

Spatio-Temporal Graph Localization Networks for Image-based Navigation

Takahiro Niwa¹, Shun Taguchi¹, and Noriaki Hirose¹

Abstract—Localization in topological maps is essential for image-based navigation using an RGB camera. Localization using only one camera can be challenging in medium-to-large-sized environments because similar-looking images are often observed repeatedly, especially in indoor environments. To overcome this issue, we propose a learning-based localization method that simultaneously utilizes the spatial consistency from topological maps and the temporal consistency from time-series images captured by the robot. Our method combines a convolutional neural network (CNN) to embed image features and a recurrent-type graph neural network to perform accurate localization. When training our model, it is difficult to obtain the ground truth pose of the robot when capturing images in real-world environments. Hence, we propose a sim2real transfer approach with semi-supervised learning that leverages simulator images with the ground truth pose in addition to real images. We evaluated our method quantitatively and qualitatively and compared it with several state-of-the-art baselines. The proposed method outperformed the baselines in environments where the map contained similar images. Moreover, we evaluated an image-based navigation system incorporating our localization method and confirmed that navigation accuracy significantly improved in the simulator and real environments when compared with the other baseline methods.

I. INTRODUCTION

Autonomous mobile robots have been attracting attention because of their potential utility in daily applications such as transportation of objects, automatic cleaning, guidance, and patrols. As opposed to navigation systems using multiple LiDARs, some studies have tackled vision-based navigation using a monocular camera owing to their low cost, light weight, compact size, robustness, and high availability.

One of the existing techniques for visual navigation is visual simultaneous localization and mapping (SLAM), which simultaneously creates a map of the environment with a three-dimensional structure and estimates self-position [1]–[4]. Although visual SLAM can produce a detailed map of the environment, it requires accurate camera calibration. In addition, collision avoidance and robustness against environmental changes create challenges for visual navigation.

To address these issues, image-based navigation using topological maps has recently attracted significant attention [5]–[13]. A topological map is a graph-structured map created from the image sequences obtained by the robot. Each node in the map contains a monocular camera image. Adjacent nodes are connected on edges based on image similarity [5] or reachability estimation [11]. During navigation, the robot identifies its own position as a node number on the topological map. The robot then generates subgoal

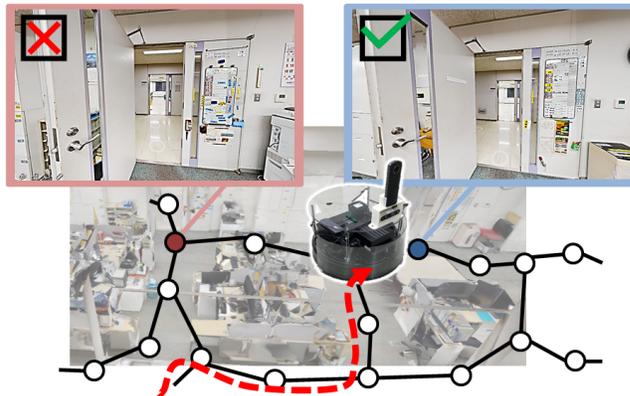


Fig. 1: Localization for image-based navigation of robots, utilizing the graph structure of topological maps and time-series information from images observed by the robots. Our method distinguishes between similar images via a recurrent-type graph neural network.

images between the identified and destination nodes [14]. The navigation system derives velocity references to control the robot using the subgoal images and the current image from the robot [12], [13]. Further details are provided in the evaluation section of this paper.

In this study, we focus on localization in a topological map. Most localization methods for image-based navigation are based on image retrieval using the current robot image as the query, and the node images as the references [5]–[8]. However, these methods pose two issues during navigation. Topological maps, especially in indoor environments, often contain similar images (as shown in Fig. 1) because indoor environments such as office rooms, corridors, meeting spaces, and airports are often composed of repetitions of one environment. In such cases, baseline localization methods often select the wrong node, which leads to navigation failures.

Another issue is that collecting massive numbers of images and the ground truth (GT) poses in a real environment is challenging. Hence, we cannot conduct supervised learning using the real images.

In this study, we propose a graph neural network-based localization method that can utilize spatial and temporal consistency using a topological map and time-series images from the robot (Fig. 1). The proposed method is based on two intuitive ideas. Adjacent node images in the map can help identify the correct nodes. Adjacent node images contain the same objects as in the current robot image and have similar appearances. This spatial consistency can be learned via a graph neural network using the proposed method. Another

¹Takahiro Niwa, Shun Taguchi, and Noriaki Hirose are with Toyota Central R&D Labs., INC, Japan niwa-takahiro@mosk.tytlabs.co.jp

idea is to learn temporal consistency using robot time-series images via the LSTM layer. The history of the robot’s motion can eliminate the possibility of selecting similar but incorrect nodes.

In addition, we propose a semi-supervised learning method that utilizes simulator images with the GT pose in addition to real images without the GT pose. The GT pose in the simulator images enables supervised learning to achieve accurate localization even in the real images [12], [15].

We evaluate the effectiveness of the proposed method by comparing it with several baseline methods. We also tested it on a robot navigation task using the Gibson simulator [16], [17] and a real environment. The evaluation results show that the proposed method provides accurate localization and achieves highly accurate navigation in both simulated and real-world environments. The main contributions of this study are as follows:

- We proposed a novel graph neural network-based localization method. To the best of our knowledge, our study is the first application that uses a recurrent-type graph neural network for localization in a topological map.
- We developed a semi-supervised learning method using real-world images without the GT pose and simulator images with the GT pose for sim2real transfer.
- We implemented an image-based navigation system that incorporated the proposed localization method into the evaluation.

II. RELATED WORKS

There is a long history of research on visual navigation for mobile robots. We begin with a comprehensive discussion of visual navigation. We then focus on visual localization, which is particularly relevant to our method.

A. Visual Navigation System

Visual navigation can be broadly divided into model- and learning-based approaches. For model-based visual navigation, researchers have proposed solutions based on visual servoing and visual SLAM.

Visual servoing [18]–[20] controls an agent to minimize the difference between the current and goal states. Because the difference is defined in the image space, performance suffers when the environment changes or large obstacles occlude large parts of the environment. Navigation methods based on visual SLAM [1]–[4] first use the camera images to construct a map that can be used by the robot to localize and compute actions to achieve a goal. The success of visual SLAM-based methods relies on acquiring an accurate metric model, and their performance decays when mapping failures occur.

To address these issues, multiple learning-based approaches have been recently proposed as image-based navigation. Image-based navigation using a topological map [5]–[10] involves localization and planning from a topological representation of the environment that represents the connectivity between regions. The latest advances in reinforcement learning [21]–[24] and imitation learning [25]–[27] have also pushed the state of the art in image-based navigation.

B. Visual Localization

Visual localization using maps can be roughly divided into camera re-localization and visual place recognition. Camera re-localization estimates the camera pose in Euclidean space in small known environments. Several approaches such as direct camera pose regression [28], [29], coarse-to-fine [30]–[32], and structure-based approaches [33]–[36] have been studied. By contrast, visual place recognition is a task that involves retrieving images from a very large image database. It is mainly based on hand-crafted or deep-learning-based image features [37], [38].

Visual localization for image-based navigation retrieves the closest node on the graph, but does not estimate the camera pose in Euclidean space. Therefore, it can be implemented for image retrieval on a graph, similar to visual place recognition. In one of the earliest studies [5], localization was performed by estimating similarity using the Siamese network [39]. This can be replaced by other image retrieval methods, such as NetVLAD [37].

Image retrieval methods are estimated from only a single image; however, it is reasonable to use time-series observations for visual navigation. Our method employs a graph convolutional LSTM to improve the localization accuracy of a graph, which can handle time-series observations and spatial information from topological maps.

III. PROBLEM STATEMENT

We consider the problem of localizing a robot that moves along the edges of a topological map in an indoor environment. A topological map is a graph-structured map created from a sequence of images obtained by the robot during its past trajectories. The images are held as nodes and spatially adjacent nodes are connected by edges in the topological map. Note that the self-position localized in this research is not the position in Euclidean space, but the index of the node closest to the robot.

In the following section, we define the topological map as a directed graph $G = (V, E)$, where $V = \{v_i\}_{i=1:n}$ is the set of n nodes, and v_i is the obtained image at node i . Also, $E = \{(r_k, s_k)\}_{k=1:m}$ is a tuple of m edges, where edge k is connected from source node s_k to target node r_k . The robot position is expressed as a sequence of node indices $Y = \{y_t\}_{t=1:T}$ corresponding to the observed images $O = \{o_t\}_{t=1:T}$. Each robot position y_t denotes the index of the node closest to the robot. The topological localization problem is defined as follows: *Given the topological map $G(V, E)$, find a current node y_t at every time step t using past observed images O .*

IV. PROPOSED METHOD

As mentioned in the introduction, baseline localization methods have difficulty accurately localizing the self-position when multiple similar node images are contained in the topological map. To solve this problem, we use the time-series images observed by the robot and spatial information from the topological map to consider spatial and temporal consistency for accurate localization. In this section, we

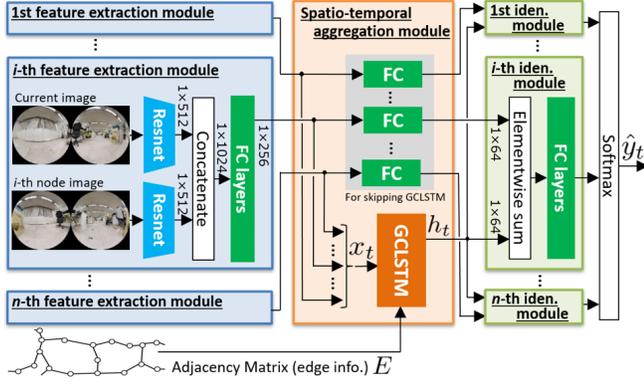


Fig. 2: **Network structure of our proposed method with GCLSTM.** The Spatio-temporal aggregation module aggregates the features from the Feature extraction module to consider spatial and temporal consistency. Note that Feature extraction and Identification modules are performed for each node image.

describe the proposed neural network configuration that utilizes spatio-temporal information sequentially.

A. Spatio-Temporal Consistent Localization

We propose a novel graph neural network-based localization method that utilizes the spatial information from topological maps and time-series images observed by the robot. An overview of the proposed method is presented in Fig. 2. Our model consists of three modules: a “Feature extraction module”, a “Spatio-temporal aggregation module”, and an “Identification module”. The Feature extraction module extracts a feature from the current node image and each remaining node image and calculates the relationship between them. The Spatio-temporal aggregation module aggregates the features of neighboring and past nodes. The Identification module calculates the localization probability for each node. The details of each module are provided in the following.

Feature extraction module This module consists of the ResNet encoder and fully connected (FC) layers. The current image o_t and node images V are encoded into feature vectors by the ResNet-18 encoder. The ResNet encoder is used to extract the features of the images and reduce the dimensions. The FC layers are employed to extract the similarity between the current image and each node image. The features extracted from the FC layers are fed into the Spatio-temporal aggregation module.

In inference tasks, ResNet-18 for the node images can be calculated offline to reduce the online computational load.

Spatio-temporal aggregation module This module consists of graph convolutional LSTM (GCLSTM) [40] and a skip path with the FC layers. We employ GCLSTM to simultaneously handle temporal information from time-series observations and spatial information from the topological maps. GCLSTM is an extended model of a graph convolutional neural network (GCN), which is a general and effective framework for learning representations of graph-structured

data. We introduced this graph-convolutional strategy into our topological localization.

The graph convolution function in GCN is generally computed as the weighted sum of the features corresponding to each node and its neighbor nodes. By introducing this in GCLSTM, we can aggregate the features of the neighboring nodes of interest. Moreover, GCLSTM can introduce time-series information to the graph convolution by using an LSTM-based structure. Therefore, by using GCLSTM, we can introduce information, such as the node which closes to the node that has achieved high probability in previous observations is more confident, into the network.

While the GCLSTM layer outputs the overall features by convolving the features of neighboring nodes, it dilutes features of a self-node. To address this issue, we employ one FC layer to skip the GCLSTM layer to directly propagate the features of self-nodes to the later layers. The FC layer reduces the dimensions of the features, which is equal to those of the GCLSTM outputs. Note that the FC layers for each node share the same weights and biases. The details of GCLSTM architecture are shown later.

Identification module This module consists of FC layers and a softmax operator. Two FC layers output the one-dimensional likelihood for each concatenated feature from the Spatio-temporal aggregation module. Finally, the likelihood of each node is fed into the softmax operator to estimate a localization probability. The node with the highest probability can be estimated as the self-position.

In each module, we employ the ReLU activation function and batch normalization for all FC layers.

B. GCLSTM Architecture

Following [40], the GCLSTM layer can be computed with the following equations:

$$i_t = \sigma(\mathcal{G}_1(x_t, E) + \mathcal{G}_2(h_{t-1}, E) + w_{ci} \odot c_{t-1} + b_i), \quad (1)$$

$$f_t = \sigma(\mathcal{G}_3(x_t, E) + \mathcal{G}_4(h_{t-1}, E) + w_{cf} \odot c_{t-1} + b_f), \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(\mathcal{G}_5(x_t, E) + \mathcal{G}_6(h_{t-1}, E) + b_c), \quad (3)$$

$$o = \sigma(\mathcal{G}_7(x_t, E) + \mathcal{G}_8(h_{t-1}, E) + w_{co} \odot c_t + b_o), \quad (4)$$

$$h_t = o \odot \tanh(c_t) \quad (5)$$

where $x_t \in \mathbb{R}^{n \times d_x}$, $h_t \in [0, 1]^{n \times d_h}$, and $c_t \in \mathbb{R}^{n \times d_h}$ are the input, cell output, and cell state, respectively. Additionally, \odot and σ are the Hadamard product and sigmoid function, respectively, and $i, f, o \in [0, 1]^{n \times d_h}$ are the input, forgetting, and output gates. Weights and biases $b_i, b_f, b_c, b_o \in \mathbb{R}^{d_h}$ are parameters of the model to be optimized in training. In addition, $\mathcal{G}_{k,k=1:8}$ is an arbitrary graph convolution function that aggregates the features of neighboring nodes. In this study, we used a graph isomorphism network (GIN) [41], which is known for its simple architecture and high discriminative/representational power. The GIN updates the node representations as follows:

$$\mathcal{G}_k(x_t, E) = \text{MLP}_k((1 + \epsilon_k)x_{t,i} + \sum_{j \in \{1:n\}} x_{t,j}) \quad (6)$$

where $\text{MLP}_k(\cdot)$, ϵ_k , and j are the multilayer perceptron consisting of two FC layers, weight coefficient, and indexes of neighboring nodes for node i , respectively.

The GCLSTM layer is structured to output h_t by inheriting the past cell output h_{t-1} and cell state c_{t-1} in addition to the input x_t . Because the cell output h_{t-1} is the output of the prior step, it corresponds to short-term memory, and the cell state c_{t-1} corresponds to long-term memory that is sequentially updated inside the cell according to (3). In addition, because the input x_t aggregates the information of the neighboring nodes using the graph convolution function, it is possible to infer, for example, that the current self-position is located around a node that showed a feature that is highly likely to be the self-position several steps in the past. Therefore, by using the GCLSTM layer in the proposed method, it is possible to estimate the self-position using the features of the past and surrounding nodes in the long and short terms.

C. Training process

To train the models shown in Fig. 2, we utilize the tuples (O', G, Y) from the training dataset. Here, $O' = \{o_t\}_{t=T-\tau:T}$ is the list of images observed by the robot from $T-\tau$ to T , and G is the topological map. $Y = \{y_t\}_{t=T-\tau:T}$ is a list of GT indices for the nodes closest to each image in O' . In addition, τ is the step length for training.

We calculate \hat{Y} by feeding O' and G . Then, we optimize our model by minimizing the cross-entropy loss for τ steps from the estimated \hat{Y} and the GT Y .

D. Semi-supervised Learning Method

In the actual navigation, the robot cannot follow the exact image path from the topological map. The robot will deviate from its path depending on environmental changes, the robot's motor performance, and obstacle collision avoidance. To localize accurately in these cases, the time-series images for topological map G and the time-series images for the robot's observation O' for training should be obtained from different trials of teleoperation. However, we cannot obtain Y for the different time-series real images because there are no GT poses.

Hence, we propose a learning method that uses both simulated and real images to train our model to improve robot localization performance in the real world. Because the simulator can provide the pose of each image o_t , the ground-truth node y_t can be calculated for training. In this study, y_t is computed as

$$y_t = \arg \min_{i=1:n} \{\|p_t - p_i\| + \omega_m |\theta_t - \theta_i|\}, \quad (7)$$

where $p_t \in \mathbb{R}^2$ is the position in the $x-y$ coordinates [m] and $\theta_t \in \mathbb{R}^1$ is the attitude angle in the yaw direction [deg]. Simulators also allow a large quantity of images to be collected in diverse environments.

If we use the simulated images only, it is expected that sim2real transfer issues will occur. Hence, we create G and O' from same time-series images for the real images. For the real images, we obtain y_t from the node number of the nearest node in the time step. To achieve a sim2real

transfer, we randomly mix the simulator dataset and the real dataset to create a mini-batch to improve the localization performance for the real images. Additionally, using the real dataset allows our model to learn static and dynamic environmental changes because our open dataset [15], [42], [43] includes pedestrians, changes in lighting conditions, and so on.

E. Map Sampler

Because the memory of computational resources has an upper limit, when the number of nodes in the topological map G is very large, it is impossible to train models by loading the images of all the nodes into memory. In this study, we devised and introduced a map sampler to construct a partial topological map G' with the number of nodes n' by sampling nodes from G to include all elements of Y , the list of ground truths of the observed time-series images.

The method of constructing G' using the map sampler is as follows. First, to include Y in G' , all the nodes in Y are copied into an empty topological map G' . Next, to add a new neighboring node to G' , we randomly sample a node in G that is connected by an edge with the node in G' and is not yet included in G' , and then copy it to G' . We sample G' from G by iterating this sampling process until the number of nodes in G' reaches the upper limit n' . According to our map sampler, the sampled G' can always include nodes of Y and can accommodate as many edges as possible to form a realistic map.

During training, the proposed method is trained by converting the tuple (O', G, Y) into (O', G', Y) using the map sampler described above. The proposed map sampler also has an effect in terms of data augmentation because it randomly generates G' with different graph structures even when the same G is used.

V. EXPERIMENTS

We evaluated the proposed localization method for both localization and navigation tasks. First, we describe the dataset and the experimental setups. Subsequently, we present our results and compare them with several baselines.

A. Datasets

In the training and testing, we used both real images without the GT pose and simulator images with the GT pose.

1) *Real images*: We employed the Go Stanford (GS) Dataset [12], [15]. The GS dataset contains 360° camera images collected at the Stanford University campus. It contains 106,560 real images of 12 buildings and 39,307 simulator images of 36 environments from a mobile robot. Because the robot has no internal or external sensors to detect its global pose, the images in the GS dataset do not include the GT pose. Further details are shown in [12], [15].

For the topological map using real image sequences, we created a node for every m images in the sequences, and an edge was set from the previously created node to the newly created node. In this study, the value of m was set at 7.

2) *Simulator images*: The simulator dataset was collected by the interactive Gibson simulator (iGibson) [16], [17]. iGibson is a photorealistic robot simulator developed at the Stanford Vision and Learning Lab, which uses the Bullet physics engine to simulate interactions between objects.

To obtain the virtual environments to be rendered by iGibson, we measured 50 rooms in our facility using the Matterport scanner 2. Then, we virtually teleoperated the robot to collect time-series images three times in each environment. The duration of one sequence was approximately 6 min. We separated the entire dataset into the following categories: 36 rooms with 108 trajectories for training, 7 rooms with 21 trajectories for validation, and 7 rooms with 21 trajectories for testing.

Observed images O' and the topological map G were created using two individual trajectories per environment. Hence, nine combinations ($= 3 O' \times 3 G$) were prepared for training, validation, and testing.

A topological map of each room was created with nodes based on the GT pose of each node. Specifically, the first observation image in each trajectory is set as the first node v_1 , and a new node $v_i \forall i \in \{2, \dots, n\}$ is added to the topological map when the position p [m] $\in \mathbb{R}^2$ in the xy coordinate and the attitude angle θ [$^\circ$] $\in \mathbb{R}^1$ in the yaw direction satisfy the following equation:

$$\|p - p_{i-1}\| + \omega_m |\theta - \theta_{i-1}| > \alpha_{th}, \quad (8)$$

where i , ω_m , and α_{th} are the index of the node in the topological map, the weight factor to balance the relative magnitude of the position and attitude, and the threshold for creating a new node, respectively. In this study, we set $\omega_m = 0.025$ and $\alpha_{th} = 1.0$ by trial and error. The edge is set to be connected from v_{i-1} to v_i . In addition, to ensure closure of the map loop, the edge was connected from the most recently created node to node $v_j \forall j \in \{1, \dots, i-2\}$ when $\|p - p_j\| + \omega_m |\theta - \theta_j| > \alpha_{th}$ was satisfied.

B. Experimental Setup

We set the length of the time-series of data during training to $\tau=90$, and set the number of nodes in the map to be extracted by the map sampler described in section IV-E to $n'=200$. For data augmentation, we randomly set the deviations in brightness, contrast, and saturation to ± 0.1 and hue to ± 0.05 for the set of robot-observed and node images. As the convolutional neural network that extracts image feature vectors, we used ResNet-18 [44], which was pre-trained with ImageNet [45].

The learning rate of our model except for ResNet-18 was set to 0.001. We provided a smaller learning rate of 0.00001 for ResNet-18 to be fine-tuned. The proposed method was trained until the minimum value of the validation loss was no longer updated for 1000 consecutive iterations.

C. Result: Localization

The baseline methods used for comparison with the proposed method in terms of the localization performance are as follows.

- 1) **Pixel MSE [46]**: The node with the smallest pixel-wise mean squared error (MSE) in each channel is localized as the current node.
- 2) **SSIM [47]**: Structural similarity index measure (SSIM) is used to measure the similarity between two images, considering the distribution of pixel values, contrast, and structure. The node image with the highest similarity to the observation is localized to the current node.
- 3) **SiameseNet [5]**: SiameseNet [5] estimates the similarity between two input images. The node image with the highest similarity to the observation is localized as the current node. We trained this model in the same manner as in [5] and with the same dataset as our method.
- 4) **NetVLAD [37]**: We employed the pretrained model (VGG-16 + NetVLAD + whitening, trained on Pittsburgh dataset) for better results. This is a convolutional neural network (CNN) architecture that is directly trainable in an end-to-end manner directly for place recognition tasks.

For the ablation study, we evaluated our method, and our method excluding the GCLSTM layer, the skip path with the FC layer, and real images from the training dataset (semi-supervised learning methodology described in section IV-D). Our method excluding the GCLSTM layer consists of four FC layers with ReLU and batch normalization.

Table I presents the results of the numerical experiments on the unseen (test) datasets. The table shows the accuracy (AC) [%], accuracy within one edge (AC*) [%], pose error (PE) [$m + \omega_m * ^\circ$], and map error (ME) [edge] for the four data categories. The category "Not deviated" refers to the simulator dataset in which the time-series images and the topological maps are created from the same trajectory. "Deviated ≤ 1.0 " and " $1.0 < \text{Deviated} \leq 2.0$ " refer to a simulator dataset of time-series images and topological maps from different trajectories; the distance $\|p_t - p_{y_t^*}\| + \omega_m |\theta_t - \theta_{y_t^*}|$ between the observed image o_t and the GT node y_t^* is less than 1.0 in the former, and greater than 1.0 and less than or equal to 2.0 in the latter. "Real image" refers to a dataset consisting of real images (GS dataset [12], [15]). The numbers in the table represents the average of the localization results.

As can be observed from Table I, the proposed method outperformed all baseline methods on all metrics. For the ablation study, our method was slightly inferior to our method without real images when testing with data in " $1.0 < \text{Deviated} \leq 2.0$ ", but showed high performance in the other metrics. It is considered that this occurred because our method without real images was trained only on the simulator images and optimized for them.

Moreover, the map error in the real data is relatively large for all methods. Because the topological map with the real images does not have the loop-closed points owing to the lack of GT poses, all methods estimate almost the same node at close poses, albeit with large map errors in some cases. Note that the time-series images of the real image do not deviate from the topological maps. In Section V-D, we evaluate the

TABLE I: **Performance comparison of localization with baseline methods in unseen environment.** Table shows the accuracy (AC) [%], accuracy within one edge (AC*) [%], pose error (PE) [$m + \omega_m^{\circ}$] and map error (ME) [edge] in localization.

Model	Not deviated		Deviated ≤ 1.0		1.0 < Deviated ≤ 2.0		Real image	
	AC / AC*	PE / ME	AC / AC*	PE / ME	AC / AC*	PE / ME	AC / AC*	PE / ME
Pixel MSE [46]	0.751 / 0.840	1.463 / 1.987	0.632 / 0.758	1.945 / 2.462	0.284 / 0.472	3.300 / 4.561	0.736 / 0.820	- / 28.423
SSIM [47]	0.837 / 0.920	0.668 / 0.958	0.758 / 0.874	0.902 / 1.114	0.290 / 0.472	3.183 / 4.294	0.744 / 0.830	- / 22.259
SiameseNet [5]	0.785 / 0.955	0.427 / 0.543	0.704 / 0.920	0.619 / 0.755	0.331 / 0.593	2.149 / 3.164	0.708 / 0.838	- / 13.650
NetVLAD [37]	0.788 / 0.929	0.568 / 0.847	0.725 / 0.891	0.777 / 1.127	0.328 / 0.493	2.824 / 4.190	0.760 / 0.856	- / 12.603
Our method	0.851 / 0.981	0.225 / 0.252	0.798 / 0.950	0.353 / 0.462	0.383 / 0.604	1.937 / 2.926	0.775 / 0.894	- / 7.038
w/o GCLSTM	0.779 / 0.941	0.505 / 0.719	0.696 / 0.892	0.770 / 1.075	0.335 / 0.579	2.157 / 3.199	0.653 / 0.778	- / 20.218
w/o skip	0.823 / 0.970	0.303 / 0.361	0.756 / 0.931	0.508 / 0.685	0.359 / 0.600	2.206 / 3.025	0.761 / 0.872	- / 8.426
w/o real image	0.839 / 0.972	0.289 / 0.384	0.782 / 0.941	0.427 / 0.549	0.380 / 0.613	2.156 / 3.118	0.718 / 0.831	- / 16.728

navigation performance of our method in largely deviated scenes in real environments.

Fig. 3 shows examples of node images localized by the baselines and proposed method for images observed by the robot. From left to right, Fig. 3 shows the image observed by the robot and the localized node images produced by SiameseNet, NetVLAD, and our method, respectively. The prediction error in pose [$m + \omega_m^{\circ}$] (PE) and that in the edge distance of the map (ME) [edge] are given at the bottom of the images. As shown in Fig. 3, in some cases the baseline method significantly misestimated the self-position when the topological map contains multiple similar node images. However, the proposed method localized the self-position accurately even when the topological map contained multiple similar node images.

D. Result: Navigation

1) Overview of navigation system with our localization:

In addition to the sole evaluation of localization, we evaluate our proposed localization approach on the navigation system. Fig. 4 shows a block diagram of the proposed navigation system with our localization. The following three modules are used: i) localization, ii) planning, and iii) control modules.

- i) **Localization module** estimates the number of nodes that correspond to the current robot position in the given topological map. We implemented our model in this module to evaluate our method on navigation.
- ii) **Planning module** generates subgoal images from the current node to the destination node. Dijkstra’s method [14] was applied to minimize the number of images to shorten the navigation time.
- iii) **Control module** derives the linear and angular velocity from the next subgoal image from “selection” and the current robot image. We provide the control policy of the DVMPC [12] to robustly control the mobile robot toward the subgoal position without collisions.

In our system, we calculate these modules every 3 fps until the robot arrives at the target position.

2) *Comparison to baselines in simulation:* First, we compared the navigation performance of our method with the following three baseline methods in a simulation.

- i) **SPTM [5]:** We construct the same navigation system as [5] and train their models with the same dataset as our method. Following [5], the localization is based on the SiameseNet.

TABLE II: **Quantitative results for image-based navigation in unseen simulator environment.** The table shows the success rate (SR), collision rate (CR), time over rate (TR), and coverage rate (CovR) [%].

Env	Method	SR	CR	TR	CovR
Area1	SPTM [5]	0.27	0.69	0.04	72.63
	SPTM with DVMPC [5], [12]	0.77	0.06	0.17	85.44
	SPTM+ with DVMPC [5], [12]	0.78	0.10	0.12	84.74
	Our method	0.85	0.05	0.01	89.49
Area2	SPTM [5]	0.44	0.55	0.01	78.18
	SPTM with DVMPC [5], [12]	0.95	0.04	0.01	95.80
	SPTM+ with DVMPC [5], [12]	0.91	0.06	0.03	94.11
	Our method	0.96	0.04	0.00	96.75
Area3	SPTM [5]	0.49	0.50	0.01	83.90
	SPTM with DVMPC [5], [12]	0.78	0.12	0.10	87.96
	SPTM+ with DVMPC [5], [12]	0.80	0.17	0.03	86.35
	Our method	0.84	0.10	0.06	88.10
Mean	SPTM [5]	0.40	0.58	0.02	78.24
	SPTM with DVMPC [5], [12]	0.83	0.07	0.09	89.73
	SPTM+ with DVMPC [5], [12]	0.83	0.11	0.06	88.40
	Our method	0.88	0.06	0.06	91.44

ii) **SPTM with DVMPC [5], [12]:** We replace the control module of SPTM with DVMPC [12].

iii) **SPTM+ with DVMPC [5], [12]:** We apply our localization method without GCLSTM instead of SiameseNet in SPTM with DVMPC. The only difference between the two methods is the localization method.

We chose three simulation environments and performed 100 trials in each environment. The distance between the robot’s initial position and the goal node is within 10 [m] and is randomly generated in each trial. Before navigation, we collected time-series images by teleoperating the virtual robot and created the topological map in each environment, following section V-A. In each trial, we stopped the navigation when the robot collided with the obstacles and when the total navigation time exceeded the threshold.

Table II shows the mean of the four metrics. SR denotes the success rate to arrive at the target final position, CR denotes the collision rate where the robot collides with obstacles, TR denotes the time over rate where the 180 [s] threshold is exceeded, and CovR denotes the coverage rate against the desired trajectories between the start and goal positions.

From Table II, we can confirm that our navigation system with our localization method outperformed all baseline methods in all three environments. The main advantage of our localization method lies in the difference in performance

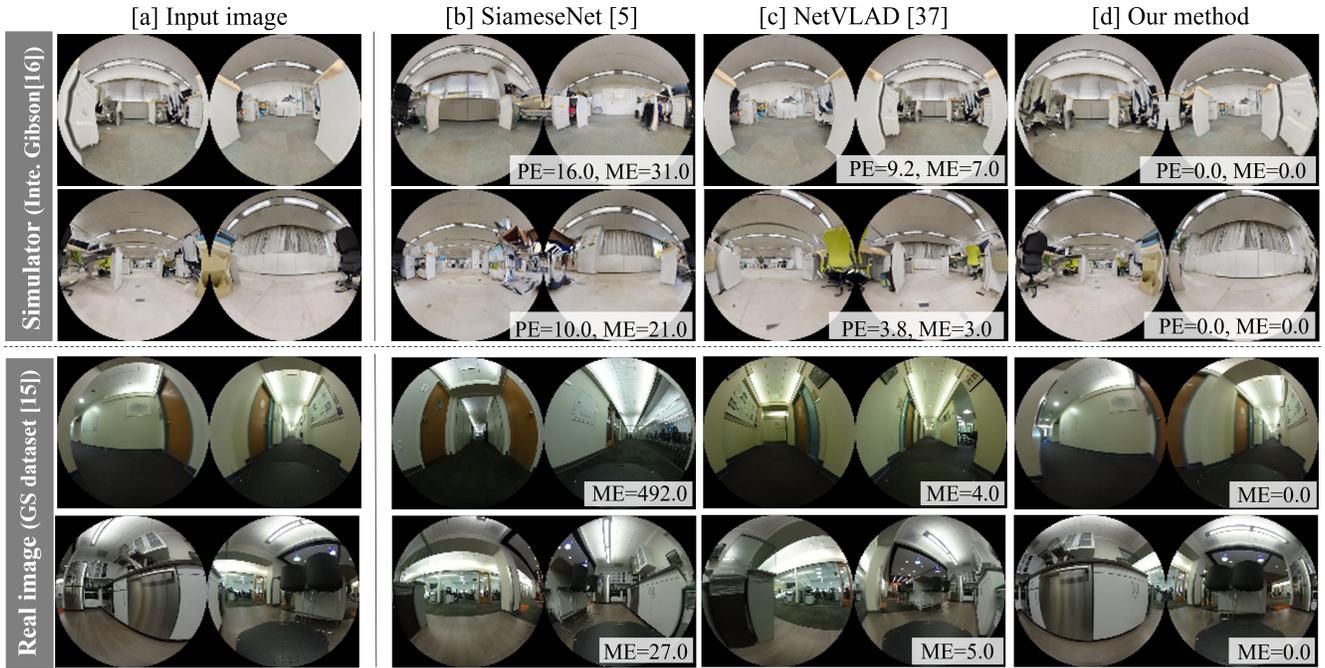


Fig. 3: Excerpts of node images localized by the baseline method (SiameseNet [5]) and the proposed method (Ours) for the robot’s observed image. The baseline method misestimated a location where the images are similar, while the proposed method is more accurate in its estimation.

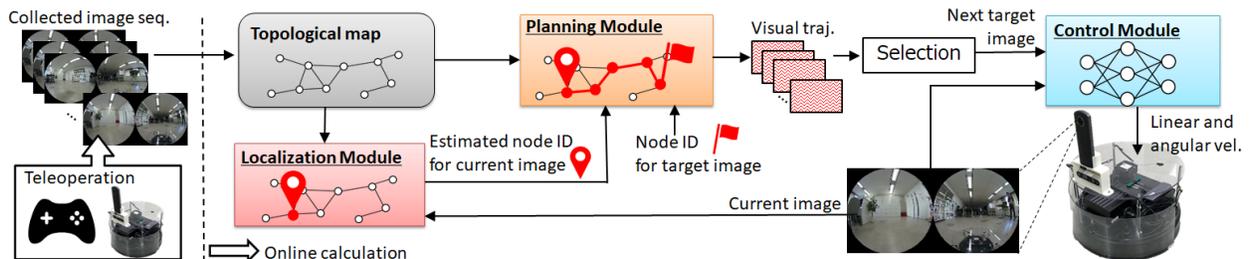


Fig. 4: Block diagram of image-based navigation system with our localization. “Localization module” is our proposed localization with the GCLSTM. “Planning module” generates the visual trajectory from the current node to a target node. “Control module” derives linear and angular velocities to control the mobile robot.

TABLE III: Navigation performance in unseen real environments. The table shows the success rate (SR) and coverage rate (CovR) [%].

Env	Method	SR	CovR
Env1	SPTM with DVMPD [5], [12]	0.70	85.00
	Our method	0.90	95.00
Env2	SPTM with DVMPD [5], [12]	0.50	73.80
	Our method	0.60	83.00
Env3	SPTM with DVMPD [5], [12]	0.60	88.00
	Our method	0.70	91.00
Mean	SPTM with DVMPD [5], [12]	0.60	82.26
	Our method	0.73	89.66

between our method and SPTM+ with DVMPD.

3) *Experiments with physical robot:* We evaluated our method with a physical robot in a real-world environment. Navigation experiments were conducted with a vizbot, a small robot platform. The mobile base of the vizbot is the

Roomba from iRobot. We utilized an Nvidia Jetson Xavier as the control personal computer and a Ricoh Theta S as a 360-degree camera to implement our image-based navigation.

We compared our method to the best baseline method in the previous section, SPTM with DVMPD [5], [12]. We chose three environments and performed 10 trials in each environment. Other conditions in the physical robot experiments were the same as those in the simulator experiments.

Table III shows the mean of the success rate (SR) and coverage rate (CovR). As in the simulator environment, our method showed a better success rate and coverage rate against the baseline method in the real world.

VI. CONCLUSIONS

We proposed a localization method utilizing recurrent-type graph neural networks that uses the spatial information of the environment and the temporal information from the robot’s trajectory. The proposed method was trained on simulator

images with the GT pose as well as real images without the GT pose for sim2real transfers. The evaluation results show that the proposed method outperforms the baseline method in localization and navigation tasks. Because the proposed method uses the graph structure of the topological map and the time-series information of the robot's observation images for localization, it can achieve an accurate estimation even when the topological map contains multiple similar node images.

In the future, we are planning to improve the localization performance in the largely deviated scene from the topological map. Even while avoiding large obstacles, the robot needs to precisely localize its own position for more robust navigation.

VII. ACKNOWLEDGMENT

We thank Kazutoshi Sukigara, Kota Sato, Yuichiro Matsuda, and Yasuaki Tsurumi for measuring the 3D environments utilized for collecting training images with the GT pose and evaluating our method for navigation. We thank Hideki Deguchi for the evaluation for navigation, and Satoshi Koide and Keisuke Kawano for the technical discussion. We would like to thank Editage (www.editage.com) for English language editing.

REFERENCES

- [1] J. Engel *et al.*, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.
- [2] J. Fuentes-Pacheco *et al.*, "Visual simultaneous localization and mapping: a survey," *Artificial intelligence review*, vol. 43, no. 1, pp. 55–81, 2015.
- [3] R. Mur-Artal *et al.*, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [4] —, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
- [5] N. Savinov *et al.*, "Semi-parametric topological memory for navigation," *arXiv preprint arXiv:1803.00653*, 2018.
- [6] D. S. Chaplot *et al.*, "Neural topological slam for visual navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 875–12 884.
- [7] A. Taniguchi *et al.*, "Pose invariant topological memory for visual navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 384–15 393.
- [8] X. Li *et al.*, "Seannet: Semantic understanding network for localization under object dynamics," *arXiv preprint arXiv:2110.02276*, 2021.
- [9] K. Chen *et al.*, "A behavioral approach to visual navigation with graph localization networks," *arXiv preprint arXiv:1903.00445*, 2019.
- [10] D. S. Chaplot *et al.*, "Neural topological slam for visual navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 875–12 884.
- [11] X. Meng *et al.*, "Scaling local control to large-scale topological navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 672–678.
- [12] N. Hirose *et al.*, "Deep visual mpc-policy learning for navigation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3184–3191, 2019.
- [13] —, "Probabilistic visual navigation with bidirectional image prediction," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1539–1546.
- [14] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [15] <https://cvgl.stanford.edu/gonet/dataset/>, (accessed Jan. 18, 2022).
- [16] F. Xia *et al.*, "Interactive gibbon benchmark: A benchmark for interactive navigation in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 713–720, 2020.
- [17] —, "Gibson env v2: Embodied simulation environments for interactive navigation," *Stanford University, Tech. Rep.*, 2019.
- [18] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.
- [19] —, "Visual servo control. ii. advanced approaches [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.
- [20] F. Codevilla *et al.*, "End-to-end driving via conditional imitation learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4693–4700.
- [21] S. Hutchinson *et al.*, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.
- [22] G. Kahn *et al.*, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5129–5136.
- [23] K. Kase *et al.*, "Learning multiple sensorimotor units to complete compound tasks using an rnn with multiple attractors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4244–4249.
- [24] D. Mishkin *et al.*, "Benchmarking classic and learned navigation in complex 3d environments," *arXiv preprint arXiv:1901.10915*, 2019.
- [25] A. Pokle *et al.*, "Deep local trajectory replanning and control for robot navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5815–5822.
- [26] E. Wijnmans *et al.*, "Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames," *arXiv preprint arXiv:1911.00357*, 2019.
- [27] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364.
- [28] A. Kendall *et al.*, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *ICCV*, 2015, pp. 2938–2946.
- [29] V. Balntas *et al.*, "Relocnet: Continuous metric learning relocalisation using neural nets," in *ECCV*, 2018, pp. 751–767.
- [30] M. Ding *et al.*, "Camnet: Coarse-to-fine retrieval for camera relocalization," in *ICCV*, 2019, pp. 2871–2880.
- [31] P.-E. Sarlin *et al.*, "From coarse to fine: Robust hierarchical localization at large scale," in *CVPR*, 2019, pp. 12 716–12 725.
- [32] J. Revaud *et al.*, "R2d2: repeatable and reliable detector and descriptor," *NIPS*, 2019.
- [33] E. Brachmann *et al.*, "Dscac-differentiable ransac for camera localization," in *CVPR*, 2017, pp. 6684–6692.
- [34] E. Brachmann and C. Rother, "Learning less is more-6d camera localization via 3d surface regression," 2018, pp. 4654–4662.
- [35] —, "Expert sample consensus applied to camera re-localization," in *ICCV*, 2019, pp. 7525–7534.
- [36] —, "Visual camera re-localization from rgb and rgb-d images using dscac," 2021.
- [37] R. Arandjelovic *et al.*, "Netvlad: Cnn architecture for weakly supervised place recognition," in *CVPR*, 2016, pp. 5297–5307.
- [38] A. Torii *et al.*, "24/7 place recognition by view synthesis," in *CVPR*, 2015, pp. 1808–1817.
- [39] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *CVPR*, 2015, pp. 4353–4361.
- [40] Y. Seo *et al.*, "Structured sequence modeling with graph convolutional recurrent networks," in *International Conference on Neural Information Processing*. Springer, 2018, pp. 362–373.
- [41] M. Defferrard *et al.*, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.
- [42] D. Lee *et al.*, "Large-scale localization datasets in crowded indoor spaces," 2021.
- [43] H. Taira *et al.*, "Inloc: Indoor visual localization with dense matching and view synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7199–7209.
- [44] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [45] <https://www.image-net.org/>, (accessed Jan. 18, 2022).
- [46] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.
- [47] Z. Wang *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.