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 a 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 (GT) 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 GT pose in addition to real images. We evaluated the proposed 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 compared to 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 patrolling. As opposed to navigation systems using single or multiple LiDARs, some studies have tackled vision-based navigation using a monocular camera owing to its low cost, light weight, compact size, robustness, and high availability.

Visual navigation widely uses metric maps. Although a metric map can represent a detailed environment, maintaining its accuracy is challenging due to estimation errors of the camera parameters and noise from the sensors and actuators [1]–[3]. These issues also create challenges in visual navigation, such as collision avoidance and robustness against environmental changes.

To address these issues, image-based navigation using topological maps has recently attracted significant attention [2]–[10]. A topological map represents an environment abstractly via a graph created from the image sequences obtained by a robot. Each node in the map contains a monocular camera image. Adjacent nodes are connected on edges based on image similarity [2] or reachability estimation [8]. During navigation, the robot identifies its own position as a node number on the topological map. Subsequently, it generates subgoal images between the identified and destination nodes [11]. The navigation system derives velocity references to control the robot using the subgoal images and the current image from the robot [9], [10]. 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 or short-term robot images as the query and the node images as the references [2]–[7]. As these methods advanced the state of the art in visual navigation, two specific challenges became apparent. 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 may select the wrong node, leading to navigation failures.

Another issue is that collecting large number of images and the ground truth (GT) poses in a real environment is challenging. Hence, supervised learning cannot be conducted using real images.

In this study, we propose a graph neural network-based localization method that can utilize spatio-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. The temporal consistency can be learned from robot time-series images via the long short-term memory (LSTM) layer. The long-term history of a 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 it. The GT pose in the simulator images enables supervised learning to achieve accurate localization even in the real images [9], [12].

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 [13], [14] 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 propose a novel graph neural network-based localization method. To the best of our knowledge, this is the first report of using a recurrent-type graph neural network for localization in a topological map.
- We develop a semi-supervised learning method using real-world images without the GT pose and simulator images with the GT pose for sim2real transfer.
- We implement an image-based navigation system via 360-degree images that incorporates the proposed localization method into the evaluation.

II. RELATED WORKS

Research on visual navigation for mobile robots has a considerably long history. We begin with a comprehensive discussion of visual navigation. Next, we focus on visual localization as a component of visual navigation, which is particularly relevant to the proposed 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 [15]–[17] 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 obstacles occlude large parts of the environment. Navigation methods based on visual SLAM [18]–[21] first use camera images to construct a map that a robot can use 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 fails.

To address these issues, multiple learning-based approaches have been recently proposed as image-based navigation. Image-based navigation using a topological map [2]–[6] involves localization and planning from a topological representation of the environment that represents the connectivity between regions. The latest advances in reinforcement learning [22]–[25] and imitation learning [26]–[28] 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 [29], [30], coarse-to-fine [31]–[33], and structure-based approaches [34]–[37], 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 [38], [39].

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. One of the earliest studies in this domain [2] involved performing localization by estimating similarity using the Siamese network [40]. This can be replaced by other image retrieval methods, such as NetVLAD [38].

Image retrieval methods are estimated from only a single image; however, it is reasonable to use time-series observations for visual navigation. The proposed 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 the spatially adjacent nodes are connected by edges in the topological map. Note that the self-position localized in this research is not a 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 $i$. Additionally, $E = \{(r_k, s_k)\}_{k=1:m}$ 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 face difficulty in 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
The Spatio-temporal aggregation module aggregates the features from the Feature extraction module to consider spatio-temporal consistency. Note that Feature extraction and Identification modules are performed for each node image from the topological map for considering spatio-temporal consistency for accurate localization. In this section, we 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 spatial information from topological maps and time-series images observed by a 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 \(x_t\) 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 a graph convolutional LSTM (GCLSTM) [41] layer and a skip path with FC layers. We employ GCLSTM to simultaneously process temporal information from the 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 the 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 in the graph convolution using an LSTM-based structure. Therefore, by introducing GCLSTM, the proposed method can infer that nodes closer to the node with higher probability in previous observations are more trusted as a self-node.

While the GCLSTM layer outputs the overall features by convolving the features of neighboring nodes, it dilutes the 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 subsequent 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 the GCLSTM architecture are provided in subsequent sections.

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

\[
i_t = \sigma(G_1(x_t, E) + G_2(h_{t-1}, E) + w_{ci} \odot c_{t-1} + b_i), \quad (1)
\]
\[
f_t = \sigma(G_3(x_t, E) + G_4(h_{t-1}, E) + w_{cf} \odot c_{t-1} + b_f), \quad (2)
\]
\[
c_t = f_t \odot c_{t-1} + i_t \odot \tanh(G_5(x_t, E) + G_6(h_{t-1}, E) + b_c), \quad (3)
\]
\[
o = \sigma(G_7(x_t, E) + G_8(h_{t-1}, E) + w_{co} \odot c_t + b_o), \quad (4)
\]
\[
h_t = o \odot \tanh(c_t) \quad (5)
\]

where \(x_t \in \mathbb{R}^{n \times d_x}\), \(h_t \in [0, 1]^{n \times d_h}\), and \(c_t \in \mathbb{R}^{n \times d_c}\) are the input, cell output, and cell state, respectively. Additionally, \(\odot\) and \(\sigma\) are the Hadamard product and sigmoid function, respectively, and \(i, f, o, \in [0, 1]^{n \times d_x}\) 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, \(G_{k, k=1; n}\) is an arbitrary graph convolution function that aggregates the features of neighboring nodes. In this study, we used a graph isomorphism network (GIN) [42], which is known for its simple architecture and high discriminative/representational power. The GIN updates the node \(i\)
representations of the \( l \)-th layer as follows:
\[
    x^{(l)}_{t,i} = \text{MLP}_k^{(l)}((1 + \epsilon^{(l-1)}_{k})x^{(l-1)}_{t,i} + \sum_{j \in \{1:n\}} x^{(l-1)}_{t,j})
\]  
(6)
where \( \text{MLP}_k(\cdot) \), \( \epsilon_k \), and \( j \) are the multilayer perceptron consisting of two FC layers, weight coefficient, and indexes of the 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, using the GCLSTM layer in the proposed method, we can 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, \( \tau \) is the step length for training.

We calculate \( \hat{Y} \) by feeding \( O' \) and \( G \). With the estimated \( \hat{Y} \) and the GT \( Y \), we optimize the proposed model by minimizing the cross-entropy loss for \( \tau \) steps.

D. Semi-supervised Learning Method

During navigation, the robot cannot follow the exact path imaged from the topological map. It will deviate from its path depending on environmental changes, performance of its motor, and obstacles. For accurate localization in these cases, the time-series images for the topological map \( G \) and for the robot’s observation \( O' \) for training should be obtained from different trials of teleoperation. However, we cannot obtain \( \hat{Y} \) for the different time-series real images because of the lack of GT poses.

Hence, we propose a learning method that uses both simulated and real images to train our model to improve the robot’s real-world localization performance. Because the simulator can provide the pose of each image \( o_t \), the GT node \( y_t \) can be calculated for training. In this study, \( y_t \) is computed as
\[
    y_t = \arg\min_{i=1:n} \{\|p_t - p_i\| + \omega_m |\theta_t - \theta_i|\},
\]  
(7)
where \( p \in \mathbb{R}^3 \) is the position in the \( x \)–\( y \) coordinates \([m]\), and \( \theta \in \mathbb{R}^\circ\) is the attitude angle in the yaw direction \([\text{deg}]\). Simulators also allow a large quantity of images to be collected in diverse environments.

If we only use the simulated images, it is expected that \text{sim2real} transfer issues will occur. Hence, we create \( G \) and \( O' \) from the 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 \text{sim2real} transfer, we randomly mix the simulator and real datasets to create a mini-batch that can improve the localization performance for the real images. Additionally, using the real dataset allows the proposed model to learn static and dynamic environmental changes because the open dataset [12], [43], [44] used in this study 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, the models cannot be trained by uploading the images of all nodes into the 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 \( Y \), and the list of GTs 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 \( G' \), a node in \( G \) that is connected by an edge with the node in \( G' \) and is not yet included in \( G' \) is randomly sampled and then copied 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 the map sampler, the sampled \( G' \) can always include nodes of \( Y \) and can accommodate as many edges as possible to form a realistic map.

The proposed method is trained by converting the tuple \((O', G, Y)\) into \((O', G', Y)\) using the map sampler. The map sampler also plays a role in data augmentation as it randomly generates \( G' \) with different graph structures even when the same \( G \) is used.

During inference, the memory of computational resources is sufficient since the gradients did not need to be retained and only one observed image was processed instead of the mini-batch. Therefore, we do not use the map sampler during inference.

V. EXPERIMENTS

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

A. Datasets

During training and testing, we used both real images without the GT pose and simulated images with the GT pose.

1) Real images: We employed the Go Stanford (GS) Dataset [9], [12]. The GS dataset contains 360-degree camera images captured at the Stanford University campus. It contains 106,560 real images of 12 buildings and 39,307 simulated images of 36 environments captured by 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 [9], [12].
For the topological map using real image sequences, we created a node for every image in the sequences and set an edge from the previously created node to the newly created node. In this study, the value of \( m \) was set to 7.

2) Simulator images: The simulator dataset was collected by the interactive Gibson simulator (iGibson) [13], [14]. 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 for rendering by iGibson, we measured 50 rooms in our office using Matterport scanner 2. Then, we virtually teleoperated the robot for collecting 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.

The observed images \( O' \) and topological map \( G \) were generated using three individual trajectories per environment. Thus, nine combinations (i.e., \( 3 \times 3 \times 3 \)) were prepared for training, validation, and testing.

A topological map of each room was generated 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 \( v_i \) \( i \in \{2, ..., n\} \) is added to the topological map when the position \( p[m] \in \mathbb{R}^2 \) in the \( xy \) coordinate and the attitude angle \( \theta[p] \in \mathbb{R}^1 \) in the yaw direction satisfy the following equation:

\[
\|p - p_{i-1}\| + \omega_m|\theta - \theta_{i-1}| > \alpha_{th},
\]

where \( i \), \( \omega_m \), and \( \alpha_{th} \) 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 \( v_i \) \( \forall j \in \{1, ..., i - 2\} \) when

\[
\|p - p_j\| + \omega_m|\theta - \theta_j| > \alpha_{th}
\]

satisfied.

B. Experimental Setup

We set the length of the time-series of data during training to \( \tau=90 \), and 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 \( \pm 0.1 \) and hue to \( \pm 0.05 \) for the set of robot-observed and node images. We used ResNet-18 [45], which was pre-trained with ImageNet [46], as the convolutional neural network that extracts image feature vectors.

The learning rate of the proposed model, except for ResNet-18, was set to 0.001. We used a smaller learning rate of 0.00001 for the fine-tuning of ResNet-18. The model was trained until the minimum value of validation loss was no longer being updated for the 1,000th consecutive iteration.

C. Result: Localization

The baselines used for comparison with our method in terms of the localization performance are as follows.

1) Pixel MSE [47]: The node with the smallest pixel-wise mean squared error (MSE) in each channel is localized as the current node.

2) SSIM [48]: 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 as the current node.

3) SiameseNet [2]: SiameseNet [2] 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 [2] and with the same dataset as the proposed method.

4) NetVLAD [38]: We employed the pre-trained model (VGG-16 + NetVLAD + whitening, trained on Pittsburgh dataset, which does not contain omnidirectional images) for better results. This is a convolutional neural network (CNN) architecture that can be directly trained in an end-to-end fashion for scene recognition tasks.

For the ablation study, we evaluated the proposed method with and without the GLSTM 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 without the GLSTM 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. Table I lists the accuracy rate (AC), accuracy rate within one edge (AC*), pose error (PE) \( |\theta + \omega_m, \text{degree}| \) and map error (ME) \( |\text{edge}| \) for the four data categories. ME represents the number of edges on the shortest path between a predicted node and the GT. The category “Not deviated” refers to the simulator dataset in which the time-series images and topological maps are generated from the same trajectory. “Deviated \( \leq 1.0 \)” and “1.0 <Deviated \( \leq 2.0 \)” refer to a simulator dataset of time-series images and topological maps from different trajectories; the distance \( \|p_i - p_{y^t}\| + \omega_m|\theta_i - \theta_{y^t}| \) 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 [9], [12]). The numbers in the table represent the averages of the localization results.

As indicated in Table I, the proposed method outperformed all baselines and ablation methods on all metrics. The map errors in the real data were relatively large for all methods. Because the topological map with the real images did not contain the loop-closed points due to the lack of GT poses, all methods estimated approximately the same node at close poses, albeit with large map errors in some cases. Note that the time-series images of the real image did not deviate from the topological maps. In Section V-D, we evaluated the navigation performance of the proposed method in largely
TABLE I: Performance comparison of localization with the baselines on the unseen environment. Table shows the accuracy rate (AC), accuracy rate within one edge (AC*), pose error (PE) \[ m + \omega_m \cdot \text{degree} \] and map error (ME) \[ \text{edge} \].

| Model               | Not deviated | Deviated \(\leq 1.0\) | Deviated \(1.0 < \text{Deviated} \leq 2.0\) | Real image |
|---------------------|--------------|-------------------------|---------------------------------------------|------------|
|                     | AC / AC*     | PE / ME                 | AC / AC*                                   | PE / ME    |
| Pixel MSE [47]      | 0.751 / 0.840| 1.463 / 1.987            | 0.632 / 0.758                              | 1.945 / 2.462 |
| SSIM [48]           | 0.837 / 0.920| 0.668 / 0.958            | 0.758 / 0.874                              | 0.902 / 1.114 |
| SiameseNet [2]      | 0.785 / 0.955| 0.427 / 0.543            | 0.704 / 0.920                              | 0.619 / 0.755 |
| NetVLAD [38]        | 0.788 / 0.929| 0.568 / 0.847            | 0.725 / 0.891                              | 0.777 / 1.127 |
| Our method          | 0.854 / 0.980| 0.244 / 0.275            | 0.806 / 0.958                              | 0.362 / 0.434 |
| w/o GCLSTM          | 0.779 / 0.941| 0.505 / 0.719            | 0.696 / 0.892                              | 0.619 / 0.755 |
| w/o skip            | 0.823 / 0.970| 0.303 / 0.361            | 0.756 / 0.931                              | 0.508 / 0.685 |
| w/o real image      | 0.839 / 0.972| 0.289 / 0.384            | 0.782 / 0.941                              | 0.427 / 0.549 |

Fig. 3: Excerpts of node images localized by the baselines (SiameseNet [2], NetVLAD [38]) and our method for the robot’s observed image. The baselines misestimate a location where the images are similar, whereas our method is more accurate in its estimation.

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 the proposed method, respectively. The prediction error in pose \( m + \omega_m \cdot \text{degree} \) (PE) and that in the edge distance of the map (ME) \[ \text{edge} \] are given at the bottom of the images. As shown in Fig. 3, in some cases, the baselines 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 [2], [9]. 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 node index that corresponds to the current robot position in a given topological map. We implemented the proposed method in this module to evaluate it on navigation.

   ii) Planning module generates subgoal images from the current node to the destination node. Dijkstra’s method [11] was employed to minimize the number of images required to reduce the navigation time.

   iii) Control module derives the linear and angular velocities from the next subgoal image from “selection” and the current robot image. We provide a control policy for Deep Visual MPC-Policy (DVMPC) [9] to robustly control the mobile robot toward the subgoal position without collision.

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

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

   i) Semi-parametric topological memory (SPTM) [2]: We constructed the same navigation system as [2]...
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 [2]                | 0.27 | 0.69| 0.04| 72.63|
|      | SPTM with DVMPC [2], [9]| 0.77 | 0.06| 0.17| 85.44|
|      | SPTM+ with DVMPC [2], [9]| 0.78 | 0.10| 0.12| 84.74|
|      | Our method              | 0.85 | 0.05| 0.01| 89.49|
| Area2| SPTM [2]                | 0.44 | 0.55| 0.01| 78.18|
|      | SPTM with DVMPC [2], [9]| 0.95 | 0.04| 0.01| 95.80|
|      | SPTM+ with DVMPC [2], [9]| 0.91 | 0.06| 0.03| 94.11|
|      | Our method              | 0.96 | 0.04| 0.00| 96.75|
| Area3| SPTM [2]                | 0.49 | 0.50| 0.01| 83.90|
|      | SPTM with DVMPC [2], [9]| 0.78 | 0.12| 0.10| 87.96|
|      | SPTM+ with DVMPC [2], [9]| 0.80 | 0.17| 0.03| 86.35|
|      | Our method              | 0.84 | 0.10| 0.06| 88.10|
| Mean | SPTM [2]                | 0.40 | 0.58| 0.02| 78.44|
|      | SPTM with DVMPC [2], [9]| 0.83 | 0.07| 0.09| 89.73|
|      | SPTM+ with DVMPC [2], [9]| 0.83 | 0.11| 0.06| 88.40|
|      | Our method              | 0.88 | 0.06| 0.06| 91.44|

and trained their models with the same dataset as our method. Following [2], the localization was based on SiameseNet. The original paper did not evaluate SPTM for collision avoidance [2].

ii) SPTM with DVMPC [2], [9]: We replaced the control module of SPTM with DVMPC [9].

iii) SPTM+ with DVMPC [2], [9]: We applied our localization method without GCLSTM, instead of SiameseNet in SPTM with DVMPC. The only difference between the two methods is the localization method.

We selected three simulation environments from our virtual office, as indicated in the upper half of Fig. 3, and performed 100 trials in each environment. The distance between the robot’s initial position and the goal node was within 10 [m], and was randomly generated in each trial. Before navigation, we captured time-series images by teleoperating the virtual robot, and then generated the topological map in each environment, following section V-A. In each trial, we terminated navigation when the robot collided into the obstacles and when the total navigation time exceeded the threshold.

Table II lists the mean of the four metrics. Here, SR denotes the success rate of arriving at the target final position, CR denotes the collision rate at which the robot collides into obstacles, TR denotes the time over rate at which the 180 [s] threshold is exceeded, and CovR denotes the coverage rate against the desired trajectories between the starting point and the goal.

As indicated in Table II, we confirm that the navigation system with our localization method outperformed all baselines in all three environments. The main advantage of the proposed localization method lies in the difference in performance between our method and SPTM+ with DVMPC.

3) Experiments with physical robot: We evaluated our method with a physical robot in a real-world environment. Navigation experiments were conducted on vizbot, a small robot platform. The mobile base of vizbot was the Roomba from iRobot. We used Nvidia Jetson Xavier as the controlling personal computer and the 360-degree camera Ricoh Theta S to implement the proposed image-based navigation method.

We compared our method with the best baseline method from the previous section, namely SPTM with DVMPC [2], [9]. We chose three office environments, as shown in the supplemental video, and performed 10 trials in each environment. Other conditions in the experiments with the physical robot were the same as those in the simulations.

Table III lists the mean of SR and CovR. As in the simulator environment, our method showed a better SR and CovR against the baselines in the real world. The robot behavior in the experiments is displayed in the supplemental video. Our navigation system can effectively work in environments where visual changes occur due to replacing furniture and passing pedestrians.

VI. CONCLUSIONS

We proposed a localization method utilizing recurrent-type graph neural networks that use the spatial information of a topological map and the temporal information from a robot’s observation images. The method was trained on simulated
images with the GT pose and on real images without the GT pose for sim2real transfers. The proposed method outperforms the baselines in localization and navigation tasks. It can accurately estimate the self-node even if the topological map contains multiple similar node images.

Future studies should improve the localization performance in a scene that considerably deviates from the topological map. For more robust navigation, the robot must precisely localize its position as it avoids obstacles.

VII. ACKNOWLEDGMENT

We thank Kazutoshi Sukigara, Kota Sato, Yuichiro Matsuda, and Yasuaki Tsurumi for measuring the three-dimensional environments utilized for collecting the training images and evaluating our method for navigation. We thank Hideki Deguchi for the evaluation of navigation, and Satoshi Koide and Keisuke Kawano for the technical discussions. We would like to thank Editage (www.editage.com) for English language editing and anonymous reviewers for their helpful comments on earlier drafts of the manuscript.

REFERENCES

[1] S. Lowry et al., “Visual place recognition: A survey,” IEEE Transactions on Robotics, vol. 32, no. 1, pp. 1–19, 2015.
[2] N. Savinov et al., “Semi-parametric topological memory for navigation,” arXiv preprint arXiv:1803.00652, 2018.
[3] 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.
[4] 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.
[5] X. Li et al., “Seannet: Semantic understanding network for localization under object dynamics,” arXiv preprint arXiv:2110.02276, 2021.
[6] K. Chen et al., “A behavioral approach to visual navigation with graph localization networks,” arXiv preprint arXiv:1903.00445, 2019.
[7] O. Kwon et al., “Visual graph memory with unsupervised representation for visual navigation,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 15 890–15 899.
[8] 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.
[9] N. Hirose et al., “Deep visual mpc-policy learning for navigation,” IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 3184–3191, 2019.
[10] ——, “Probabilistic visual navigation with bidirectional image prediction,” in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 1539–1546.
[11] E. W. Dijkstra et al., “A note on two problems in connexion with graphs,” Numerische Mathematik, vol. 1, no. 1, pp. 269–271, 1959.
[12] https://cvgl.stanford.edu/dataset/, (accessed Jan. 18, 2022).
[13] F. Xia et al., “Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 713–720, 2020.
[14] ——, “Gibson env v2: Embodied simulation environments for interactive navigation,” Stanford University, Tech. Rep., 2019.
[15] F. Chaumette and S. Hutchinson, “Visual servo control. I. basic approaches,” IEEE Robotics & Automation Magazine, vol. 13, no. 4, pp. 82–90, 2006.
[16] ——, “Visual servo control. II. advanced approaches [tutorial],” IEEE Robotics & Automation Magazine, vol. 14, no. 1, pp. 109–118, 2007.
[17] C. Durruti 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.
[18] J. Engel et al., “Lsd-slam: Large-scale direct monocular slam,” in European conference on computer vision. Springer, 2014, pp. 834–849.
[19] J. Fuentes-Pacheco et al., “Visual simultaneous localization and mapping: a survey,” Artificial Intelligence review, vol. 43, no. 1, pp. 55–81, 2015.
[20] 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.
[21] ——, “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.
[22] S. Hutchinson et al., “A tutorial on visual servo control,” IEEE transactions on robotics and automation, vol. 12, no. 5, pp. 651–670, 1996.
[23] 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.
[24] 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.
[25] D. Mishkin et al., “Benchmarking classic and learned navigation in complex 3d environments,” arXiv preprint arXiv:1901.10915, 2019.
[26] 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.
[27] E. Wijmans et al., “Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames,” arXiv preprint arXiv:1911.00357, 2019.
[28] 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.
[29] A. Kendall et al., “Posenet: A convolutional network for real-time 6 dof camera relocalization,” in ICCV, 2015, pp. 2938–2946.
[30] V. Balntas et al., “Relocnet: Continuous metric learning relocalisation using neural nets,” in ECCV, 2018, pp. 751–767.
[31] M. Ding et al., “Camnet: Coarse-to-fine retrieval for camera relocalization,” in ICCV, 2019, pp. 2871–2880.
[32] P.-E. Sarlin et al., “From coarse to fine: Robust hierarchical localisation at large scale,” in CVPR, 2019, pp. 12 716–12 725.
[33] J. Revaud et al., “R2d2: repeatable and reliable detector and descriptor,” NIPS, 2019.
[34] E. Brachmann et al., “Dsac-differentiable ransac for camera localization,” in CVPR, 2017, pp. 6684–6692.
[35] E. Brachmann and C. Rother, “Learning less is more: 6d camera localization via 3d surface regression,” 2018, pp. 4654–4662.
[36] ——, “Expert sample consensus applied to camera re-localization,” in ICCV, 2019, pp. 7525–7534.
[37] ——, “Visual camera re-localization from rgb and rgb-d images using dsac,” 2021.
[38] R. Arandjelovic et al., “Netvlad: Cnn architecture for weakly supervised place recognition,” in CVPR, 2016, pp. 5297–5307.
[39] A. Torii et al., “24/7 place recognition by view synthesis,” in CVPR, 2015, pp. 1808–1817.
[40] S. Zagoruyko and N. Komodakis, “Learning to compare image patches via convolutional neural networks,” in CVPR, 2015, pp. 4353–4361.
[41] Y. Seo et al., “Structured sequence modeling with graph convolutional recurrent networks,” in International Conference on Neural Information Processing. Springer, 2018, pp. 362–373.
[42] M. Defferrard et al., “Convolutional neural networks on graphs with fast localized spectral filtering,” Advances in neural information processing systems, vol. 29, 2016.
[43] D. Lee et al., “Large-scale localization datasets in crowded indoor spaces,” 2021.
[44] 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.
[45] 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.
[46] https://www.image-net.org/, (accessed Jan. 18, 2022).
[47] Z. Wang et al., “Mean squared error: Love it or leave it? a new look at error visibility to structural similarity,” IEEE transactions on image processing, vol. 13, no. 4, pp. 600–612, 2004.