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

Author Names Omitted for Anonymous Review. Paper-ID 395

I. Robot Experiments

We conduct a zero-shot transfer to the real world as shown in Figure 1. We deploy a mobile robot with a perspective camera and a panoramic camera. Depicted in the figure are two paths that are 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 subsections.

A. Experiment Settings

To implement KG-Nav for a real-world setting, we deploy a Clearpath Jackal robot. Our robot is equipped with an Intel RealSense D435 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 D435 which has a size of 640$\times$360 and a field of view (FoV) of 69$^\circ\times$42°. For the panoramic camera setting, we use the equirectangular image which is captured by the Theta V and resized to 512$\times$256. 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 [1] and test the zero-shot transfer performance of the KG-Nav agents.

For test sites, we choose 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 test 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 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.

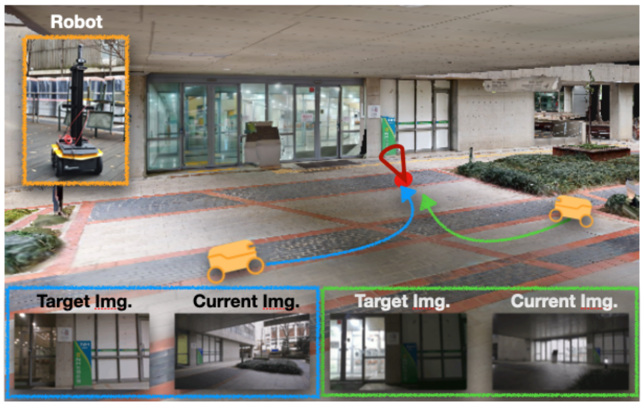

Fig. 1. Sim2Real Experiments. Two paths and the target and currently observed image of each path are depicted.

B. Sim-to-Real Gap

Robot experiments are conducted in the office building and outdoor environments, while KG-Nav agents are trained with simulation in the Gibson scenes [1] 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 [2]–[4]. 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.

C. Quantitative Results

In the perspective camera setting, the KG-Nav agent succeeded with a success rate of 73.3%, and an average final distance 1.65 m. In the perspective camera setting, the KG-Nav-Equirect agent succeeded with a success rate of 46.7%, and an average final distance 0.80 m. OVRL [5] could not succeed in any episode since their offline representation learning

TABLE I
ABLATION STUDIES ON REWARDS AND KEYPOINT GRAPH PROCESSING

		SR	SPL
Rewards	$r_{success} + r_{slack} + \Delta_{dtg}$	33.9	7.3
	$r_{success} + r_{slack} + \Delta_{dtg} - r_{collision} - r_{backward}$	33.6	10.1
	$r_{success} + r_{slack} - r_{collision} - r_{backward} + r_{keypoint\_detect} + \Delta_{keypoint\_dtg}$	26.2	17.0
	$r_{success} + r_{slack} + \Delta_{dtg} - r_{collision} - r_{backward} + r_{keypoint\_detect} + \Delta_{keypoint\_dtg}$	30.5	13.5
Keypoint Graph	`Linear Layers` (No Graph)	24.8	7.9
	`KFN` + `GAT` + `Mean Pool`	27.0	12.5
	`KFN` + `KNN` + `GAT` + `Mean Pool`	30.1	9.4
	`KFN` + `KNN` + `GAT` + `Max Pool`	33.6	10.1
	`KFN` + `KNN` + `GCN` + `Max Pool`	32.6	9.3

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. In addition, its performance is mostly affected by the FoV of the observations.

D. *Qualitative Results*

We provide qualitative results in the various settings in the attached video.

II. ABLATION STUDIES ON REWARDS AND KEYPOINT GRAPH CONSTRUCTION

We conduct a comprehensive ablation study on KG-Nav to understand the contribution of each component. For the ablation experiments, we train KG-Nav agents with a perspective camera with 10M interactions in the Gibson medium scenes [1]. We vary reward terms and keypoint graph constructions when training KG-Nav agents and evaluated their success rate (SR) and success weighted by path length (SPL) from 500 episodes in the test scenes. Success is defined by the tighter condition of distance and angle just as the experiments for Tab. I in the paper. Unless stated otherwise, default settings are $r = r_{success} + r_{slack} + \Delta_{dtg} - r_{collision} - r_{backward}$ for the reward function and `KFN` + `KNN` + `GAT` + `Max Pool` for the keypoint graph construction, respectively.

A. *Rewards*

We evaluate KG-Nav agents trained with different combinations of reward terms presented in the paper. and the result is given in Table I. We refer explanations about each term to Section III-D of the paper. With the most basic reward combination $r = r_{success} + r_{slack} + \Delta_{dtg}$ for the image-goal navigation task, the agent succeeds with a decent success rate while its SPL is relatively low due to the wall following behavior and collision with obstacles. Adding penalties for collision and `MOVE_BACKWARD` action, which makes the default reward setting, strikes a good balance between preventing collision with obstacles and navigation performance which is also validated in [3]. Replacing the distance reward Δ_{dtg} with the keypoint detection reward $r_{keypoint\_detect}$ and the keypoint distance reward $\Delta_{keypoint\_dtg}$ loses success rate but gains SPL since the keypoint detection reward facilitates exploration in unseen environments. Loss in success rates attributes to the mismatch between the reward function and task definition where success is defined by distance and angle. Therefore, putting the seven reward terms altogether gains exploration performance with a cost of slight loss in success rate compared to the default setting as seen in Tab. I.

B. *Keypoint Graph Processing*

We evaluate KG-Nav agents trained with different methods for keypoint graph processing and the result is given in Table I. The result shows that constructing a keypoint graph and processing it with a graph neural network improves navigation performance with a margin. For keypoint graph construction, utilizing both k-nearest neighbors (`KNN`) and k-farthest neighbors (`KNN`). For the graph neural network to process the keypoint graph, the graph attention layer (`GAT`) followed by global max pooling leads to better performance compared to the graph convolution (`GCN`) and global mean pooling. Varying the number of neighbors from 2 to 10 does not change performance much in our experiments and therefore we choose the number of neighbors considering the computation efficiency.

TABLE II
RL 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

III. Training Details of KG-Nav

We train our agents with Proximal Policy Optimization (PPO) algorithm [6] implemented in the `habitat-lab` repository [7]. The parameter set used for training KG-Nav is presented in Tab. II.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

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

[7] 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.