Pose Invariant Topological Memory for Visual Navigation

Asuto Taniguchi
asuto.taniguchi@jp.ricoh.com

Fumihiro Sasaki
fumihiro.fs.sasaki@jp.ricoh.com

Ryota Yamashina
ryohta.yamashina@jp.ricoh.com

Ricoh Company, LTD.

Abstract

Planning for visual navigation using topological memory, a memory graph consisting of nodes and edges, has been recently well-studied. The nodes correspond to past observations of a robot, and the edges represent the reachability predicted by a neural network (NN). Most prior methods, however, often fail to predict the reachability when the robot takes different poses, i.e. *the direction the robot faces, at close positions. This is because the methods observe first-person view images, which significantly changes when the robot changes its pose, and thus it is fundamentally difficult to correctly predict the reachability from them. In this paper, we propose pose invariant topological memory (POINT) to address the problem. POINT observes omnidirectional images and predicts the reachability by using a spherical convolutional NN, which has a rotation invariance property and enables planning regardless of the robot's pose. Additionally, we train the NN by contrastive learning with data augmentation to enable POINT to plan with robustness to changes in environmental conditions, such as light conditions and the presence of unseen objects. Our experimental results show that POINT outperforms conventional methods under both the same and different environmental conditions. In addition, the results with the KITTI-360 dataset show that POINT is more applicable to real-world environments than conventional methods.*

1. Introduction

Planning for visual navigation has been widely studied. Typically, a geometric map of an environment is constructed by visual simultaneous localization and mapping (vSLAM) [33, 23], and a path to a goal is planned by using a planning algorithm such as A* search [31]. However, in practical cases, vSLAM requires much effort to obtain a geometrically accurate map, such as combination with high-precision odometry or global navigation satellite system (GNSS) sensors, and map modification and parameter tuning by an expert. On the other hand, a number of works have proposed planning methods using *topological memory* [11, 32, 9, 24, 26, 36, 3], which is a memory graph where each node corresponds to a past observation of a robot and the edge between two nodes represents their reachability, *i.e.* whether the positions corresponding to the nodes are physically close or distant. In particular, recent works, such as semi-parametric topological memory (SPTM) [32] and hallucinative topological memory (HTM) [26], leverage a neural network (NN) to predict the reachability. We call such an NN an *edge predictor*. These methods do not aim to build a geometrically accurate map, and thus do not require any combination with other high-precision sensors or map modification and parameter tuning by an expert. Since it is to be desired that visual navigation can be easily performed without such requirements, we focus on topological memory based methods. The overview of planning based on topological memory is shown in Figure 1.

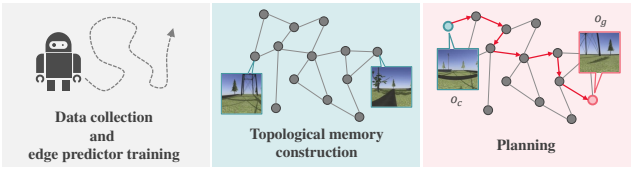
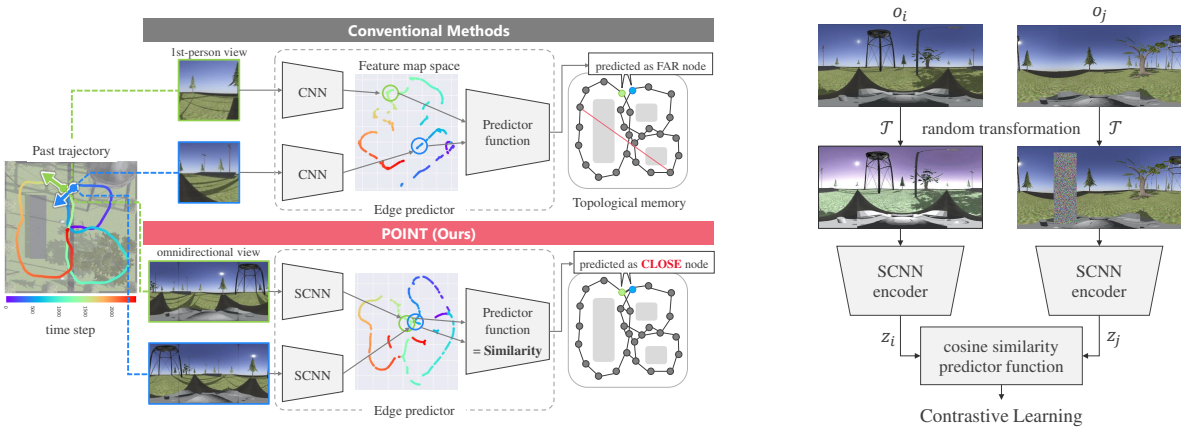


Figure 1. Overview of topological memory based planning. A robot moves around an environment to collect data and train an edge predictor with the data. Then, a topological memory is constructed on the basis of the edge predictor. The topological memory has nodes sampled from collected data and edges connected between nodes if the nodes are predicted to be close (thus reachable) by the edge predictor. Finally, given current observations o_c and goal observations o_g, planning is performed on the topological memory using an algorithm such as the Dijkstra algorithm [27].

Previous topological memory based planning methods, however, have a difficulty; the edge predictor often fails prediction given observations with different poses. To explain the reason, we illustrate the procedure of the topologi-



(a) Topological memory construction of conventional methods and POINT

(b) Edge predictor training of POINT

Figure 2. (a) Illustration of topological memory construction of the conventional methods and POINT (our method). The left image represents the bird's eye view image of the environment and the past trajectory of the robot where the color represents a time step. The green and blue arrows represent the positions and directions of the robot. The upper and bottom rows show the conventional methods and POINT, respectively. (b) Overview of edge predictor training of POINT . We use contrastive learning with data augmentation.

cal memory construction of the conventional methods in the top row of Figure 2(a). The left image represents the past trajectory of the robot. While the robot positions, shown as positions of the green and blue arrows in the left image, are close and thus supposed to be predicted as reachable, the robot poses, shown as the direction of the arrows, are different. The conventional methods use first-person view images as observations and convolutional NNs (CNNs) as the edge predictor. Since first-person view images observed at the green and blue arrows are totally dissimilar because their poses are different, the feature maps extracted by the CNNs are also dissimilar (see "Feature map space" in Figure 2(a)). Therefore, it is fundamentally difficult to correctly predict that the nodes corresponding to the blue and green arrows are close. This mis-prediction, moreover, leads to a failure of localization and planning when the robot takes the pose it has never taken. To deal with the problems by the mis-prediction of the reachability due to the dissimilarity of the feature maps, the conventional methods exploit a trainable function, which is referred to as a *predictor function* in this paper, that maps two feature maps to the reachability from one to another, such as ones implemented as fully connected (FC) layers at the top of the siamese network [15], to implement the edge predictor functions. However, such a predictor function increases the risk of incorrectly predicting dissimilar images observed at distant points as ones observed at close points, as depicted by the pink line in the topological memory in Figure 2(a).

To address the difficulty, we propose *pose invariant topological memory (POINT)*. The bottom row of Figure 2(a) shows an overview of the topological memory construction of POINT. POINT observes omnidirectional view images and uses an edge predictor consisting of spherical CNNs (SCNNs) [5] and a cosine similarity predictor function. Thanks to a rotation invariance property of SCNNs, the feature maps corresponding to observations at close positions are similar even if the robot takes different poses, as shown by the green and blue arrows in Figure 2(a). Hence, the corresponding nodes can be easily predicted to be close by simply determining the similarity between feature maps. Besides, the predictor function outputting the similarity of feature maps rarely predicts dissimilar images observed at distant positions to be close. In addition, considering more practical situations, we attempt to achieve robust planning to environmental condition changes, *e.g.* shadow direction, light conditions, and the presence of unseen objects such as pedestrians. To obtain the robustness, we train the edge predictor by contrastive learning with data augmentation [4] that learns to obtain similar features from randomly transformed images, as shown in Figure 2(b). In our experiments, we show that POINT outperforms the conventional methods under both the same and different environmental conditions. In addition, the results with the KITTI-360 dataset [37] indicate that POINT is more applicable to real-world environments than conventional methods.

2. Related Work

2.1. Planning using Topological Memory

Our method is built upon existing topological memory based planning methods [11, 32, 9, 26, 24, 36, 20, 3]. Many works suppose that actual movement along a planned path is performed by using a local movement policy trained in addition to the edge predictor. We focus on only topological

memory construction and planning but not the local policy training to purely evaluate planning performance.

The major differences between our method and most previous works, such as SPTM and HTM, are that we use omnidirectional images instead of first-person view images for the observation and SCNNs [5] in the edge predictor. This enables the robot to find the closest nodes on the topological memory regardless of its pose. While [36] also uses omnidirectional images as inputs of an edge predictor which does not have the rotation invariance property theoretically, our edge predictor theoretically guarantees the rotation invariance for the omnidirectional images. Another difference is that our method trains the edge predictor to be robust to changes in the environmental conditions by contrastive learning with data augmentation.

2.2. Planning using Dynamical Model

A number of works construct a dynamical model for planning [7, 14, 16, 13, 1, 10, 28, 18]. Most of them plan action series by using model predictive control. Their advantage compared with the topological memory based planning methods including our methods is that they do not require to additionally learn the local movement policy. However, dynamical model errors accumulate over time, making it difficult to deal with long-horizon tasks. On the other hand, the topological memory based planning methods require learning the local policy while being able to perform long-horizon planning without such error accumulations.

2.3. Spherical Convolutional Neural Networks

There have been a number of studies proposing CNNs for omnidirectional data [5, 8, 6, 29, 21, 36]. [5] and [8] proposed theoretically rotation-equivariant CNNs on the unit sphere. SphereNet proposed in [6] deals with distortion on omnidirectional images but is not invariant to rotation. DeepSphere [29] leverages graph CNNs to achieve rotation equivariance and computational efficiency. [21] proposed a spherical CNN on an unstructured grid that improves parameter efficiency. In this paper, we use the work of [5] because it theoretically guarantees rotation invariance and can be easily by using its official code[1].

2.4. Contrastive Learning

Contrastive learning [15, 35, 2, 19, 4, 17, 12] is categorized as unsupervised / self-supervised learning and aims to learn good representations for downstream tasks such as image recognition [19, 4, 17, 12] and reinforcement learning [35, 25]. In this paper, we leverage contrastive learning for visual navigation. Many contrastive learning methods learn to minimize a loss function called InfoNCE [35] so that a positive pair is more similar to negative pairs in some measures. [4, 34] has demonstrated that data augmentation helps to acquire robust good representations. We expect our method to be robust to environmental condition changes by using contrastive learning with data augmentation.

[1] https://github.com/jonas-koehler/s2cnn

3. POINT: Pose Invariant Topological Memory

In this section, we propose pose invariant topological memory (POINT) for visual navigation to address the problem of the conventional methods. We first explain the procedure of planning using topological memory including SPTM, HTM, and POINT briefly. Figure 1 illustrates the procedure. The purpose of planning using topological memory is to provide waypoint observations to reach a goal from the current state, where the robot observes o_g and o_c, respectively. The topological memory $\mathcal{M}$ has N nodes sampled from past observations $\mathcal{D}_{\text{train}} = \{o_t | 0 \leq t \leq T-1\}$, where o_t denotes the observation at time step t and T denotes the number of the past observations. The edge predictor predicts the reachability between two nodes o_i and $o_j \in \mathcal{D}_{\text{train}}$. When planning from o_c to o_g, they are added to $\mathcal{M}$ as nodes and the shortest path problem is solved in an algorithm such as the Dijkstra algorithm [27].

Importantly, the prediction accuracy of the edge predictor directly affects the planning performance. POINT, as well as SPTM and HTM, trains the edge predictor to distinguish whether past observations o_i and o_j are separated by at most Δt_p time steps ($|i-j| < \Delta t_p$) or by at least Δt_n time steps ($|i-j| > \Delta t_n$), where Δt_p and Δt_n are hyperparameters. The edge predictor is implemented as the siamese network [15] consisting of two encoders $E : \mathcal{O} \to \mathbb{R}^d$ with shared parameters and a predictor function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, where $\mathcal{O}$ denotes the observation space and d denotes the dimension of feature vectors extracted by E. The differences between POINT and the conventional methods lie in the composition and training of the edge predictor. We summarize the differences among SPTM, HTM, and POINT in Table 1. We describe the composition and training of our edge predictor below.

3.1. Edge Predictor using Spherical CNNs and Cosine Similarity Predictor Function

To achieve accurate edge prediction regardless of the robot's pose, we propose two solutions. First, we observe omnidirectional images and use SCNNs [5] for E to eliminate the pose dependency, instead of first-person view images and CNNs. The SCNNs are known to be able to extract rotation-invariant feature vectors [5]. It indicates that similar feature vectors $z \in \mathbb{R}^d$ can be obtained from observations at different robot's poses, and thus we can easily predict the accurate reachability even when the robot faces different directions. Second, we employ $f(z_i, z_j) = \exp(\text{sim}(z_i, z_j)/\tau)$ as the predictor function, where $\text{sim}(z_i, z_j) = z_i^T z_j / \|z_i\| \|z_j\|$ is the cosine similar-

Method	Encoder E	Predictor function f	Training	Data Aug.
SPTM	CNN	FC layers	Classification	No
HTM	CNN	$\exp\left(z_i^T W z_j\right)$	Contrastive Learning	No
POINT (Ours)	SCNN	$\exp(z_i^T z_j / \tau \|z_i\| \|z_j\|)$	Contrastive Learning	Yes

Table 1. Difference among SPTM, HTM, and POINT. FC layers for SPTM and $W \in \mathbb{R}^{d \times d}$ for HTM are trainable and not positive definite.

ity between $z_i = E(o_i)$ and $z_j = E(o_j)$, $\|\cdot\|$ denotes L_2-norm, and $\tau > 0$ is a temperature parameter. Since z do not depend on the robot's pose because of the rotation invariance property of the SCNNs, we can predict the reachability by simply calculating the similarity of z. POINT connects an edge between nodes o_i and o_j with edge cost $c_{ij} = 1/f(z_i, z_j)$, in the same way as HTM. An expected advantage of our predictor function model relative to those of SPTM and HTM is that it rarely connects an edge between distant nodes, such as the pink line in the topological memory in Figure 2(a). This is because z corresponding to observations at distant positions are almost always dissimilar. We empirically found that predictor functions with a positive definite property work reasonably well, and are not limited to the cosine similarity. For more detail, see 4.1.2.

These two solutions are simple but expected to be effective for accurate topological memory construction and planning. The overview of the difference between the conventional methods and POINT is illustrated in Figure 2(a).

3.2. Training for Robust Edge Prediction to Environmental Condition Changes

Considering more realistic situations, we improve POINT to be robust to changes in environmental conditions, *e.g.* light conditions, shadow direction, and the presence of unseen objects like pedestrians. Figure 2(b) shows the overview of the training. We train the edge predictor using contrastive learning with data augmentation. During training, we apply a random transformation $\mathcal{T}$ to the observed images such as color jitter and random erasing. That is, we train the edge predictor to minimize the following loss function called InfoNCE:

$$\mathcal{L} = -\mathbb{E}_{o_t \sim \mathcal{D}_{\text{train}}} \left[\log \frac{f(\tilde{z}_t, \tilde{z}_k)}{\sum_{o_i \in \mathcal{D}_{\text{Ctr}}} f(\tilde{z}_t, \tilde{z}_i)} \right], \quad (1)$$

where $\tilde{z}_i = E(\mathcal{T}(o_i))$ denotes the feature vector corresponding to the randomly transformed observation $\mathcal{T}(o_i)$. $\mathcal{D}_{\text{Ctr}}$ contains one observation o_k where (o_t, o_k) satisfies $|t-k| < \Delta t_p$, and $K-1$ observations randomly sampled from $\mathcal{D}_{\text{train}}$. Δt_p is a hyperparameter.

This training aims to extract similar features from images whose observed times are close, even when the images are differently transformed. It is expected that POINT would be able to predict edge connections with robustness to environmental condition changes.

4. Experiments

We conducted experiments using synthetic and real-world datasets. In experiments with the synthetic dataset, we address the following questions:

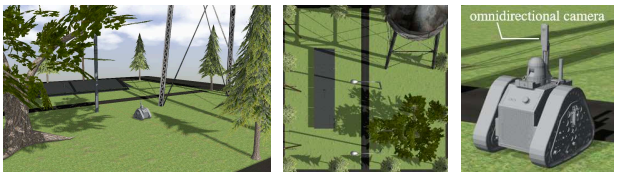
- Does POINT outperform the conventional topological memory based planning methods?
- Does the SCNN encoder allow successful planning regardless of the robot's pose?
- Does the contrastive learning with data augmentation improve the robustness to the environmental condition changes?
- What property should the predictor function have for better topological memory construction?

On the other hand, in experiments with the real-world dataset, we evaluate the applicability of POINT and the conventional methods in real-world environments. The experimental protocol and results for the synthetic and real-world dataset are described in Sections 4.1 and 4.2, respectively.

4.1. Evaluation with Synthetic Dataset

4.1.1 Setup

Dataset To create a synthetic dataset, We implemented the simulation environment using the robot operation system (ROS) [30] and the Gazebo simulator [22] as shown in Figure 3. We set up four different environmental conditions shown in Figure 4; *base, sunset, obstacles*, and *different shadow*. We manually operated the robot under the base environmental condition (Figure 4(a)) to collect $\mathcal{D}_{\text{train}}$ four times. Each path of $\mathcal{D}_{\text{train}}$ is shown in Figure 6. $\mathcal{D}_{\text{train}}$ is composed of omnidirectional images the robot observed along the paths. $\mathcal{D}_{\text{test}}$, from which o_c and o_g were sampled for planning, was also collected four times under each environmental condition (base, sunset, obstacles, and different shadow) in the same way as one for collecting $\mathcal{D}_{\text{train}}$. Although the routes for collecting $\mathcal{D}_{\text{test}}$ were similar to ones for collecting $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{test}}$ has the observations along both the same and opposite route directions – for example, if both $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ have observations at the same position under the same environmental condition, $\mathcal{D}_{\text{test}}$ has observations with different poses from the ones in $\mathcal{D}_{\text{train}}$.



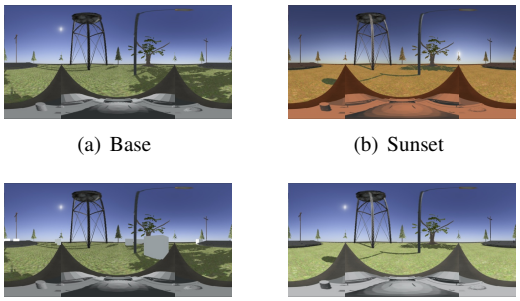
(a) Simulation environment (b) Bird's eye view (c) Robot

Figure 3. (a) Simulation environment and (b) its bird's eye view. The environment has an area of 20 × 20 m. (c) Robot that has an omnidirectional camera on the top and can move about 3 m in 100 time steps if it continued to move straight ahead. More details of the simulatior can be found in the supplementary material.

Training and evaluation procedure Given $\mathcal{D}_{\text{train}}$, we trained edge predictors by several variants of the topological memory based planning methods including SPTM, HTM, SoRB [9][2], and POINT, each of which is defined as a different combination of input image formats, the image transform functions, the encoders, and the predictor functions. The input image format is chosen from either the first person view or the omnidirectional image. If the first person view images were chosen, the input images were acquired by cropping the center of the omnidirectional images in $\mathcal{D}_{\text{train}}$ so that its field of view was 90 degrees as shown in Figure 5(a), otherwise by masking the lower third of the omnidirectional images in D as shown in Figure 5(b). The mask is to remove images of the robot body that disrupt the pose invariance of the omnidirectional images. The input images were downsized to 128 × 128 before being fed to the image transform function. The image transform function was chosen from the identity function (thus the data augmentation was not performed) or the random transform function $\mathcal{T}$. The image transform function was applied to the input images before being fed into the encoder. the encoder is chosen from either a CNN encoder or an SCNN encoder. The edge predictor functions are chosen from the one used in SPTM, the one used in HTM, the one used in SoRB, or the one described in Section 3.1. The training objectives of those variants are determined by the choice of edge predictor as shown in Table 1. The hyperparameters Δt_p and Δt_n were set to 100 and 200, respectively, since these empirically worked well for all variants. The implementation of the edge predictors and other hyperparameters are described in the supplementary material.

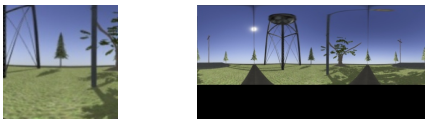
The topological memories $\mathcal{M}$ were constructed with $N = 300$ nodes. We used the k-nearest neighbor (kNN) edge clean-up method proposed in [24] because it empirically worked well to reduce edge mis-connection between actual distant nodes. It permits only the k edges with the largest $f(z_i, z_j)$ connected for each node. We set $k = 20$.

[2]While the original SoRB trains the edge predictor by exploring the environment, we trained it by using data collected offline.



(a) Base (b) Sunset (c) Obstacles (d) Different shadow

Figure 4. Environmental conditions. $\mathcal{D}_{\text{train}}$ is collected under the (a) base condition, and planning is performed under the (a) base, (b) sunset, (c) obstacles, and (d) different shadow conditions. (b) has different colors and shadow directions from (a). (c) has a number of white boxes in random positions. (d) has different shadow directions from (a) but has the same color.



(a) CNN (b) SCNN

Figure 5. Input images for encoders. (a) is the input for the CNN encoder, *i.e.* first-person view images, while (b) is the input for the SCNN one. Both were downsized to 128 × 128.

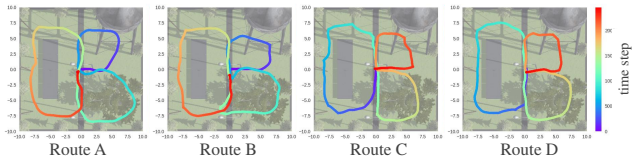


Figure 6. Route paths for datasets. The color represents the time step. The total time steps T for routes A, B, C, and D were 2442, 2404, 2424, and 2456, respectively.

We evaluate the topological memory based planning methods as follows. Given o_c and o_g randomly sampled from $\mathcal{D}_{\text{test}}$, we performed planning from o_c to o_g by the Dijkstra algorithm based on the constructed topological memories. A planning result is evaluated as a success if (i) the planned path connecting positions where o_c and o_g were observed exists in $\mathcal{M}$ and (ii) it does not pass through an edge that connects two nodes that are more than 6 m[3] apart. We conducted 100 trials for each $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$, that is, 100 (trials) × 4 ($\mathcal{D}_{\text{train}}$) × 4 ($\mathcal{D}_{\text{test}}$) = 1600 trials for each environmental condition.

[3]It was set to the distance that the robot was able to move during Δt_n time steps if it continued to move straight ahead.

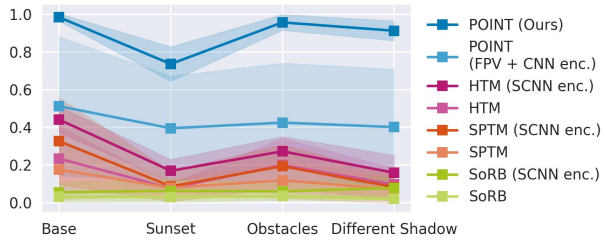


Figure 7. Planning success rates of SPTM, HTM, SoRB [9], and POINT. SPTM (SCNN enc.) denotes a variant of SPTM that exploit SCNN encoder with omnidirectional images instead of the ones SPTM exploits, and the same goes for HTM (SCNN enc.) and SoRB (SCNN enc.). POINT (FPV + CNN enc.) denotes a variant of POINT that exploits CNN encoder with first-person view images instead of the ones POINT exploits.

4.1.2 Results

Does POINT outperform the conventional topological memory based planning methods? The experimental results are summarized in Figure 7. While the success rates of the conventional methods were low even under the base condition, in which $\mathcal{D}_{\text{train}}$ is collected, POINT showed much higher success rates. Besides, under the other conditions, POINT maintained high success rates while the performance of the conventional methods deteriorated. It suggests that POINT significantly outperformed the conventional methods. We found that POINT outperformed POINT (FPV + CNN enc.), which uses first-person view images and the CNN encoder, in all conditions, while POINT (FPV + CNN enc.) outperformed the conventional methods especially under the conditions except for the base condition. It indicates that both the SCNN encoder with omnidirectional images and contrastive learning with data augmentation described in Section 3.2 are effective for robust and better planning. We further investigate how the SCNN and constrastive learning with data augmentation work. The investigation details are described below.

Does the SCNN encoder allow successful planning regardless of the robot's pose? We compared POINT with a variant of POINT denoted by POINT (Omni. + CNN enc.), which observes omnidirectional images but uses a CNN encoder. We separately evaluated the success rates under two cases: the route direction of $\mathcal{D}_{\text{test}}$ was either the *same* with or *opposite* to that of $\mathcal{D}_{\text{train}}$. Figure 8 shows the success rates in each case. Remarkably, the success rates of POINT (Omni. + CNN enc.) significantly decreased when the route direction of $\mathcal{D}_{\text{test}}$ was opposite to the ones of $\mathcal{D}_{\text{train}}$, while POINT maintains its performance. It suggests that the SCNN enabled proper edge prediction even between observations in completely different directions, resulting in successful planning regardless of the robot's pose.

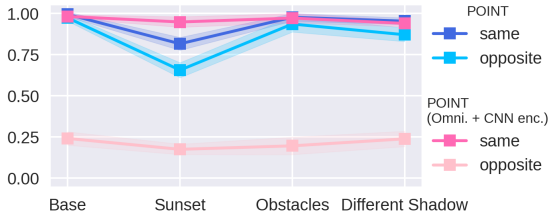


Figure 8. Planning success rates of POINT and POINT (Omni. + CNN enc.), which observes omnidirectional images but uses a CNN encoder.

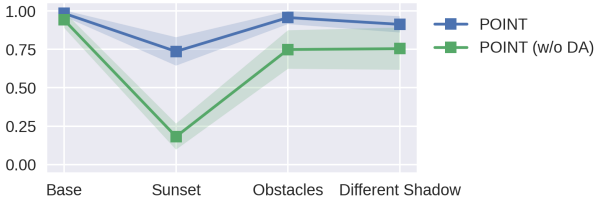


Figure 9. Planning success rates of POINT and POINT (w/o DA), which trains the edge predictor by contrastive learning without data augmentation.

Does the contrastive learning with data augmentation improve the robustness to the environmental condition changes? We compared POINT with POINT (w/o DA), which trains the edge predictor by contrastive learning without data augmentation. Figure 9 shows the success rates of POINT and POINT (w/o DA). We found that the performance of POINT (w/o DA) deteriorated when environmental conditions changed. It suggests that data augmentation helps robust planning to changes in environmental conditions. We expect the robustness to play an important role for practical use in ever-changing environments.

What property should the predictor function have for better topological memory construction? We further investigate if the choice of predictor function affects the planning performance. We compared various predictor function models as shown in Table 2, where $\tau = 0.01$. The cosine similarity and log-bilinear models correspond to the predictor functions of POINT and HTM, respectively. The inner product model and linear inner product model can be interpreted as a log-bilinear model with a positive definite matrix $W = I$ and $W = U^T U$, respectively. The Gaussian kernel model is also a positive definite function.

Figure 10 shows the success rates of them. We found that the cosine similarity achieved the highest success rates under the base, sunset, and different shadow conditions, and was competitive with the inner product model under the obstacles condition. On the other hand, the log-bilinear model used in HTM obtained the worst success rates. For more analysis, we show the topological memories built when us-

Model name	Definition of $\log f$
log-bilinear	$z_i^T W z_j$
cosine similarity	$z_i^T z_j / (\tau \|z_i\| \|z_j\|)$
inner product	$z_i^T z_j$
linear inner product	$(U z_i)^T (U z_j)$
Gaussian kernel	$\frac{1}{\tau} \exp(-\|z_i - z_j\|^2)$

Table 2. Predictor function models. Both W and $U \in \mathbb{R}^{d \times d}$ are trainable parameters.

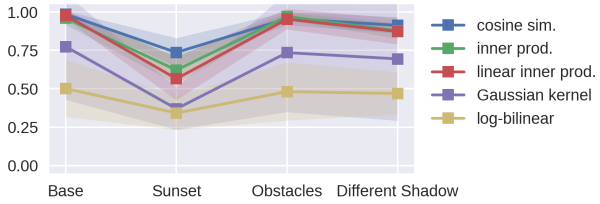


Figure 10. Planning success rates of POINT with various predictor functions.

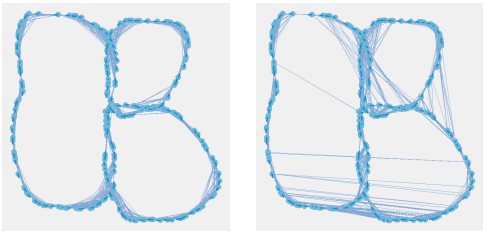
ing the cosine similarity and log-bilinear models in Figure 11. It can be seen that the topological memory of the cosine similarity does not connect edges between distant nodes, whereas that of the log-bilinear often does. These improper edge connections of the log-bilinear model caused the planning to pass an edge between distant nodes and resulted in the worst score. Not limited to the cosine similarity model, the inner product, linear inner product, and Gaussian kernel models, which are all positive definite, achieved higher success rates than the log-bilinear model under all environmental conditions. It suggests that the positive definite property helps to achieve accurate topological memory construction and planning. We further investigate the difference among the predictor functions with the positive definite property. The investigation details can be found in the supplementary material. As a result, we conclude that the predictor function is desirable to take the normalized inputs as the cosine similarity function in addition to having the positive definite property.

4.2. Evaluation with Real-World Dataset

4.2.1 Setup

Dataset We used the KITTI-360 dataset [37], which contains real-world omnidirectional images captured on streets. We divided the dataset into $\mathcal{D}_{\text{train}} = \{o_{3k} | k \in \mathbb{N}, k \leq (T_{\text{all}} - 1)/3\}$ and $\mathcal{D}_{\text{test}} = \mathcal{D}_{\text{all}} \setminus \mathcal{D}_{\text{train}}$ for each data sequence $\mathcal{D}_{\text{all}} = \{o_t | 0 \leq t \leq T_{\text{all}} - 1\}$. The trajectory and an example image of each sequence are shown in Figure 12.

Training and evaluation procedure The edge predictor for each method is trained with $\mathcal{D}_{\text{train}}$. The first-person view input images were acquired by cropping the center of the omnidirectional images so that its field of view was 90 degrees[4] in the same way as the simulation experiments. The input image resolution was 256×256 and Δt_p and Δt_n were set to 10 and 20, respectively. The number of nodes in a topological memory was set to $N = 1200$ and k for kNN edge clean-up was set to 10. To obtain a robust topological memory, we used n-step mean of the predictor function output $\sum_{i=0}^{n-1} (f(z_t, z_{t'-i}) + f(z_{t-i}, z_{t'}))/2n$ instead of $f(z_t, z_{t'})$ for edge prediction between nodes o_t and $o_{t'}$, where $z_t = E(o_t), z_{t'} = E(o_{t'})$, and $n = 3$. Planning was performed 100 times for each $\mathcal{D}_{\text{all}}$. The implementation of the edge predictor and the other hyperparameters are detailed in the supplementary material.



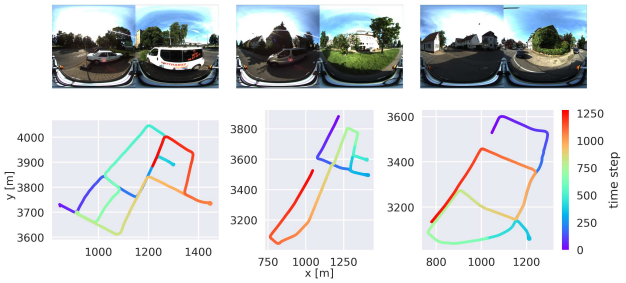
(a) cosine similarity (b) log-bilinear

Figure 11. Visualization of topological memories. The circles represent nodes and their locations correspond to the xy-coordinates of the position when the corresponding observations are acquired. The lines represent edges.

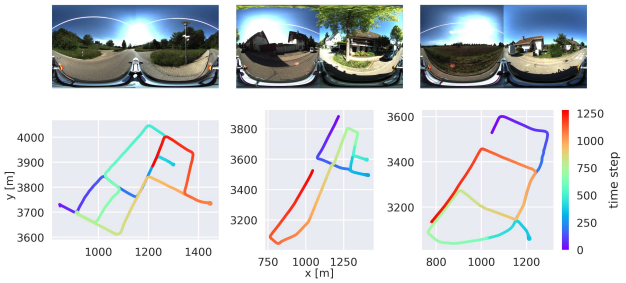
4.2.2 Results

Table 3 shows the mean success rate of POINT, the conventional methods, and variants of those methods. We can see POINT achieved the highest success rate than other methods. It indicates that POINT is more applicable to real-world environments than conventional methods. POINT (Omni. + CNN enc.) obtained better performance than SPTM (Omni.) and HTM (Omni.). It suggests that contrastive learning with data augmentation and the cosine similarity predictor function significantly improve planning performance. POINT (Omni. + CNN enc.) also obtained better performance than POINT (Omni. + CNN enc.) while POINT outperformed POINT (Omni. + CNN enc.). It indicates that exploiting the omnidirectional images rather than the first-person view images as the input of edge predictors is better suited for building topological maps in real-world environments and, thanks to the rotation invariance property, the SCNN encoder can utilize the omnidirectional images more effectively than the CNN encoder. To see the effect of SCNN qualitatively, we visualize the successful

[4] We did not use perspective images provided by [37] for first-person view images because we needed to match frame rate with omnidirectional image observations.



(a) 2013_05_28_drive_0000_sync



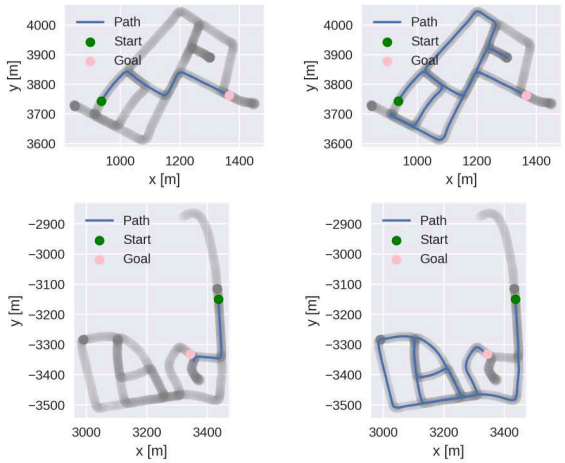
(b) 2013_05_28_drive_0004_sync

Figure 12. Trajectories and example images of the sequences we used. These sequences were obtained by dividing either 2013_05_28_drive_0000_sync or 2013_05_28_drive_0004_sync sequence into three. Each $\mathcal{D}_{\mathrm{train}}$ and $\mathcal{D}_{\mathrm{test}}$ have 1280 and 2560 images, respectively, for 2013_05_28_drive_0000_sync, whereas 1287 and 2574, respectively, for 2013_05_28_drive_0004_sync.

Method	Mean success rate
SPTM	0.020 ± 0.011
SPTM (Omni.)	0.040 ± 0.025
SPTM (SCNN enc.)	0.033 ± 0.016
HTM	0.045 ± 0.037
HTM (Omni.)	0.083 ± 0.028
HTM (SCNN enc.)	0.103 ± 0.034
POINT (FPV + CNN enc.)	0.435 ± 0.471
POINT (Omni. + CNN enc.)	0.908 ± 0.152
POINT	$\mathbf{0.932 \pm 0.163}$

Table 3. Planning success rates of SPTM, HTM, and POINT for the KITTI-360 dataset. SPTM (Omni.) and HTM (Omni.) are variants of SPTM and HTM respectively, which exploit CNN encoder with the omnidirectional images instead of the first-person view images.

planned path of POINT and POINT (Omni. + CNN enc.) in Figure 13. It can be found that POINT took the shortest paths, whereas POINT (Omni. + CNN enc.) took detouring paths. It suggests that, whereas CNNs often plan a redundant path even with omnidirectional image observations, the rotation invariance property of SCNNs helps to plan not only successful but also shorter paths.



(a) POINT (b) POINT (Omni. + CNN enc.)

Figure 13. Planned paths of (a) POINT and (b) POINT (Omni. + CNN enc.). Gray circles represent the nodes in each topological memory. The blue line is the planned path from the green circle to the pink circle.

5. Conclusion

In this paper, we have proposed pose invariant topological memory (POINT), a topological memory based planning method for visual navigation. Comparing with the conventional methods like SPTM and HTM, POINT can construct an accurate topological memory and perform planning regardless of the robot's pose, that is, the direction the robot faces. This advantage is obtained by the edge predictor consisting of the SCNNs, which theoretically guarantees the rotation invariance, and the cosine similarity predictor function, which rarely predicts dissimilar images observed at distant positions as close. Moreover, considering more practical situations, we train the edge predictor to be robust to changes in environmental conditions by using contrastive learning with data augmentation. We conducted experiments with synthetic images as well as real-world images. The results have demonstrated that (i) POINT outperformed the conventional methods under both the same and different environmental conditions, and (ii) POINT is more applicable to real-world environments than conventional methods.

We have focused only on the planning in this paper. In future work, we will examine the autonomous navigation performance based on the planning by conventional visual navigation methods such as vSLAM as well as the topological memory based methods including POINT. We believe that the comparison between vSLAM based methods and the topological memory based methods while taking into account the cost for building the map would provide insight into the real-world visual navigation problem.

References

[1] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable MPC for end-to-end planning and control. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8289–8300. Curran Associates, Inc., 2018. 3

[2] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*. 3

[3] Kevin Chen, Juan Pablo de Vicente, Gabriel Sepulveda, Fei Xia, Alvaro Soto, Marynel Vzquez, and Silvio Savarese. A behavioral approach to visual navigation with graph localization networks. In *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019. 1, 2

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. Feb. 2020. 2, 3

[5] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations*, 2018. 2, 3

[6] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 518–533, 2018. 3

[7] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-Based deep reinforcement learning for Vision-Based robotic control. Dec. 2018. 3

[8] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68. openaccess.thecvf.com, 2018. 3

[9] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems 32*, pages 15246–15257. Curran Associates, Inc., 2019. 1, 2, 5, 6

[10] C Finn and S Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. ieeexplore.ieee.org, May 2017. 3

[11] F Fraundorfer, C Engels, and D Nister. Topological mapping, localization and navigation using image collections. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3872–3877, Oct. 2007. 1, 2

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. June 2020. 3

[13] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2450–2462. Curran Associates, Inc., 2018. 3

[14] David Ha and Jürgen Schmidhuber. World models. Mar. 2018. 3

[15] R Hadsell, S Chopra, and Y LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. ieeexplore.ieee.org, June 2006. 2, 3

[16] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565, Long Beach, California, USA, 2019. PMLR. 3

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738. openaccess.thecvf.com, 2020. 3

[18] N Hirose, F Xia, R Martín-Martín, A Sadeghian, and S Savarese. Deep visual MPC-Policy learning for navigation. *IEEE Robotics and Automation Letters*, 4(4):3184–3191, Oct. 2019. 3

[19] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. 3

[20] Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. In *Advances in Neural Information Processing Systems 32*, pages 1942–1952. Curran Associates, Inc., 2019. 2

[21] Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Niessner. Spherical CNNs on unstructured grids. In *International Conference on Learning Representations*, 2019. 3

[22] N Koenig and A Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3. ieeexplore.ieee.org, Sept. 2004. 4

[23] Mathieu Labbé and François Michaud. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019. 1

[24] Michael Laskin, Scott Emmons, Ajay Jain, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. Mar. 2020. 1, 2, 5

[25] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In Hal Daumé Iii and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5639–5650, Virtual, 2020. PMLR. 3

[26] Kara Liu, Thanard Kurutach, Christine Tung, Pieter Abbeel, and Aviv Tamar. Hallucinative topological memory for Zero-Shot visual planning. In Hal Daumé Iii and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6259–6270, Virtual, 2020. PMLR. 1, 2

[27] Kairanbay Magzhan and Hajar Mat Jani. A review and evaluations of shortest path algorithms. *International journal of scientific & technology research*, 2(6):99–104, 2013. 1, 3

[28] Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-Supervised learning of Long-Horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2020. 3

[29] N Perraudin, M Defferrard, T Kacprzak, and R Sgier. DeepSphere: Efficient spherical convolutional neural network with HEALPix sampling for cosmological applications. *Astronomy and Computing*, 27:130–146, Apr. 2019. 3

[30] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. willowgarage.com, 2009. 4

[31] N Sariff and N Buniyamin. An overview of autonomous mobile robot path planning algorithms. In *2006 4th Student Conference on Research and Development*, pages 183–188. ieeexplore.ieee.org, June 2006. 1

[32] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018. 1, 2

[33] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16, June 2017. 1

[34] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6827–6839. Curran Associates, Inc., 2020. 3

[35] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. July 2018. 3

[36] Tsun-Hsuan Wang, Hung-Jui Huang, Juan-Ting Lin, Chan-Wei Hu, Kuo-Hao Zeng, and Min Sun. Omnidirectional CNN for visual place recognition and navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2018. 1, 2, 3

[37] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic instance annotation of street scenes by 3D to 2D label transfer. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. 2, 7