KG-Nav: Bridging the Gap between Visual Servoing and Image-Goal Navigation via Keypoint Graph-Based Reinforcement Learning

Yunho Choi, Jaeseok Heo, Nuri Kim, and Songhwai Oh

***Abstract*—In this paper, we tackle the image-goal navigation problem where an agent navigates to a target viewpoint, given the image taken from the target viewpoint. It is important to understand the underlying geometry of the environment in order to efficiently solve the problem. Earlier approaches tend to overfit training environments with long training times. They utilize additional sensors such as a depth sensor, a pose sensor, and an object detector, which make them impractical and hardly transferable to unseen environments. Hence, we propose a keypoint graph-based reinforcement learning method inspired by classical visual servoing. Utilizing keypoint matching-based features, our method enables exact viewpoint matching with up to 25 times higher sample efficiency. Our method outperforms state-of-the-art algorithms in both perspective and panoramic RGB image-goal navigation benchmark datasets with significant margins. We also demonstrate that the proposed method is robust to environmental changes and easily transferable to real-world experiments with zero-shot sim-to-real.**

I. Introduction

With a growing body of research in embodied AI, semantic navigation tasks such as image-goal navigation, object-goal navigation, and vision-language navigation (VLN) have been extensively explored. For such tasks, a navigation agent or a robot should explore and interact with environments just as humans do, which makes the classical navigation algorithms such as simultaneous localization and mapping (SLAM) methods inefficient since they require cumbersome modifications for semantic exploration. Recent studies in learning-based navigation algorithms [1]–[7] have made notable advances in solving semantic navigation tasks, with the combination of reinforcement learning algorithms [1]–[3], [8], navigation datasets [9], and simulators [10], [11].

Especially, the performances of recent semantic navigation algorithms mostly owe to the powerful simulators [10], [11], which enable high FPS and photorealistic rendering of observations that an agent makes while navigating in the 3D environments scanned from real houses. However, the limitation of such algorithms also lies in exploiting the photorealistic simulators. In the limited number of compatible training scenes, those algorithms commonly learn their navigation policies with hundreds of millions of training frames to deal with high-dimensional input data [1], [3], [5], [6], [12]. These policies are prone to overfit and hard to generalize in unseen environments. To alleviate the sample inefficiency, they often rely on additional depth and pose observations in addition to visual observations. However, it makes their policies impractical and less robust to sensor noises, making it hard to apply the algorithms to real robotic systems. We claim the explicit use of local features that humans use when exploring unfamiliar places can enhance sample efficiency and solve the overfitting problem.

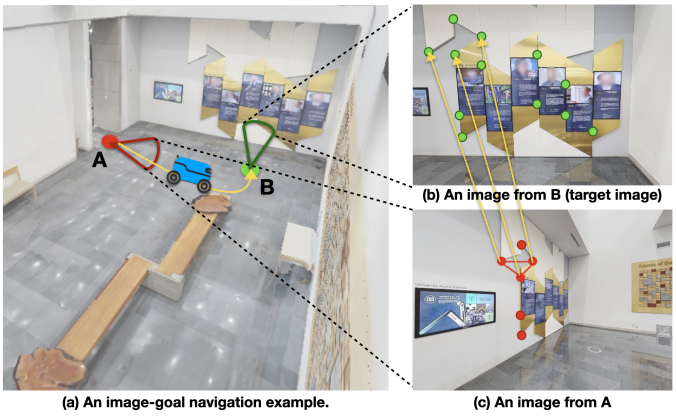


Fig. 1. (a) An image-goal navigation example. Given an initial observation (c) and a target image (b), our agent is navigating to the viewpoint (the green point) of the target image by comparing two images, (b) and (c).

Imagine an image-goal navigation case, which is our target problem. Suppose you have a photo of a bed, and you have to explore the place you are in and navigate to the bed. If you are in a familiar house, you will easily navigate to a bedroom since a bedroom likely has a bed. However, that prior information cannot be utilized when you are in an unfamiliar place, say a furniture store. Then, you would rather compare the photo of the bed with the pieces of furniture in the store while walking around. You might also take advantage of the noticeable details of the bed to precisely move to the target bed among similar beds.

Inspired by how a human navigates in an unfamiliar place, we aim to build a navigation method that compares target and current observations, finds similar details in two images, and infers the relative spatial relationship from the comparison. Displacement of the parts with similar local appearances implies which action to take to match the viewpoint. We have an example in Figure 1. In an initial observation at the red point (A), three points are found to be similar with the target

Y. Choi, J. Heo, N. Kim, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea. (e-mail: {yunho.choi, jaeseok.heo, nuri.kim}@rllab.snu.ac.kr, songhwai@snu.ac.kr).

image. Since the displacements of the three pairs of similar points depicted as yellow arrows diverge to the left, we can easily guess that the robot should move forward to the right. A spatial relationship expressed with such comparison will work as an inductive bias, making our methods sample efficient and able to capture the underlying geometry of unfamiliar environments.

To this end, we propose a novel image-goal navigation method, termed *Keypoint Graph Navigation (KG-Nav)*, which predicts a navigation action from an image feature comparison between the target and current observations represented as *keypoint graphs*. We utilize the location of local interest points, or *keypoints*, and match between two keypoint sets which are extracted from the target and current observations. With the matched keypoints, we construct keypoint graphs defined with node features inspired by the classical image-based visual servoing [13]. To predict the most rewarding action, we then apply graph attention (GAT) layers to the keypoint graphs which are trained with reinforcement learning.

Inheriting the strength of image-based visual servoing methods with the keypoint graph and node features, our method can navigate the agent to exactly match the target viewpoint while other learning-based algorithms fail and just stop near the target viewpoint since they do not explicitly utilize local features. The use of keypoint graphs also enables a panoramic camera setting as well as a perspective camera setting. KG-Nav also makes up for the weaknesses of the visual servoing methods. Unlike image-based visual servoing methods that require an initial viewpoint with a considerable overlap to a target viewpoint, KG-Nav can also navigate from a larger translation and rotation without any field-of-view (FoV) overlap since it fuses the prediction from a global feature for the final prediction. We use a feature extractor that jointly computes global and local features from one neural network [14], [15] which makes it computationally efficient, though our method works with any off-the-shelf feature extractor. The proposed method works solely with RGB observations, while the performance of visual servoing methods heavily depends on a depth sensor or depth estimation.

In our simulation experiments with the Gibson dataset [11], we validate that KG-Nav successfully navigates to a target viewpoint even when a success is defined in a narrower region in terms of distance and angle from the target viewpoint. When compared to other image-goal navigation methods, our methods enjoy the state-of-the-art performance for both perspective camera and panoramic camera setups. We also show that KG-Nav is robust to environmental changes by evaluating navigation performance when new objects are gradually introduced in the environment. In real robot experiments, we validate the generalization performance of KG-Nav by showing the successful zero-shot transfer from simulation to the real world.

In summary, our contributions are threefold:

- We propose a keypoint graph-based image-goal navigation policy inspired by image-based visual servoing, which enables exact viewpoint matching with up to 25 times higher sample efficiency.
- Our method, KG-Nav, shows state-of-the-art performance in image-goal navigation with both perspective and panoramic RGB camera settings with significant margins.
- We demonstrate that KG-Nav is robust to environmental changes and easily transferable to a real-world robot with zero-shot sim-to-real.

II. Related Work

Feature Extractors. There have been many attempts for the robot navigation task to utilize image features that are pretrained for the visual recognition task. ResNet [16] feature trained for the ImageNet dataset [17] is a typical choice for many work [4], [6], [18]. Mid-level visual representations [19] were also explored to solve downstream robot navigation tasks [20]. Global image features [21], [22] and local features [23]–[26] are widely used for the visual localization task, which is a task closely related to robot navigation tasks. For efficient feature extraction, models that jointly compute global and local features are proposed [14], [15], [22]. [14] distills pre-trained local [23] and global [21] features into a single model called HF-Net for hierarchical localization. [15] trains a model end-to-end to jointly extract global and local features for image retrieval. KG-Nav exploits this type of model, and to our knowledge, our method is the first work that utilizes both the global and local features for end-to-end navigation policy learning.

Learning-Based Visual Servoing. Classical visual servoing methods [13] are sensitive to camera calibration error and depth error or prone to stuck at local minima. Most of all, the accuracy of feature matches has a huge impact on the servoing performance. Photometric visual servoing method [**?**] obviates the need to extract and match features but results in a small convergence basin. To overcome the limitation of classical visual servoing methods [13], many learning-based visual servoing frameworks have been proposed. Commonly they utilize an optical flow estimation network [27] and a depth map or a depth estimation. [28] constitutes the interaction matrix and follows the classical visual serving, and [29] directly learns actions from the optical flow with RL. DIViS [30] additionally uses free space estimation and object localization. [31], [32] proposes model predictive control (MPC) frameworks to reduce the flow error. These learning-based visual servoing methods rely on the flow estimation network, making them hard to be applied to problems with a large translation and rotation.

Image-Goal Navigation In an image-goal navigation problem, an agent must navigate to a target viewpoint, given a target image taken from the target viewpoint. Image-goal navigation algorithms utilize various kinds of memories to store semantic information and solve long-range navigation. [6] retains a set of previous state embeddings and learns a policy that attends over that set with RL. Topological memory is a common choice for recent studies [2], [4], [7], [33], [34]. They build topological memory based on the similarity of observations,

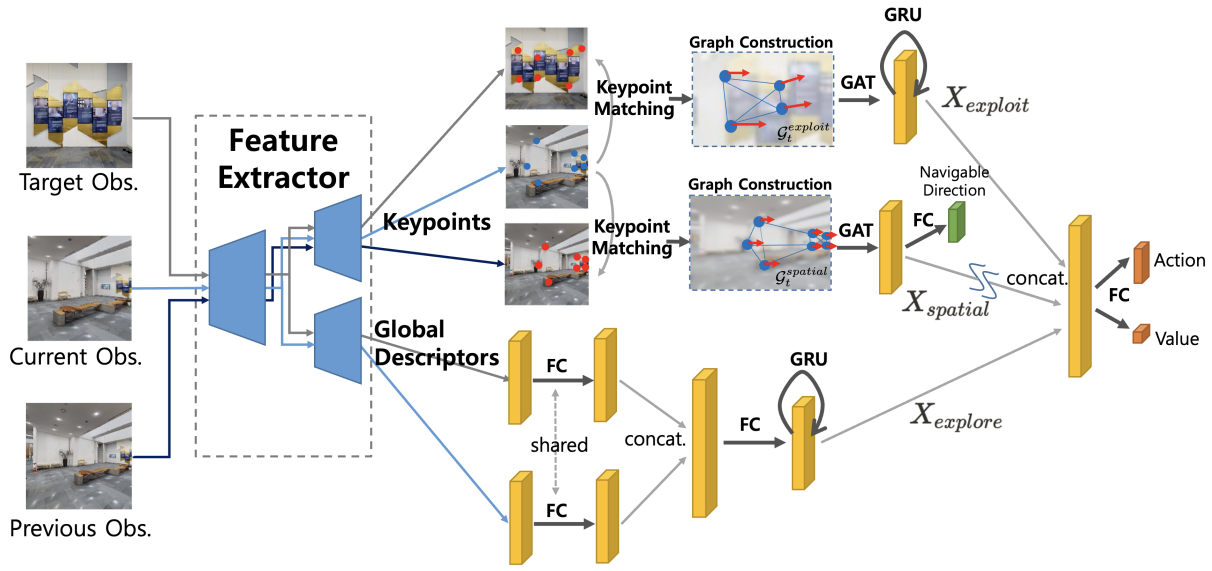


Fig. 2. The network architecture of the proposed agent KG-Nav for image-goal navigation. FC, Concat, and GRU denote fully connected layer, concatenation, and gated recurrent unit, respectively.

where the node feature is the visual feature at that location and the connectivity is defined with navigability. [4], [7] learn a modularized policy where each module is trained with supervised learning. [33] builds a topological map with pre-exploration and learns a policy to navigate to a subgoal node. [2] encodes a topological memory with a graph neural network and learns a policy from the encoding and a current observation with RL. [34] additionally uses object detection information for the node feature. [3], [12] do not keep an explicit memory, but learn a visual representation with self-supervised learning and have a recurrent state in their policy structure. Our algorithm falls into this category, except that we use visual representations defined with visual servoing-inspired node features. The aforementioned previous methods require 100M to 500M training data which are costly even for simulation. In addition to the sample efficiency issue, they commonly rely on depth and pose sensors to build a map or topological memory, which harms their utility. Usually, they are trained to execute a *stop* action within $1\,m$ of the target viewpoint. Succeeding by just reaching near the target viewpoint without aligning with the viewpoint can provide a wrong learning signal which makes this methods hard to match the viewpoint exactly. [35] is a complementary method for other image-goal navigation, which can be utilized after discovering the target point. It computes a relative pose to the goal with keypoint matching and classical RANSAC methods.

III. Problem Formulation

In this paper, we focus on solving the image-goal navigation problem. In an episode of the image-goal navigation task, an agent is placed in an unknown environment and provided with an RGB observation I_G taken from the target viewpoint. At each time t, the agent observes an RGB image I_t and takes action a_t to reach the target viewpoint. The agent observes only RGB sensor data and no other data, such as depth, pose and a map, is accessible. The RGB sensor can be either a perspective camera or a panoramic camera, since our method is compatible for both settings. An example is shown in Figure 1. The followings are details of our problem formulation.

Observation Space. I_t and I_G are assumed to be RGB images captured by a camera with known camera parameters. Each RGB image has a size of 256×256 and an FoV of $90\,^\circ$ for the perspective camera setting. For a panoramic camera, the image size is 128×512 which is a concatenation of four 128×128 RGB images captured at four predefined angles $(0\,^\circ, 90\,^\circ, 180\,^\circ, 270\,^\circ)$.

Action Space. The agent has a discrete action space. At each time step, the agent must select one of five actions: `MOVE_FORWARD` by $0.25\,m$, `TURN_LEFT` by $10\,^\circ$, `TURN_RIGHT` by $10\,^\circ$, `MOVE_BACKWARD` by $0.25\,m$, and `STOP`. `MOVE_BACKWARD` is added for collision avoidance to the standard action set, following [36]. An actuation noise model from [37] is used for realistic action modeling.

Success Condition. In the default setting, an episode is considered successful if the agent takes the `STOP` action only when the agent is within $1\,m$ of the target viewpoint, which is the standard setup in the literature. In the *tighter* success condition, the success region is defined within $0.5\,m$ and $20\,^\circ$ from the target viewpoint. Issuing the `STOP` outside the success region or exceeding the maximum number of steps terminates an episode as a failure.

IV. Method

In this section, we demonstrate how our keypoint graph navigation (KG-Nav) agent predicts actions and how it is trained. The network structure of the KG-Nav agent is shown in Figure 2. It consists of three modules that compute the *exploitation* feature, *exploration* feature, and *spatial* feature.

We first present how visual descriptors and keypoints are extracted from observations (Section IV-A), how each module computes its features (Section IV-B), how an agent makes the final prediction (Section IV-C), and then how to learn this network end-to-end (Section IV-D).

A. Visual Descriptor and Keypoint Extraction

We make use of a low-level feature extractor to extract global and local descriptors from both I_t and I_G. We can use any off-the-shelf extractors for global descriptor [21], [22] and local descriptors [23], [24], [26], [38]. Nevertheless, since we train the policy network in a sequential manner with reinforcement learning (RL), timing is crucial, and consequently, we use joint global and local descriptor extractors [14], [15] which reduces computational burden by extracting both descriptors in a single network. We choose HF-Net [14] for our experiments due to its low complexity and features that make it well-suited for navigation purposes. For a global descriptor, HF-Net outputs a NetVLAD [21] descriptor which is widely used for visual localization. For a local descriptor, HF-Net jointly outputs SuperPoint [23] keypoint detections and keypoint descriptors which work well in unseen environments since they are trained to discriminate local geometries.

Specifically, the feature extractor outputs a global descriptor and detects M keypoints from the current observation I_t and target observation I_G, respectively. The global descriptors from two images are denoted by Y_t and Y_G, respectively. The keypoint sets detected from two images are denoted by $\{(u_t^i, v_t^i)\}$ and $\{(u_G^i, v_G^i)\}$ respectively, where u and v are the normalized x-coordinate and y-coordinate of each keypoint and i is the index of a keypoint, i.e., $i = 1, \cdots, M$. We call two keypoint sets current keypoints and target keypoints, respectively, for convenience. To find the keypoints related to the target observation, we match the keypoints between current keypoints and target keypoints, and denote the set of matched keypoints in the current keypoints by A_t. Additionally, our method matches the keypoints between two consecutive frames, i.e., $\{(u_t^i, v_t^i)\}$ and $\{(u_{t-1}^i, v_{t-1}^i)\}$, in order to obtain flow information. Here, we denote the set of matched keypoints in the current keypoints by B_t. For simplicity, we use a simple nearest-neighbor algorithm to find matches.

B. Feature Computation

We take a modular approach to compute the exploitation feature, exploration feature, and spatial feature as seen in Figure 2. The exploitation feature helps exact viewpoint matching near the destination, the exploration feature enables the agent to explore the environment in the case of no FoV overlapping, and the spatial feature helps the agent perceive the spatial layout and avoid collision. We demonstrate the computation process of each module for the proposed features along with how we fuse the resulting features in the subsequent sections.

1) Exploitation Feature: With the keypoints extracted in Section IV-A, we aim to construct a keypoint graph for the exploitation of local visual cues at time t which is denoted by $\mathcal{G}_t^{exploit}$ to compute the exploitation feature $X_t^{exploit}$. To count meaningful keypoints in I_t for viewpoint matching, we find the intersection between the two matched keypoint sets A_t and B_t. The keypoints in this intersection, i.e., $\{(u_t^i, v_t^i) \mid i \in A_t \cap B_t\}$ are the nodes that constitute the node set denoted by $\mathcal{V}_t^{exploit}$ of the current keypoint graph for exploitation $\mathcal{G}_t^{exploit}$.

To efficiently learn robot actions from a keypoint graph, our method constructs a unique node feature set for the j-th node in $\mathcal{V}_t^{exploit} = \{(u_t^j, v_t^j)\}$, which is inspired by the interaction matrix that relates a camera motion to a motion of feature points in the classical image-based visual servoing [13]. Considering the fact that the interaction matrix is the function of the feature point coordinate and the depth of the feature point, we count (u_t^j, v_t^j) and the magnitude of the optic flow at (u_t^j, v_t^j) as a proxy for depth information following [28]. The optic flow at (u_t^j, v_t^j) is denoted by $f_t^j = (\Delta u_t^j, \Delta v_t^j) = (u_t^j - u_{t-1}^j, v_t^j - v_{t-1}^j)$, where (u_{t-1}^j, v_{t-1}^j) is the keypoint in I_{t-1} matched with (u_t^j, v_t^j). Naturally, we include a relative motion to the matched target keypoints $(u_G^j - u_t^j, v_G^j - v_t^j)$, where (u_G^j, v_G^j) is the keypoint in I_G matched with (u_t^j, v_t^j). We also include a keypoint detection score q_t^j and the number of matches with target keypoints $|A_t|$ to consider the reliability of the keypoint detection and progress to the target viewpoint, respectively. To sum it up, we define the node feature $x_t^{exploit,j}$ for the j-th node in $\mathcal{V}_t^{exploit}$ as follows:

$$x_t^{exploit,j} = \left\{u_t^j, v_t^j, u_G^j - u_t^j, v_G^j - v_t^j, f_t^j, \|f_t^j\|, q_t^j, |A_t|\right\}.$$

We refer to this nine-dimensional feature set as *the exploitation node feature*. Concatenating the exploitation node features, we denote the resulting node feature matrix by $Y_t^{exploit} \in \mathbb{R}^{N \times 9}$, where N is the number of keypoints in the keypoint graph $\mathcal{G}_t^{exploit}$.

We build edges for a node in $\mathcal{V}_t^{exploit}$ by connecting to the nodes with the k-nearest value of flow magnitude $\|f_t^j\|$, to emulate the k-nearest neighbor keypoints in the 3D space. With these edges denoted by $\mathcal{E}_t^{exploit}$, we now construct a keypoint graph at time t, i.e., $\mathcal{G}_t^{exploit} = (\mathcal{V}_t^{exploit}, \mathcal{E}_t^{exploit})$ with node feature matrix $Y_t^{exploit}$.

In the case of the panoramic camera setting, the distance between two image coordinates is calculated in consideration of the cylindrical coordinate system, and the neighbor and node feature of a keypoint is calculated accordingly.

From the keypoint graph $\mathcal{G}_t^{exploit}$ and node feature matrix $Y_t^{exploit}$, we extract the information for the action prediction. We use a graph neural network similar to Vision GNN [39] to reduce the over-smoothing effect as follows:

$$X_t' = \sigma(Y_t^{exploit} W_0^{exploit} + b_0^{exploit})$$

$$X_t'' = \sigma\big(\texttt{GAT}(X_t', \mathcal{E}_t)\big) + X_t'$$

$$X_t''' = \texttt{Global_Max_Pool}\Big(\sigma\big(\texttt{GAT}(X_t'', \mathcal{E}_t)\big) + X_t''\Big)$$

$$X_t^{exploit} = \texttt{GRU}(X_t''').$$

Here, σ denotes a $tanh$ activation function, and W and b indicate a weight matrix and bias vector of a linear layer, respectively. Unlike Vision GNN, we use a multi-headed graph attention (GAT) layer [40] as a building block. Our method computes the final exploitation feature vector $X_t^{exploit}$ from the keypoint graph $\mathcal{G}_t^{exploit}$ through a linear layer, two GAT layers, and a gated recurrent unit (GRU) [41].

2) Exploration Feature: On the other hand, we also need information from the global descriptor from the feature extractor, in order to explore unseen regions and deal with low FoV situations. With Y_t and Y_G denoting the global descriptors extracted from I_t and I_G respectively, we compute the final exploration feature vector $X_t^{explore}$ through two linear layers and a GRU.

$$X'_t = \sigma(Y_t W_1^{explore} + b_1^{explore})$$
$$X'_G = \sigma(Y_G W_1^{explore} + b_1^{explore})$$
$$X_t^{explore} = \texttt{GRU}\Big(\sigma\big([X'_t || X'_G] W_2^{explore} + b_2^{explore}\big)\Big)$$

Here, '||' denotes a concatenation, and the weight of the first linear layer is shared as stated.

3) Spatial Feature: With the keypoints extracted in Section IV-A, we construct a keypoint graph for spatial perception at time t which is denoted by $\mathcal{G}_t^{spatial}$. To get the spatial information and ego-motion information from the optic flows in I_t, we use the keypoints in B_t to constitute the node set denoted by $\mathcal{V}_t^{spatial}$ of the current keypoint graph $\mathcal{G}_t^{spatial}$.

Our method constructs a node feature set for the j-th node in $\mathcal{V}_t^{spatial} = \{(u_t^j, v_t^j)\}$. We count (u_t^j, v_t^j) and the magnitude of the flow as a proxy for depth information following [28]. The optic flow is denoted by $f_t^j = (\Delta u_t^j, \Delta v_t^j) = (u_t^j - u_{t-1}^j, v_t^j - v_{t-1}^j)$, where (u_{t-1}^j, v_{t-1}^j) is the keypoint in I_{t-1} matched with (u_t^j, v_t^j). We also include the previous robot action a_{t-1}. To sum it up, we define the node feature $x_t^{spatial,j}$ for the j-th node in $\mathcal{V}_t^{spatial}$ as follows:

$$x_t^{spatial,j} = \big\{u_t^j, v_t^j, f_t, \|f_t^j\|, a_{t-1}\big\}.$$
$$f_t^j = (\Delta u_t^j, \Delta v_t^j) = (u_t^j - u_{t-1}^j, v_t^j - v_{t-1}^j)$$

Concatenating these features, we denote the resulting node feature matrix by $Y_t^{spatial} \in \mathbb{R}^{L \times 6}$, where L is the number of keypoints in the keypoint graph $\mathcal{G}_t^{spatial}$.

We build edges for a node in $\mathcal{V}_t^{spatial}$ by connecting to the nodes with the k-nearest value of flow magnitude $\|f_t^j\|$, to emulate the k-nearest neighbor keypoints in the 3D space. With these edges denoted by $\mathcal{E}_t^{spatial}$, we now construct a keypoint graph at time t, i.e., $\mathcal{G}_t^{spatial} = (\mathcal{V}_t^{spatial}, \mathcal{E}_t^{spatial})$ with node feature matrix $Y_t^{spatial}$.

From the keypoint graph $\mathcal{G}_t^{spatial}$ and node feature matrix $Y_t^{spatial}$, we summarize the information for spatial perception. To that end, we use a graph neural network similar to Vision GNN [39] to reduce the over-smoothing effect as follows:

$$X'_t = \sigma(Y_t^{spatial} W_0^{spatial} + b_0^{spatial})$$
$$X''_t = \sigma\big(\texttt{GAT}(X'_t, \mathcal{E}_t)\big) + X'_t$$
$$X_t^{spatial} = \texttt{Global_Max_Pool}\Big(\sigma\big(\texttt{GAT}(X''_t, \mathcal{E}_t)\big) + X''_t\Big)$$

Here, σ denotes a $tanh$ activation function, and W and b indicate a weight matrix and bias vector of a linear layer, respectively. We use a multi-headed graph attention (GAT) layer [40] as a building block. We compute the final spatial feature vector $X_t^{spatial}$ from the keypoint graph $\mathcal{G}_t^{spatial}$ through a linear layer and two GAT layers with $tanh$ activation functions.

C. Fusing Features from All Modules

Finally, our method aggregates the feature vectors from every module by concatenating them into a single vector, i.e., $X = \big[X_t^{explore} || X_t^{exploit} || X_t^{spatial}\big]$, and predict the action probabilities and value of the current state by applying a linear layer to X. Note that $X_t^{spatial}$ is trained with an auxiliary task of predicting navigable directions in order to learn features solely related to spatial perception, while $X_t^{explore}$ and $X_t^{exploit}$ are trained with reinforcement learning that incentivizes navigation toward the target viewpoint. The above steps are depicted in Figure 2.

D. Learning to Navigate

KG-Nav learns the parameters of its layers in Section IV-B with reinforcement learning, with the feature extractor frozen. A common reward function from [10] consists of a success reward $r_{success}$, a change in the geodesic distance to the goal Δ_{dtg}, and a constant slack penalty r_{slack} that encourages short trajectories. In addition, our reward function is composed of additional terms such as a collision penalty, a proximity penalty, a backward action penalty, a keypoint detection reward, and a change in the keypoint distance to the goal reward. We elaborate on each reward term in the following.

- Collision penalty r_{col}: 0.1 if the agent collides with any obstacle and zero otherwise.
- Proximity penalty r_{prox}: 0.05 if the agent is closer than $0.15\,m$ to any obstacle and zero otherwise.
- Backward action penalty r_{back}: 0.05 if the agent executes `MOVE_BACKWARD` and zero otherwise. The backward action penalty prevents the frequent use of `MOVE_BACKWARD` which worsens the navigation performance.
- Keypoint detection reward $r_{keypoint_detect}$: It rewards 0.1 per each target keypoints which is newly matched with the current keypoints observation.
- Change in the keypoint distance to the goal reward $\Delta_{keypoint_dtg}$: It rewards change in the average keypoint distances to the matched keypoints in the normalized pixel coordinate, i.e., $keypoint_dtg = \|(u_G^j - u_t^j, v_G^j - v_t^j)\|$. It is an analogy of Δ_{dtg} in the keypoint coordinate.

The newly proposed keypoint detection reward and keypoint distance to the goal reward facilitate exploration and exploitation respectively, exploiting the keypoint matches to the target

keypoints. To sum up, our reward function r for training the agent is defined as below.

$$r(s_t = I_t, a_t) = r_{success} + \Delta_{dtg} - r_{slack} \\ - r_{col} - r_{prox} - r_{back} + r_{keypoint_detect} + \Delta_{keypoint_dtg},$$

where $r_{success}$ is 10 if the agent executes the `STOP` action while satisfying a success condition and zero otherwise, and r_{slack} is set to 0.01.

We train our network with the *tighter* success condition defined in Section III with oracle `STOP`, i.e., once the agent gets into the success region, the episode is considered a success. Since the agent solely focuses on exactly matching the target viewpoint, we consequently eliminate the need for learning the `STOP` action and improve sample efficiency significantly, which is validated in experiments. Instead, the KG-Nav agent can decide whether to `STOP` or not in an evaluation phase by setting a threshold on the number of matches, which is shown to be more effective than learning the `STOP` action in experiments. Exceptionally, we train the panoramic camera agent with a less tight success condition of $0.75\,m$ and $30\,^\circ$ from the target viewpoint. This is because the low resolution of the panoramic camera makes it hard to distinguish the images taken from close locations.

We train our KG-Nav network to maximize the sum of rewards with proximal policy optimization (PPO) [8]. We also train for an auxiliary task of navigability prediction to learn the spatial feature $X_t^{spatial}$ and avoid collisions with obstacles. As seen in Figure 2, KG-Nav predicts navigable directions by applying a linear layer to $X_t^{spatial}$. The predictions are six binary variables that are supervised by navigability ahead $0.5\,m$ and $1.0\,m$ with headings of $\{-30\,^\circ, 0\,^\circ, 30\,^\circ\}$. Consequently, a loss function we optimize is a sum of two losses, i.e., $\mathcal{L} = \mathcal{L}_{ppo} + \mathcal{L}_{aux}$, where $\mathcal{L}_{ppo}$ is a PPO loss and $\mathcal{L}_{aux}$ is a sum of six binary cross-entropy losses for navigability prediction.

V. Experiments

To demonstrate the effectiveness of our image-goal navigation method, we conducted simulation experiments on the Habitat simulator [10] with the Gibson scenes [11] which contain photorealistic indoor scenes. We compare KG-Nav with the state-of-the-art algorithms and show that our method is capable of exact viewpoint matching and robust to environmental changes. We also validate the transferability of our approach with real robot experiments. Following are the implementation details of our simulation experiments.

A. Implementation Details.

Image-Goal Navigation Dataset. For training the perspective camera agent, we want our image goals to contain target objects, not just empty walls. Thus, we follow the work [42] to generate training episodes for the Gibson scenes with image goals containing objects. As a result, we get 500 image goals for each of the 16 Gibson scenes, and we generate training episodes by sampling an initial position $1.5{\sim}10\,m$ away from the goal points. We used the image-goal navigation test episodes from [4] for the evaluation of the perspective camera agent. Note that this test set does not include an overlapping scene with our training data. For training and evaluation of the panoramic camera agent, we used training and evaluation splits from the image-goal navigation dataset in [6].

Training Setup. We choose HF-Net [14] for the feature extractor, and the numbers of keypoints we extract are 100 and 200 for the perspective camera and panoramic camera settings, respectively. For matching keypoints, we used a simple brute-force nearest-neighbor algorithm with a maximum ratio test which provides fast matching due to the small number of keypoints. When constructing keypoint graphs, the number of neighbors k for KNN is 3 for both agents. We used four heads for the multi-head GAT layers. The maximum number of time steps in an episode is set to 500. When training, the success condition of an episode is defined as the tight success condition with oracle `STOP` in Section IV-D. The keypoint reward terms $r_{keypoint_detect}$ and $\Delta_{keypoint_dtg}$ are scaled to half for training the perspective camera agent. We trained our agents with PPO algorithm [8] implemented in the `habitat-lab` repository [10]. The hyperparameter set used for training KG-Nav is provided in the Table VII at the end of the paper. We trained the KG-Nav agent for 20 million frames for the perspective camera agent and 25 million frames for the panoramic camera agent.

Evaluation Setup. In the evaluation phase, the KG-Nav agent takes `STOP` action if the ratio of current keypoints which are matched to target keypoints, denoted by ρ_{th}, exceeds a predefined threshold. Default threshold values are set to 0.55 and 0.5 for the perspective camera and the panoramic camera, respectively. The default success condition is stopping within $1\,m$ of the target viewpoint, for comparison with state-of-the-art methods. For evaluation metrics, we use success rate (SR) and success weighted by inverse path length (SPL) [9]. SPL weights each success by how efficiently the agent navigated to the goal, in terms of a path length compared to the shortest possible path length. Episodes are divided into three categories: Easy Medium, and Hard which have paths with length of $1.5 \sim 3\,m$, $3 \sim 5\,m$, and $5 \sim 10\,m$, respectively. Test episodes from [4] are further divided into straight and curved paths.

B. Baseline Methods

We compare KG-Nav with three groups of baseline methods depending on the camera setting and success criteria. The followings are the baseline methods for each group.

Perspective Camera Setting - Visual Servoing Baselines. We compare the KG-Nav agent using the perspective camera with visual servoing baselines. For this comparison, learning-based baseline methods are trained with the *tighter* success condition ($0.5\,m$, $20\,^\circ$). Evaluation metrics are based on the same success condition with the oracle `STOP` to assess how exactly the agents navigate to match the target viewpoints.

- **Image-Based Visual Servoing (IBVS)** [13]: A classical image-based visual servoing method.

- **Domain Invariant Visual Servoing (DIViS)** [30]: A learning-based visual servoing method which consists of a free-space prediction network and an object localization network. We replace the free-space prediction network with the ground-truth using the depth map. Accordingly, we denote the method by **DIViS-oracle**.
- **Target-Driven RL** [18]: A baseline method that ablates the local feature parts from our KG-Nav.
- **KeypointRL** [43]: A baseline method that similarly fuses the global and local features but process all keypoint features with the same linear layer without graph structure.

Perspective Camera Setting - Image-Goal Navigation Baselines. We compare the KG-Nav agent using the perspective camera with perspective camera-based image-goal navigation methods which are evaluated with the standard success condition ($1.0\,m$).

- **Behavior Cloning (BC) with Spatial Memory**: This method uses ResNet-18 [16] features and a metric map for the input of a behavior cloning policy.
- **Behavior Cloning (BC) with GRU**: Another behavior cloning method uses a GRU [41] instead of a metric map.
- **Decentralized Distributed Proximal Policy Optimization (DDPPO)** [1]: A method trained with a distributed RL algorithm.
- **No RL No Simulation (NRNS)** [4]: This method builds a topological memory and selects the most probable unexplored node, where all submodules are trained with supervised learning on internet images. We report the result of the above methods from [4] and its official code.
- **OVRL** [3]: This method pretrains a self-supervised representation to improve a visual navigation policy.
- **Zero Experience Required (ZER)** [5]: An improved RL policy trained with view reward and view augmentation.
- **Switchable Last-Mile Image-Goal Navi. (SLING)** [35]: A method for navigation after the goal discovery, which utilizes the classical vision method to compute the relative pose.

Panoramic Camera Setting - Image-Goal Navigation Baselines. We compare the KG-Nav agent using the panoramic camera with panoramic camera-based image-goal navigation methods which are evaluated with the standard success condition ($1.0\,m$).

- **ResNet + GRU + Imitation Learning (IL)**: A baseline that extracts ResNet-18 [16] feature and computes a GRU-based policy, trained with imitation learning. We report the performance from [6].
- **Target-Driven RL** [18]: A baseline with the same structure as ResNet+GRU+IL but trained with reinforcement learning (RL). We report the performance from [6].
- **Active Neural SLAM (ANS)** [37]: An exploration policy based on the metric memory, adapted for image-goal navigation.
- **Neural Topological SLAM (NTS)** [7]: A neural SLAM method building a topological memory for long-term planning. There are two versions of this method: **NTS** for RGB and pose sensors input and **NTS-D** for additional usage of a depth sensor.
- **Visual Graph Memory (VGM)** [2]: This method builds a topological memory based on unsupervised representation and decode the memory with attention to predict actions.
- **Semi-Parametric Topological Memory (SPTM)** [33]: A topological memory-based method that pre-explores an environment and follows a Dijkstra path.
- **Memory-Augmented RL** [6]: This method keeps a sparse set of previous state embeddings, and predict actions with attention on the memory.

C. Exact Viewpoint Matching Capability of KG-Nav

We conducted two experiments to show that our method, inspired by the classical visual servoing methods, is capable of exact viewpoint matching while other algorithms do not. First, we show that KG-Nav outperforms previous visual servoing methods. KG-Nav and the visual servoing baselines are evaluated in the image-goal navigation episodes with the *tighter* success condition, and the result is shown in Table I. The average proportion of pixels in the initial view overlapped with the target view is computed and presented in the third column for each path category. As seen in Table I, the classical visual servoing method (IBVS) and the learning-based visual servoing method (DIViS-oracle) perform poorly due to the large translation and rotation from targets. On the other hand, KG-Nav performs the best with large margins, and comparison with Target-Driven RL and KeypointRL confirms the effectiveness of our proposed features and keypoint graph-based learning.

We measured how much the navigation performance of the image-goal navigation baselines degrades when tightening the success condition in Table II. As seen in the table, the performance degradation of KG-Nav does not exceed half of the original performance, while other image-goal navigation algorithms perform poorly in the *tighter* success condition. From the two experiments, we conclude that KG-Nav specializes in exactly matching a viewpoint when navigating toward an image goal thanks to the proposed exploitation feature.

D. Image-Goal Navigation Experiments

We compare the performance of KG-Nav with the state-of-the-art image-goal navigation methods. Results in the perspective camera setting benchmarking dataset [4] are shown in Table III. The other results in the panoramic camera setting benchmarking dataset [6] are presented in Table IV, where variants of KG-Nav are highlighted. KG-Nav-Equirect is the model trained with the equirectangular image instead of stitched panoramic images common in the literature. Algorithms with panoramic image inputs usually utilize episodic memories for long-range navigation, and the types of memory and additional sensors they use are stated in Table IV. Here, 'implicit' denotes the recurrent neural networks' recurrent state as implicit memory. In both settings, variants of KG-Nav outperform the state-of-the-art methods especially in the

TABLE I
COMPARISON OF KG-NAV WITH VISUAL SERVOING BASELINES ON STRAIGHT/CURVED EPISODES IN GIBSON SCENES

Path Type	Difficulty	Overlap	IBVS [13]		DIViS-oracle [30]		Targ.-Driven RL [18]		KeypointRL [43]		KG-Nav	
			SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL
Straight	Easy	0.061	13.9	9.3	35.7	28.3	41.9	31.4	75.9	64.9	**80.4**	**71.1**
	Medium	0.019	2.6	0.9	21.9	15.9	17.7	11.2	62.3	47.3	**65.3**	**51.3**
	Hard	0.009	0.9	1.5	10.3	6.5	5.6	2.9	37.2	22.0	**42.4**	**26.3**
Curved	Easy	0.019	1.9	0.6	13.0	4.5	21.6	9.6	52.2	24.9	**62.2**	**32.4**
	Medium	0.006	0.6	0.3	9.5	4.0	13.5	6.2	44.0	18.7	**48.0**	**25.9**
	Hard	0.002	0.1	0.1	6.2	3.1	3.9	1.7	24.6	9.7	**27.3**	**13.4**

TABLE II
CHANGES IN THE PERFORMANCE OF THE IMAGE-GOAL NAVIGATION METHODS ON STRAIGHT/CURVED EPISODES IN GIBSON SCENES WHEN THE SUCCESS CONDITION BECOMES TIGHTER

Type	Difficulty	NRNS [4]		OVRL [3]		KG-Nav	
		SR	SPL	SR	SPL	SR	SPL
Straight	Easy	70.8 → 3.6	62.4 → 3.1	52.9 → 7.5	28.8 → 4.7	**86.5 → 55.7**	**75.2 → 48.3**
	Medium	47.0 → 2.9	39.1 → 2.5	48.4 → 6.3	30.6 → 4.4	**71.2 → 48.0**	**56.5 → 37.2**
	Hard	24.4 → 1.9	17.1 → 1.6	34.0 → 3.6	21.2 → 2.4	**44.9 → 28.2**	**28.0 → 18.1**
Curved	Easy	32.7 → 1.4	10.2 → 0.3	51.0 → 4.1	26.3 → 2.5	**70.3 → 38.8**	**37.1 → 21.0**
	Medium	26.4 → 0.9	9.8 → 0.2	48.1 → 5.2	28.2 → 3.2	**58.4 → 33.7**	**31.0 → 18.3**
	Hard	10.5 → 0.6	4.4 → 0.3	33.0 → 4.6	**19.7** → 2.9	**33.5 → 17.9**	15.8 → **8.4**

TABLE III
COMPARISON OF KG-NAV WITH IMAGE-GOAL NAVIGATION BASELINES WITH PERSPECTIVE IMAGE INPUTS ON STRAIGHT/CURVED EPISODES ON GIBSON SCENES. METHODS WRITTEN IN RED UTILIZE DEPTH AND POSE SENSORS.

Path Type	Method	Easy		Medium		Hard		Overall	
		SR	SPL	SR	SPL	SR	SPL	SR	SPL
Straight	BC w/ Spatial Memory	24.8	23.9	11.5	11.2	1.3	1.2	12.5	12.1
	BC w/ GRU State	34.9	33.4	17.6	17.0	6.0	5.9	19.5	18.7
	DDPPO [1]	43.2	38.5	36.4	34.9	7.4	7.2	29.0	26.9
	NRNS [4]	70.8	62.4	47.0	39.1	24.4	17.1	47.4	39.5
	OVRL [3]	53.6	34.7	48.6	33.3	32.5	21.9	44.9	30.0
	KG-Nav	**86.5**	**75.2**	**71.2**	**56.5**	**44.9**	**28.0**	**67.5**	**53.2**
Curved	BC w/ Spatial Memory	3.1	2.5	0.8	0.7	0.2	0.1	1.3	1.1
	BC w/ GRU State	3.6	2.8	1.1	0.9	0.5	0.3	1.7	1.3
	DDPPO [1]	22.2	16.5	20.7	18.5	4.2	3.7	15.7	12.9
	NRNS [4]	32.7	10.2	26.4	9.8	10.5	4.4	23.2	8.1
	ZER [5]	48.0	34.2	36.0	25.9	15.1	10.8	33.0	23.6
	OVRL [3]	53.6	31.8	47.6	30.2	**35.6**	**22.0**	45.6	**28.0**
	KG-Nav	**70.3**	**37.1**	**58.4**	**31.0**	33.5	15.8	**54.1**	**28.0**

straight-path episodes where viewpoint matching ability of KG-Nav can be effective. Note that KG-Nav variants do not make use of any additional sensors, such as a depth and pose sensor. In the panoramic camera setting, KG-Nav-Equirect improves slightly over KG-Nav since there is no distortion resulting from stitching four images.

To understand the impact of each component, an ablation study was conducted. We trained and evaluated the image-goal navigation agents under various ablations in the perspective camera setting as in Table III. First, we examined the agent's performance when varying the feature set used for navigation. We additionally tested three agents: one trained with the explicit `STOP` action, another constructing edges of its keypoint graphs based on nearest neighbors in the pixel coordinates instead of utilizing the flow magnitudes, and a third one trained using distance-based rewards, i.e., without keypoint reward terms $r_{keypoint_detect}$ and $\Delta_{keypoint_dtg}$. The results are presented in Table V. The results suggest that the proposed exploitation feature provides the most significant performance improvement for image-goal navigation, especially in cases where there is viewpoint overlap at close range. The exploration feature alone is insufficient for the image-goal navigation task, where comprehension of local geometries is crucial. Nonetheless, the exploration feature is required for

TABLE IV

COMPARISON OF KG-NAV WITH IMAGE-GOAL NAVIGATION BASELINES WITH PANORAMIC IMAGE INPUTS ON TEST EPISODES IN GIBSON SCENES.

Method	Memory	Pose Sensor	Depth Sensor	Easy		Medium		Hard		Overall	
				SR	SPL	SR	SPL	SR	SPL	SR	SPL
NTS-D [7]	graph	✔	✔	0.87	0.65	0.58	0.38	0.43	0.26	0.63	0.43
ResNet + GRU + IL	implicit	✔	✗	0.57	0.23	0.14	0.06	0.04	0.02	0.25	0.10
Target-Driven RL [18]	implicit	✔	✗	0.56	0.22	0.17	0.06	0.06	0.02	0.26	0.10
ANS [37]	metric map	✔	✗	0.63	0.45	0.31	0.18	0.12	0.07	0.35	0.23
NTS [7]	graph	✔	✗	0.80	0.60	0.47	0.31	0.37	0.22	0.55	0.38
VGM [2]	graph	✗	✔	0.76	0.40	**0.76**	0.56	**0.62**	**0.49**	0.71	0.48
SPTM [33]	graph	✗	✗	0.64	0.35	0.52	0.27	0.36	0.19	0.51	0.27
Memory-Augmented RL [6]	set of state embeddings	✗	✗	0.78	0.63	0.70	**0.57**	0.60	0.48	0.69	**0.56**
KG-Nav-Equirect	implicit	✗	✗	**0.92**	**0.68**	0.75	0.44	0.50	0.24	**0.72**	0.45
KG-Nav	implicit	✗	✗	**0.92**	0.67	0.74	0.41	0.50	0.25	**0.72**	0.44

TABLE V

ABLATION STUDIES ON KG-NAV WITH PERSPECTIVE IMAGE INPUTS ON STRAIGHT/CURVED EPISODES ON GIBSON SCENES

Path Type	Method	Easy		Medium		Hard		Overall	
		SR	SPL	SR	SPL	SR	SPL	SR	SPL
Straight	KG-Nav	86.5	75.2	**71.2**	**56.5**	**44.9**	**28.0**	**67.5**	**53.2**
	Feature - Exploitation + Exploration	77.8	61.7	62.2	40.4	37.2	19.8	59.1	40.7
	Feature - Exploitation + Spatial	**86.7**	75.2	64.9	52.0	18.5	12.6	56.7	46.6
	Feature - Exploration + Spatial	29.1	17.7	20.4	10.4	11.8	5.8	20.4	11.3
	Feature - Exploration	25.4	16.3	17.8	9.6	11.2	5.7	18.1	10.5
	Action Set - Stop Action Added	81.2	**75.5**	61.6	48.8	34.1	22.6	59.0	49.0
	Graph Construction - Pixel Coord.	83.9	73.0	68.1	51.1	43.4	24.9	65.1	49.7
	Reward - No Keypoint Rewards	85.6	74.8	70.7	52.2	43.9	25.2	66.7	50.7
Curved	KG-Nav	**70.3**	**37.1**	**58.4**	**31.0**	33.5	15.8	**54.1**	**28.0**
	Feature - Exploitation + Exploration	65.3	34.0	53.5	26.6	32.2	**16.1**	50.3	25.6
	Feature - Exploitation + Spatial	66.6	30.0	34.5	17.2	6.3	3.0	35.8	16.7
	Feature - Exploration + Spatial	22.4	9.2	18.8	7.0	9.6	3.9	16.9	6.7
	Feature - Exploration	16.2	7.1	15.5	5.9	9.2	4.2	13.6	5.7
	Action Set - Stop Action Added	57.1	31.3	44.0	23.9	23.1	11.7	41.4	22.3
	Graph Construction - Pixel Coord.	69.4	33.2	57.0	26.4	33.3	14.1	53.2	24.6
	Reward - No Keypoint Rewards	68.7	33.6	58.0	27.0	**34.0**	14.4	53.6	25.0

TABLE VI

AVERAGE SUCCESS RATES AND SUCCESS WEIGHTED BY PATH LENGTH (SPL) FOR THE GIBSON MEDIUM IMAGE-GOAL NAVIGATION TEST SET WITH VARYING OBJECT ADDITION LEVELS

Level	# of Objects	Match Rate	NRNS [4]		OVRL [3]		KG-Nav ($\rho_{th} = 0.55$)		KG-Nav ($\rho_{th} = 0.4$)	
			SR	SPL	SR	SPL	SR	SPL	SR	SPL
0	0	100	25.7 (100%)	15.6 (100%)	60.0 (100%)	29.8 (100%)	**63.8** (100%)	47.1 (100%)	58.5 (100%)	**50.0** (100%)
1	17.4	87.2	23.5 (91.4%)	14.7 (94.0%)	57.6 (96.0%)	27.6 (92.7%)	**63.7** (99.8%)	46.7 (**99.2%**)	62.8 (**107.2%**)	**49.6** (**99.2%**)
2	34.8	80.4	18.3 (71.2%)	11.5 (73.5%)	43.5 (72.5%)	19.1 (64.1%)	52.0 (81.5%)	36.2 (76.8%)	**62.6** (**106.9%**)	**48.1** (**96.3%**)
3	52.2	76.3	14.8 (57.6%)	9.0 (57.5%)	41.2 (68.7%)	16.8 (56.4%)	50.8 (79.7%)	35.1 (74.5%)	**59.6** (**101.7%**)	**42.6** (**85.2%**)
4	69.6	73.9	14.0 (54.5%)	8.6 (55.0%)	36.1 (60.2%)	14.5 (48.7%)	48.6 (76.2%)	31.1 (66.0%)	**53.2** (**90.9%**)	**40.3** (**80.7%**)

episodes in which the agent begins from a distance, and environmental exploration is necessary, as demonstrated by the performance of the Exploitation + Spatial agent in the hard episodes. The additional performance boost by the spatial feature supports the efficacy of the proposed feature computation with the keypoint graph. KG-Nav outperforming KG-Nav with the `STOP` action validates the idea of training with the oracle `STOP` and taking `STOP` actions by the number of keypoints matching when testing. By not learning to `STOP` and focusing on the navigation, we simplify the learning process and improve the sample efficiency of KG-Nav, requiring 20 million and 25 million training frames respectively for the perspective camera and panoramic camera, which are about 4∼25 times fewer training frames than the other state-of-the-art methods in the experiments. The decrease in performance resulting from changing the graph construction method and eliminating keypoint-based rewards provides additional support for the proposed approach to keypoint graph construction and keypoint graph-based reinforcement learning with the newly suggested keypoint-based reward terms.

We present an example of successful image-goal navigation episodes in Figure 3. The agent starts from (a) while observing a curtain in the room. Due to the mismatch of keypoints with a curtain in the target image, the agent wanders around the room, as seen in (b). The agent eventually overcomes confusion and finds a clue at (c), where a curtain in a different room is

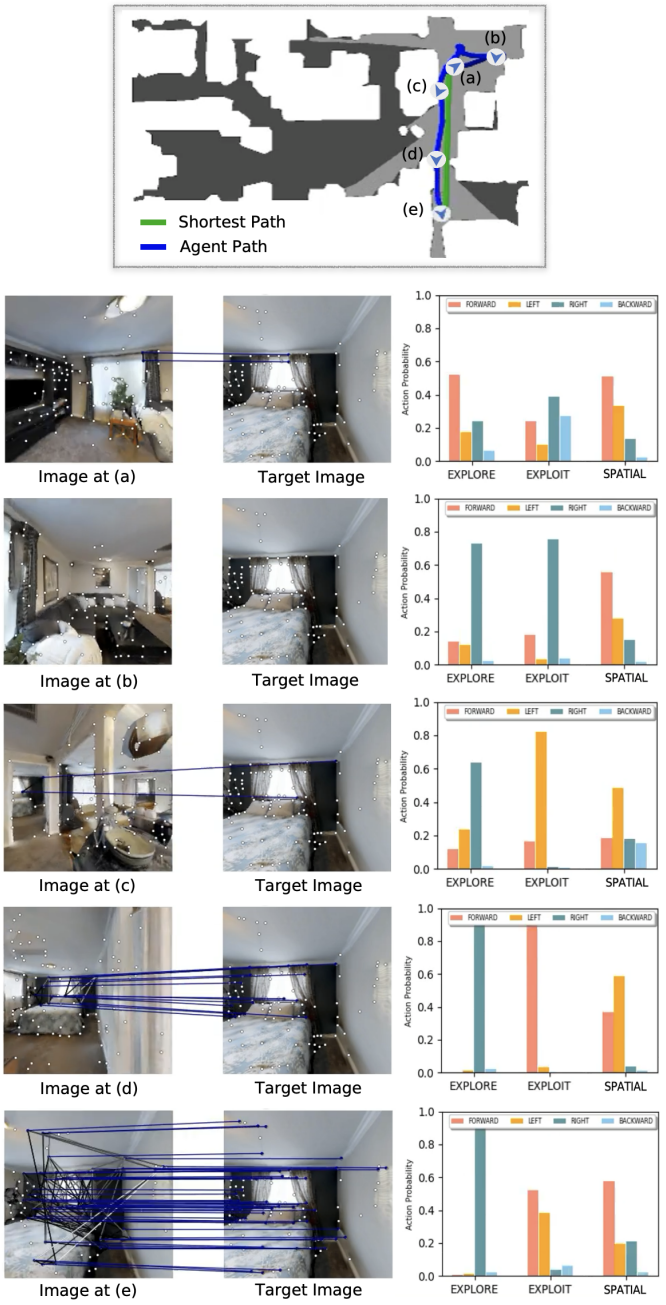
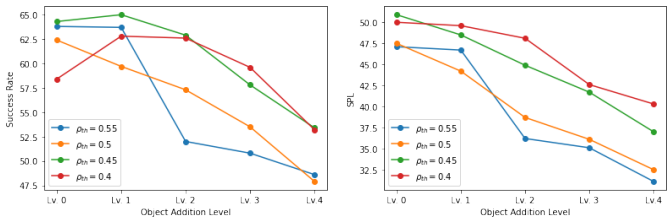


Fig. 3. Example visualization of a successful image-goal navigation episode in the simulation experiment. The top figure depicts the agent's path on a top-down map, where (a)-(e) denotes the starting point, three intermediate points, and the terminal point. Subsequent figures illustrate the corresponding keypoint matches between observed and target images for each (a)-(e), as well as the action probabilities predicted by each module. Edges from keypoint graphs are also depicted in the observed images, where an edge with a darker color has a larger attention weight in the first graph attention layer. Note that the relative scales between the predictions of the three modules are not displayed.

matched, and the exploitation module becomes confident about the direction. At (d), the spatial module makes the agent turn left to avoid collision with the door frame. Finally, the agent arrives near the target viewpoint and executes the STOP action



(a) Success Rates (b) SPL

Fig. 4. Change of the success rate and SPL according to object addition level and ρ_{th}. (a) Success rates (b) SPL

due to sufficient keypoint matches.

E. Robustness to Environmental Changes

We demonstrate the robustness of our method to environmental changes by introducing new objects in five Gibson scenes. Objects were manually curated among the Google Scanned Objects [44] to properly fit the scenes, and we measured how the navigation performance degrades across different levels, gradually introducing more objects. No objects are added to the scene at the object addition level 0, and the objects chosen to suit each scene are placed at the most suitable position at level 1. From level 2 onwards, each object in the selection for each scene is placed once more at the navigable position sampled near the target, disturbing viewpoint matching. The results are shown in Table VI. *Match Rate* denotes the proportion of the keypoints in the target image after placing objects that match the keypoints of the original target image, which is a measure of difficulty. As seen in the table, KG-Nav shows robustness to environmental changes and enjoys an interesting capability to adjust trade-offs between the navigation performance and the robustness to environmental changes by controlling ρ_{th}, which is more clearly shown in Figure 4. Our agent can maintain navigation performance even when the environment gradually changes by choosing relatively low ρ_{th}.

F. Robot Experiments

We conducted a zero-shot transfer to the real world as shown in Figure 5. We deployed a mobile robot with a perspective camera and a panoramic camera. Depicted in the figure are two paths that were taken day and night, respectively. Our method generalizes well in unseen environments due to the use of the keypoint graph constructed with local interest points. We report on the experimental details and results in the following. **Experiment Settings** To implement KG-Nav for a real-world setting, we deploy a Clearpath Jackal robot. Our robot is equipped with an Logitech C930e and a Ricoh Theta V camera for the perspective camera setting and the panoramic camera setting, respectively. For the perspective camera setting, we use the RGB image captured by the C930e which has a size of 360×640 and a diagonal field of view (FoV) of 90 °. For the panoramic camera setting, we use the equirectangular image which is captured by the Theta V and resized to 256×512. We can adapt the policy of our KG-Nav agents trained

with the different image sizes since we use the normalized coordinates for computing the flow features of the keypoint graph. Therefore, we directly apply our KG-Nav and KG-Nav-Equirect agent trained with the Gibson dataset [11] and test the zero-shot transfer performance of the KG-Nav agents. For test sites, we chose three indoor places and two outdoor places on the campus, which are entrances to an elevator, a stationery store, and a bank inside a building and two entrances to a building from outside. With initial distances of $1.5{\sim}10\,m$ from the target viewpoints, we tested three episodes per test site, thereby 15 episodes in total. The number of keypoints we extract from the observation is the same as in simulation experiments, and the stopping condition ρ_{th} is set to 0.5 in the real-world experiments. Exceeding the episode length of 100 or colliding with obstacles also terminates an episode. We measure the approximate distance at the agent's final location to the target viewpoint with Perspective-n-Point (PnP) RANSAC methods.

Sim-to-Real Gap Robot experiments were conducted in the office building and outdoor environments, while KG-Nav agents are trained with simulation in the Gibson scenes [11] which are scanned from indoor houses. Dissimilarity in visual observations between the simulation and real data makes it hard to directly apply the model trained with the simulation data to the real data. Especially models relying on RGB observations have difficulty transferring to unseen data, which makes other works exploit depth observation and modularize navigation policies [1], [36], [45]. However, depth images are often unreliable in outdoor environments and modularization in policy structures often suffers from compounding errors. On the other hand, the KG-Nav agent has strength in transferability since it exploits the underlying geometry from the keypoint graph, obviating the need for applying sim-to-real techniques such as domain adaptation and domain randomization.

Results In the perspective camera setting, the KG-Nav agent succeeded with a success rate of 73.3% and an average final distance of $1.65\,m$. In the panoramic camera setting, the KG-Nav-Equirect agent succeeded with a success rate of 46.7% and an average final distance of $0.80\,m$. OVRL [3] could not succeed in any episode since their offline representation learning is overfitted to the training environment. On the other hand, KG-Nav is shown to generalize well in unseen environments thanks to the use of the keypoint graph constructed with local interest points. Its performance loss compared with the simulation was mostly affected by the camera FoV and the keypoint mismatches due to illumination changes. We also provide qualitative results of the robot experiments in Figure 6 where two successful navigation episodes of the panoramic camera agents are presented. The initial viewpoint in trajectory A is oriented in a direction similar to that of the target viewpoint, but its position is far to the side, yielding few keypoint matches. Taking advantage of the panoramic camera, the agent succeeded by going near the target and then turning around. On the other hand, the initial viewpoint in trajectory B is oriented in a direction opposite to that of the target viewpoint. Again, taking advantage of the panoramic camera, the agent succeeded by making early turns while reaching the target viewpoint. We provide more qualitative results in the various settings in the attached video.

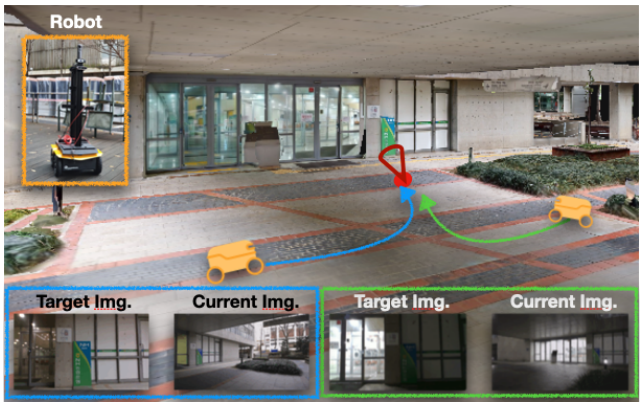


Fig. 5. Sim2Real Experiments. Two paths and the target and currently observed image of each path are depicted.

VI. Conclusion

In this work, we have proposed a keypoint graph-based image-goal navigation policy inspired by classical visual servoing and verified its effectiveness with extensive experiments. Inductive biases from underlying keypoint geometry offer a high sample efficiency and leads to state-of-the-art navigation performance even without a depth and pose sensor. The proposed method also facilitates robust transfer to unseen environments. Meanwhile, the proposed method expects visual descriptor extraction and matching to be precise. An additional performance boost can be made by utilizing more robust keypoint matching methods rather than the simple brute-force matching used for simplicity. KG-Nav also has a relative weakness in long-range navigation due to the lack of matched local features. A simple workaround for this issue is combining KG-Nav with the existing long-range navigation methods which are designed for exploration. We expect this weakness can be ultimately complemented by augmenting the proposed method with external navigation memory and self-supervised learning with data augmentation, which will be our future work.

References

[1] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2020.

[2] O. Kwon, N. Kim, Y. Choi, H. Yoo, J. Park, and S. Oh, "Visual Graph Memory with Unsupervised Representation for Visual Navigation," in *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[3] K. Yadav, R. Ramrakhya, A. Majumdar, V.-P. Berges, S. Kuhar, D. Batra, A. Baevski, and O. Maksymets, "Offline Visual Representation Learning for Embodied Navigation," *arXiv:2204.13226*, 2022.

[4] M. Hahn, D. S. Chaplot, S. Tulsiani, M. Mukadam, J. M. Rehg, and A. Gupta, "No RL, No Simulation: Learning to Navigate without Navigating," in *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

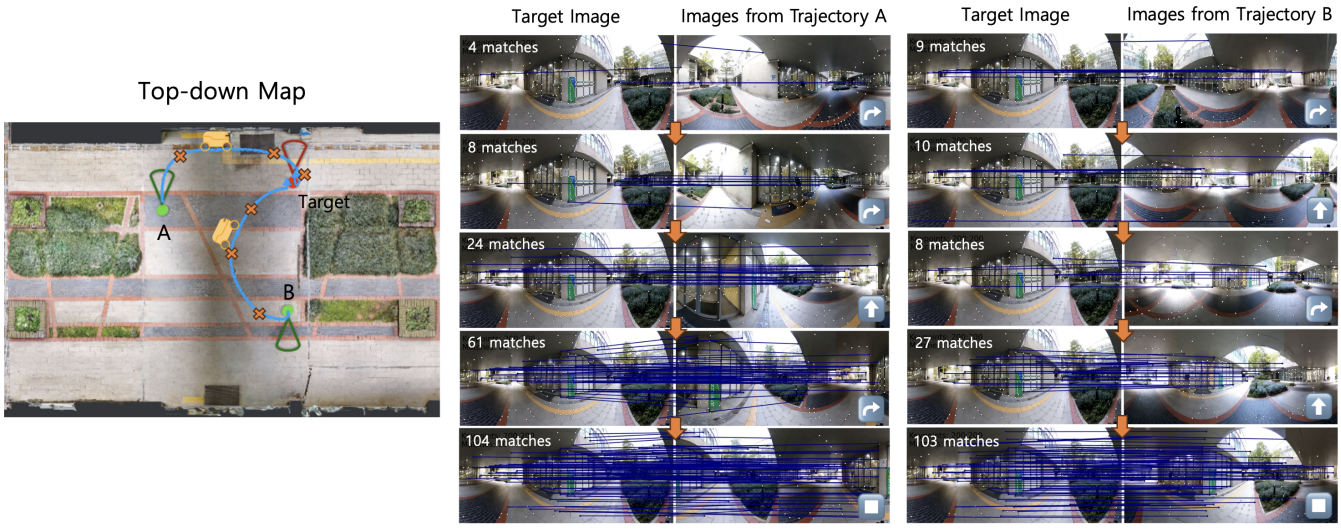


Fig. 6. Example visualization of two successful image-goal navigation episodes in the robot experiments. The leftmost figure depicts the agent's paths on a top-down map where green fan-shapes represent the initial viewpoints of trajectory A and B, and the red fan-shape indicates the target viewpoint. The middle and right figures show keypoint matches between the target image and the observed image at the initial viewpoint, intermediate viewpoints marked with '$\times$' on the top-down map, and the last viewpoint of each trajectory. The figure also displays the number of keypoint matches and the executed action at each viewpoint. Note that the top-down map is just for visualization and not given to the agent.

TABLE VII
PPO PARAMETERS FOR TRAINING KG-NAV

Parameters	Value
clip_param	0.1
ppo_epoch	4
num_mini_batch	2
value_loss_coef	0.5
entropy_coef	0.01
lr	2.5e-4
eps	1e-5
max_grad_norm	0.5
num_steps	128
hidden_size	512
use_gae	True
gamma	0.99
tau	0.95
use_linear_clip_decay	False
use_linear_lr_decay	False
reward_window_size	50

[5] Z. Al-Halah, S. K. Ramakrishnan, and K. Grauman, "Zero Experience Required: Plug & Play Modular Transfer Learning for Semantic Visual navigation," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[6] L. Mezghani, S. Sukhbaatar, T. Lavril, O. Maksymets, D. Batra, P. Bojanowski, and K. Alahari, "Memory-Augmented Reinforcement Learning for Image-Goal Navigation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[7] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural Topological SLAM for Visual Navigation," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.

[9] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On Evaluation of Embodied Navigation Agents," *arXiv:1807.06757*, 2018.

[10] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik *et al.*, "Habitat: A Platform for Embodied AI Research," in *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9339–9347.

[11] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-World Perception for Embodied Agents," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] A. Majumdar, G. A. Sigurdsson, R. Piramuthu, J. Thomason, D. Batra, and G. S. Sukhatme, "SSL Enables Learning from Sparse Rewards in Image-Goal Navigation," in *Proc. of the International Conference on Machine Learning (ICML)*, 2022.

[13] F. Chaumette and S. Hutchinson, "Visual Servo Control. I. Basic Approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[14] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From Coarse to Fine: Robust Hierarchical Localization at Large Scale," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[15] B. Cao, A. Araujo, and J. Sim, "Unifying Deep Local and Global Features for Image Search," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2020.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[18] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[19] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling Task Transfer Learning," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[20] A. Sax, J. O. Zhang, B. Emi, A. Zamir, S. Savarese, L. Guibas, and J. Malik, "Learning to Navigate Using Mid-Level Visual Priors," in *Proc. of the Conference on Robot Learning (CoRL)*, 2019.

[21] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] S. Hausler, S. Garg, M. Xu, M. Milford, and T. Fischer, "Patch-

netvlad: Multi-Scale Fusion of Locally-Global Descriptors for Place Recognition," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[23] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-Supervised Interest Point Detection and Description," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018.

[24] J. Revaud, P. Weinzaepfel, C. De Souza, N. Pion, G. Csurka, Y. Cabon, and M. Humenberger, "R2D2: Repeatable and Reliable Detector and Descriptor," *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[25] Z. Luo, L. Zhou, X. Bai, H. Chen, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan, "ASLFeat: Learning Local Features of Accurate Shape and Localization," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "Loftr: Detector-free local feature matching with transformers," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[27] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks," in *Proc. of the IEEE international conference on computer vision*, 2015.

[28] Y. Harish, H. Pandya, A. Gaud, S. Terupally, S. Shankar, and K. M. Krishna, "DFVS: Deep Flow Guided Scene Agnostic Image Based Visual Servoing," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[29] Y. Li and J. Košecka, "Learning View and Target Invariant Visual Servoing for Navigation," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[30] F. Sadeghi, "DIViS: Domain Invariant Visual Servoing for Collision-Free Goal Reaching," in *Proc. of the Robotics: Science and Systems (RSS)*, 2019.

[31] M. N. Qureshi, P. Katara, A. Gupta, H. Pandya, Y. Harish, A. Sanchawala, G. Kumar, B. Bhowmick, and K. M. Krishna, "RTVS: A Lightweight Differentiable MPC Framework for Real-Time Visual Servoing," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[32] P. Katara, Y. Harish, H. Pandya, A. Gupta, A. Sanchawala, G. Kumar, B. Bhowmick, and M. Krishna, "DeepMPCVS: Deep Model Predictive Control for Visual Servoing," in *Proc. of the Conference on Robot Learning (CoRL)*, 2021.

[33] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-Parametric Topological Memory for Navigation," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

[34] N. Kim, O. Kwon, H. Yoo, Y. Choi, J. Park, and S. Oh, "Topological Semantic Graph Memory for Image-Goal Navigation," in *Proc. of the Conference on Robot Learning (CoRL)*, 2022.

[35] J. Wasserman, K. Yadav, G. Chowdhary, A. Gupta, and U. Jain, "Last-Mile Embodied Visual Navigation," in *Proc. of the Conference on Robot Learning (CoRL)*, 2022.

[36] N. Yokoyama, Q. Luo, D. Batra, and S. Ha, "Benchmarking Augmentation Methods for Learning Robust Navigation Agents: the Winning Entry of the 2021 iGibson Challenge," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[37] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to Explore Using Active Neural Slam," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2020.

[38] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[39] K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision GNN: An Image is Worth Graph of Nodes," in *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2019.

[41] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[42] J. S. Heo, Y. Choi, and S. Oh, "Image-Goal Navigation via Metric Mapping and Keypoint-based Reinforcement Learning," in *Proc. of the International Conference on Control, Automation and Systems (ICCAS)*, 2021.

[43] Y. Choi and S. Oh, "Image-Goal Navigation via Keypoint-Based Reinforcement Learning," in *Proc. of the International Conference on Ubiquitous Robots (UR)*, 2021.

[44] A. G. Francis, B. Kinman, K. A. Reymann, L. Downs, N. Koenig, R. M. Hickman, T. B. McHugh, and V. O. Vanhoucke, "Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items," *arXiv:2204.11918*, 2022.

[45] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," *arXiv:2212.00922*, 2022.