen, the agent will navigate itself along the path using the local controller. At each time step, the agent perform $L(O_{current}, O_{target}) > \Theta_{reach}$ to check if it has reached the next vertex in the planned path. Upon on arrival at the next vertex, the agent uses $R(O_{current}, O_i) = \theta_i$ to estimate the rotation difference between the current observation and the observation of the current vertex. The hint of relative pose change to the next vertex can be calculated as $(\theta_i + \theta_{i,i+1}, D_{i,i+1})$ where $\theta_{i,i+1}$ and $D_{i,i+1}$ are metrics stored in $E_{i,i+1}$. This is then feed to the local controller and updated at each time step using IMU reading. This process repeats until the goal vertex is reached. To handle navigation and localization error, if $L(O_{current}, O_j)$ is detected to be continuously decreasing for more than 5 time steps during navigation from V_i to V_j, global localization will be performed and a new path will be planned accordingly if the current vertex changes. The workflow is shown in Figure 3.

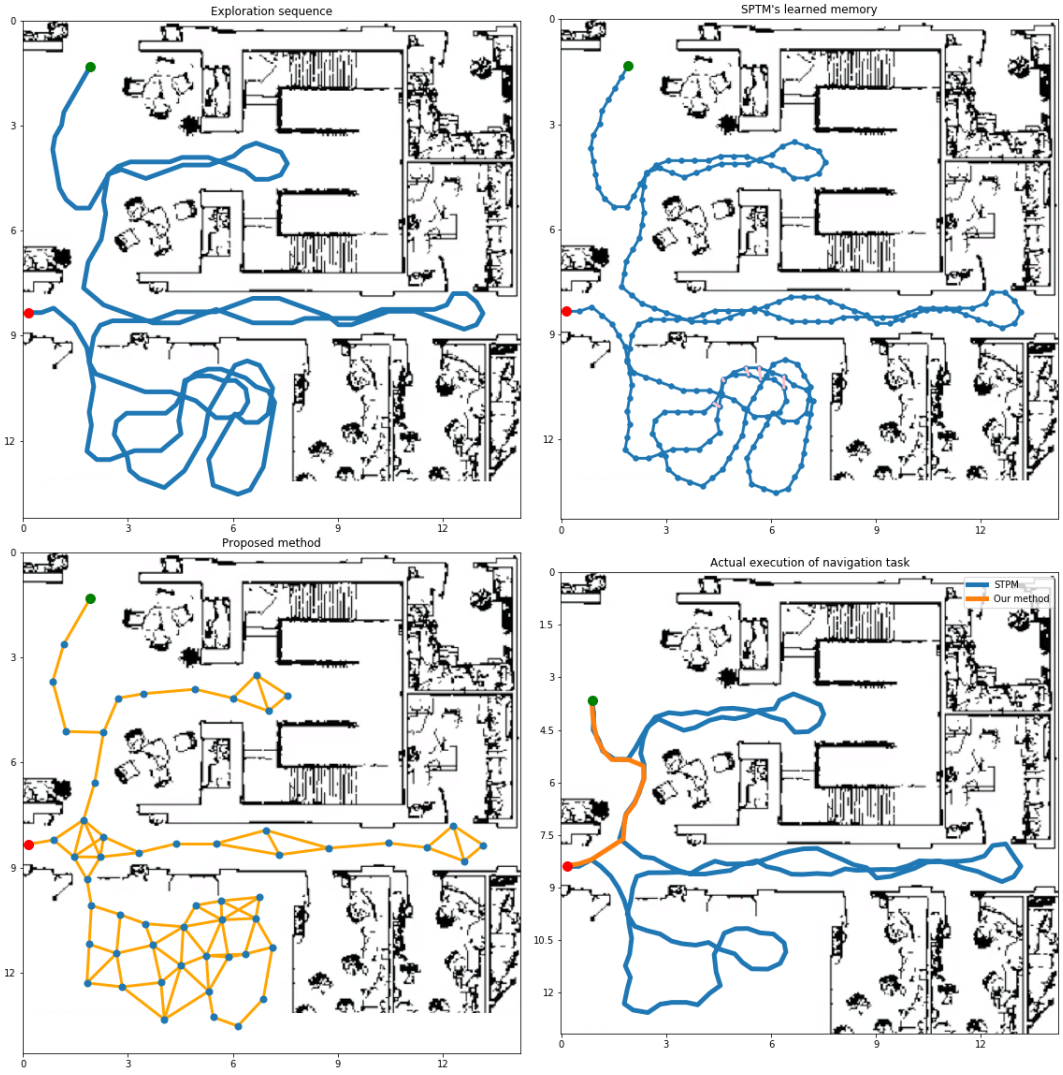

Figure 4: An example comparison between the internal map constructed by our method and the baseline SPTM. Our method builds a map with more connectivity, allowing close to optimal path planning and navigation. Out method is also efficient in computation and space as we generate less vertices to form the map. The result is more robust and optimal performance.

5 EXPERIMENTS

We trained C, L, R in the Gibson simulation environments using a selection from the small dataset split and 6 environments form the Stanford2D3DS dataset. The data-set consists of realistic 3D reconstruction of indoor house environments. The agent was trained using 20 environments and was tested on 5 environments that are disjoint with the training set. The observation space consists of RGB panoramic views of size $3 \times 128 \times 342$ and the IMU sensor readings are of size 3×1 to include 2-D coordinate and orientation change. We uses action space of $\{forward, left, right\}$ and assume noisy execution and sensor reading as discussed in Section 3.

We generate optimal trajectories using Dijkstra's algorithm and costmap optimization similar to Watkins-Valls et al. (2019). We generated 50k trajectories with random distant starting and ending point and sampled 1M pairs of panoramic observations along these trajectories for the training of C, L. R is trained by random sampling of 500k observations until convergence. The exploration trajectory are generated by continuous random sampling of points in unexplored areas of the maps and executing optimal paths navigating to the sampled location. The 5 layer CNN encoder takes

Table 1: Navigation success rate

Exploration length	500	1000	2000
SPTM	49.7%	64.8%	69.2%
SPTM $p_{sparsify} = 0.2$	3.8%	12.3%	15.4%
Ours $\Theta_{sparsify} = 0.6$	64.9%	79.1%	82.3%
Ours $\Theta_{sparsify} = 0.8$	81.8%	89.0%	91.2%
Ours $\Theta_{sparsify} = 0.9$	**82.5%**	**92.4%**	**93.9%**

Table 2: Learned map connectivity

	$r = 0.2m$	$r = 0.5m$	$r = 1.0m$
SPTM	32.3%	21.0%	3.5%
Ours $\Theta_{shortcut} = 0.8$	97.5%	89.0%	78.3%
Ours $\Theta_{shortcut} = 0.9$	96.8%	87.5%	64.2%
Ours $\Theta_{shortcut} = 0.95$	96.3%	81.5%	42.0%

input of 128×128 and outputs 1024 dimension embedding. The baseline we compare with is SPTM, a recent work on visual topological navigation which is closely related to ours.

5.1 NAVIGATION

Exploration sequence of different length is provided to the agent and then the agent performs goal-directed navigation. A goal location observation on average 15m away from the staring point is provided to the agent and the agent must navigate itself to the within 0.5 meter of the goal location under 90 seconds to be counted as a successful trajectory. The environments we tested on have an average area of $294m^2$. We performed 500 testing trajectory in each testing environment and results are averaged. Table 1. shows the success rate of different models with different length of exploration sequence. The graph of SPTM is also sparsified uniformly for testing. We see that our approach outperforms the baseline by a large margin thanks to increased local control reliability from panoramic view and pose estimation. We can see a significant performance drop from sub-sampling SPTM's memory. Our method perform relatively well even when in sufficient exploration is provided since out panoramic view greatly increase connectivity.

5.2 CONNECTIVITY

One major advantage of having panoramic views is the improvement of map connectivity. With only narrow angle view, the learned map is not really a graph but rather a memorization of different paths where short cuts are formed only when they are traveling in the same direction. The ideal topological graph should connect edges that are spatially close to each other. Our panoramic representation allows vertices to be connected with other close by vertices from all directions, providing more optimal path planning. We test the connectivity of observations that are spatially close in Euclidean distance and check if these pairs of vertices are connected by the learned topological memory of models. Exploration sequence of length 2000 is generated for 100 times for each testing environments and here r is the Euclidean distance between pair of vertices. Table 2. shows our model's connectivity outperforms the baseline by a large margin. As a result of that, the planned path in our model are generally shorter and closer to optimal compare to the baseline. Table 3. shows our method was able to find shorter path to the goal especially when less exploration sequences are provided. Figure 4. shows an example illustrating the advantage of having good connectivity.

5.3 SCALABILITY

Another big advantage of our method is the scalability. SPTM's memory size grows linearly with the size of observation and localization performed at each time step is directly proportional to memory size which makes it very problematic in large environments. SPTM also does not aggregate new experience with old ones and the space efficiency drop quickly as areas are repeated explored. Our

Table 3: Length of executed path compare to baseline

Exploration length	500	1000	2000
SPTM	100.0%	100.0%	100.0%
Ours $\Theta_{shortcut} = 0.9$	73.7%	84.6%	91.0%

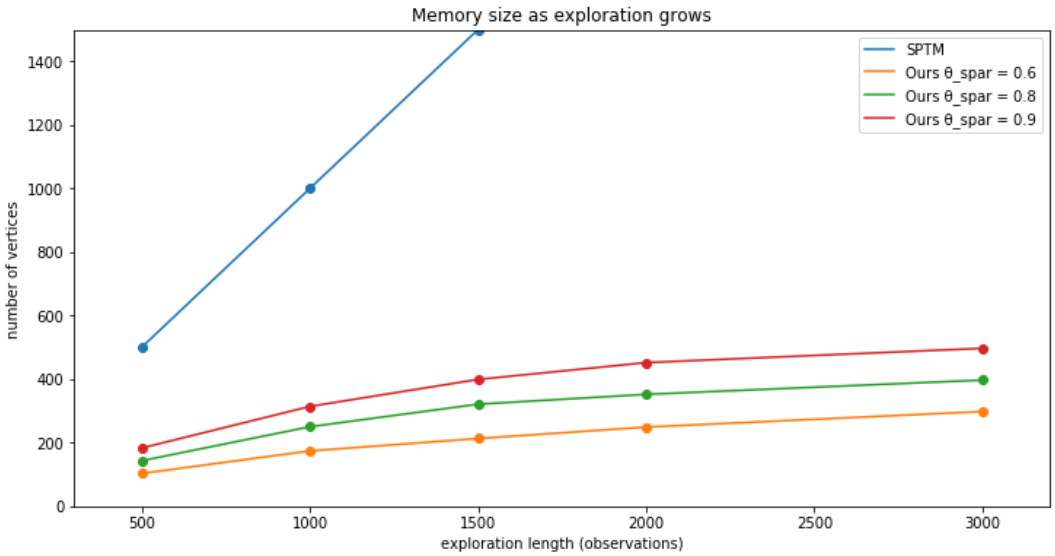

Figure 5: The size of the map memory as the length of exploration sequence grows.

method eliminates unnecessary edges and vertex count converges upon reaching maximum density. The local controller in our model is also more capable thus reduce the amount of observations needed to reliably navigate. We test the scalability by generating 100 exploration sequence for each test environment. Figure 5 shows the amount of vertices stored in the memory as the exploration sequence grows.

5.4 Localization accuracy

One of the main reason why we propose panoramic view over narrowed view angle is its ability to counteract visual aliasing which can be problematic in topological navigation. Intuitively, a panoramic view contains more information from more angles thus should improve localization performance. Moreover, our view angle estimator provided extra capability to verify localization accuracy. We tested different model's localization accuracy given 500 sample locations per observation sequences length for all test environments. We count as successful when the localized observation is within 1.5m of ground truth. The averaged result is shown in Table 4. This improvement helps our agent to make less mistakes during navigation and also resulting in less incorrect short-cuts during map construction.

Table 4: Localization accuracy

Exploration length	500	1000	2000
SPTM	91.0%	84.2%	85.3%
Ours	95.2%	96.2%	95.8%
Ours 2-way crs-vali	98.2%	98.1%	97.3%
Ours 4-way crs-vali	98.9%	98.2%	98.5%

Table 5: Results of local controller success rate ablation study

goal distance(m)	0.5 ± 0.2	1.0 ± 0.2	3.0 ± 0.2	5.0 ± 0.2
C	97.9%	95.2%	82.8%	44.9%
C w/o Δp	98.0%	93.5%	42.0%	23.1%

5.5 Ablations

Relative pose difference for local control. We introduced metric information to be factored into a topological graph, and our main reason is that such noisy estimation of where the local target is can improve the performance and reliability of our local controller. We performed ablation study by training a local controller with almost identical architecture compare to ours but without relative pose estimation information as input and trained using the same data. We then tested the capability of these local controllers on points randomly sampled on exploration trajectories by measuring their success rate to navigate to targets at different distances. 2000 pairs of points are generated per testing environment. We mark as success navigation if agent reaches within 0.5m of the goal location in 30 steps. Table 5. shows the ablation experiments result.

6 Conclusion

In this work we proposed a new form of topological memory for indoor navigation. By allocating a panoramic view observation to each vertex and factoring metric relationship into edges of the graph, we shown our method has many advantage. The panoramic view enhances the connectivity and perception while metric relationship improves the capability and robustness of local control, allowing our method to achieve better performance and scalability. We believe topological memory can bring great potential to navigation and planning especially in larger environments.

6.1 Future work

Although we factored metric information into the topological memory in this work, we believe topological and metric representation of the environment can be fused together more tightly, by having more layer of hierarchy. We are interested in having topological memory for high level planning and building connection between lower level segments metric spaces. It would be useful to develop methods that can connect metric spaces together through joints in the space such as doors. The metric map representation are ideal for exploration, optimal control and would be reliable if references can be added for better reuse. Another interesting topic is how to maintain and improve the internal map as the agent gains repeated experience over some space. We leave these as our future work.

References

Shehroze Bhatti, Alban Desmaison, Ondrej Miksik, Nantas Nardelli, N. Siddharth, and Philip H. S. Torr. Playing doom with slam-augmented deep reinforcement learning, 2016.

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. *arXiv preprint arXiv:1903.00445*, 2019.

Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future, 2016.

Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks, 2019.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2616–2625, 2017a.

Saurabh Gupta, David Fouhey, Sergey Levine, and Jitendra Malik. Unifying map and landmark based representations for visual navigation, 2017b.

Noriaki Hirose, Fei Xia, Roberto Martin-Martin, Amir Sadeghian, and Silvio Savarese. Deep visual mpc-policy learning for navigation. *IEEE Robotics and Automation Letters*, 4(4):3184–3191, Oct 2019. ISSN 2377-3774. doi: 10.1109/lra.2019.2925731. URL http://dx.doi.org/10.1109/LRA.2019.2925731.

Ashish Kumar, Saurabh Gupta, David Fouhey, Sergey Levine, and Jitendra Malik. Visual memory for robust path following. In *Advances in Neural Information Processing Systems*, pp. 765–774, 2018.

Xiangyun Meng, Nathan Ratliff, Yu Xiang, and Dieter Fox. Scaling local control to large-scale topological navigation, 2019.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17, 2017.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation, 2018.

Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

David Watkins-Valls, Jingxi Xu, Nicholas Waytowich, and Peter Allen. Learning your way without map or compass: Panoramic target driven visual navigation, 2019.

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. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017. doi: 10.1109/icra.2017.7989381. URL http://dx.doi.org/10.1109/ICRA.2017.7989381.