Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

Metric-Free Exploration for Topological Mapping by Task and Motion Imitation in Feature Space

Yuhang He[1,*] Irving Fang[2,*] Yiming Li[2] Rushi Bhavesh Shah[2] Chen Feng[2,✉]
[1] Department of Computer Science, University of Oxford, Oxford, United Kingdom yuhang.he@cs.ox.ac.uk
[2] Tandon School of Engineering, New York University, New York, United States {zf540;yimingli;rs7236;cfeng}@nyu.edu

***Abstract*—We propose *DeepExplorer*, a simple and lightweight metric-free exploration method for topological mapping of unknown environments. It performs task and motion planning (TAMP) entirely in image feature space. The task planner is a recurrent network using the latest image observation sequence to hallucinate a feature as the next-best exploration goal. The motion planner then utilizes the current and the hallucinated features to generate an action taking the agent towards that goal. *Our novel feature hallucination* enables *imitation learning with deep supervision* to jointly train the two planners more efficiently than baseline methods. During exploration, we iteratively call the two planners to predict the next action, and the topological map is built by constantly appending the latest image observation and action to the map and using visual place recognition (VPR) for loop closing. The resulting topological map efficiently represents an environment's connectivity and traversability, so it can be used for tasks such as visual navigation. We show *DeepExplorer*'s exploration efficiency and strong sim2sim generalization capability on large-scale simulation datasets like Gibson and MP3D. Its effectiveness is further validated via the image-goal navigation performance on the resulting topological map. We further show its strong zero-shot sim2real generalization capability in real-world experiments. The source code is available at https://ai4ce.github.io/DeepExplorer/.**

I. Introduction

Mobile agents often create maps to represent their surrounding environments [1]. Typically, such a map is either topological or metrical (including hybrid ones). We consider a topological map to be metric-free, which means it does not explicitly store global/relative position/orientation information with measurable geometrical accuracy [2], [3]. Instead, it is a graph that stores local sensor observations, such as RGB images, as graph nodes and the spatial neighborhood structure (and often navigation actions) as graph edges that connects observations taken from nearby locations. While metric maps are often reconstructed by optimizing geometric constraints between landmarks and sensor poses from classic simultaneous localization and mapping (SLAM), topological maps have recently attracted attention in visual navigation tasks due to the simplicity, flexibility, scalability, and interpretability [4]–[9].

There are two robot exploration methods to collect data to construct a topological map in a new environment. The first and also the simplest one is to let the agent explore the new environment through metric-free *random walk*, after which the topological map could be built by projecting the recorded observations into a feature space and adding graph edges from temporal connections and loop closures [4]. However random walking is very inefficient especially in large or complex rooms, leading to repetitive visits to the nearby local areas. The other way is to design a navigation policy that controls the agent to more effectively explore the area while creating the map. It is known as *active SLAM* and often involves some metric information (e.g., distance and orientation) from either additional input modalities [8], [10] or intermediate estimations [5]. Could we combine the merits of the two ways by finding *an exploration policy that (1) is metric-free thus simple and lightweight in hardware and model complexity, and (2) exhibits strong generalization ability to explore unknown environment* for topological map construction?

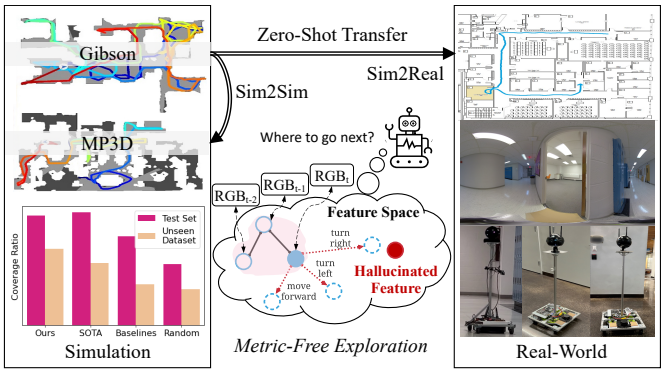


Fig. 1. *DeepExplorer* illustration: it is metric-free and plans in the feature space. It jointly hallucinates the next step feature to visit and predicts the appropriate action taking the agent to the hallucinated feature. We show that *DeepExplorer* is efficient in simulation environment and possesses strong zero-shot sim2real capability in exploring real-world environment.

To achieve this objective, we propose *DeepExplorer* (see Fig. 1), a new framework to achieve metric-free efficient exploration by imitating easy-to-access expert exploration demonstrations [11]. The expert demonstration is a sequence of image and action pairs taken on a route that efficiently covers a new environment. This could come from either an *oracle policy* having full access to virtual environments or simply a *human expert* in the real world.

DeepExplorer follows the task and motion planning formalism (TAMP) and entirely works in image feature space. Its task planner, a two-layer LSTM [12] network, conceives the next best goal feature to be explored by hallucination from the latest sequence of observed image features. Its motion planner, a simple multi-layer perceptron (MLP), fuses the current and

* Equal contributions.
✉ Corresponding author: cfeng@nyu.edu. The work is supported by NSF grants 2238968 and 2026479.

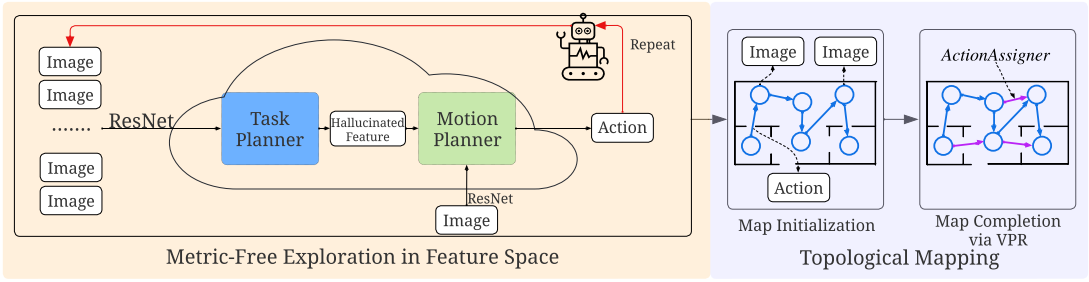


Fig. 2. **Workflow of *DeepExplorer* and the following topological mapping.** The agent explores a novel environment by task and motion planning in feature space (left); The following topological mapping completes the initial topological map by adding new edges via VPR and *ActionAssigner*.

the hallucinated features to predict the best action moving the agent toward the hallucinated feature. Both the two planners are trained jointly by deep supervision [13] of per-step feature hallucination and action prediction. The trained *DeepExplorer* are deployed by iteratively calling the task and the motion planners to predict the next action.

DeepExplorer is designed for active topological mapping of unknown environments. During each exploration step, the topological map is updated by adding the latest image observation as a new node and the action on the new edge. We further adopt VLAD-based [14] visual place recognition (VPR) [15] for loop closing, adding additional new edges between image pairs that are temporally disjoint but spatially close. In the meantime, we train an *ActionAssigner* to assign each VPR-added new edge with corresponding actions that move the agent from one node to the other. We call the above process as *Topological Mapping* (see Fig. 2). Finally, the completed topological map efficiently represents environment connectivity and traversability. We can apply it to various robot tasks like visual navigation [4].

We demonstrate the advantage of *DeepExplorer* on both visual **exploration** and **navigation** tasks. We train it on Gibson [16] simulation dataset and test its **exploration** efficiency on both Gibson validation and MP3D [17] dataset (for zero-shot sim2sim generalization test). We further show its strong zero-shot sim2real generalization capability by directly deploying the Gibson-trained *DeepExplorer* to explore a real-world environment. For the navigation task, we run experiments on both Gibson [16] and MP3D [17] dataset with the topological map built by *DeepExplorer*.

In summary, our contributions are listed as follows:

- We propose *DeepExplorer* for efficient metric-free visual exploration based on task and motion planning entirely in an image feature space.
- Our novel feature hallucination enables the imitation learning of such a feature space via deep supervision, whose importance is shown in our ablation study.
- Through both exploration and navigation experiments, we show the efficiency and strong sim2sim/sim2real generalization capability of *DeepExplorer*.

II. Related Work

Topological Map in Exploration and Navigation. Inspired by the animal and human psychology [18], a large amount of work has recently proposed to build topological map to represent an environment [4], [5], [7], [19]–[23]. They use the topological map for tasks such as navigation [4], [5], [7], [8], [23], exploration [4], [8], [9], [19], [20], [22] and planning [21]. To build the topological map, they combine various sensors such as RGB image, depth map [9], [23], pose [5], [8], [21] and even LiDAR scanner [19], [22]. Some of them further adopt data-hungry and computation-demanding Reinforcement Learning (RL) techniques to train the model to construct the topological map [5], [7], [8]. Kwon *et al.* [7] combine imitation learning (IL) and RL to train the model. Some of these methods [5], [8], [21] involve metric information to construct the topological map. N. Savinov *et. al.* [4] use the random walk to construct the topological map, which inevitably leads to an inefficient topological map. TSGM [9] jointly adds surrounding objects during topological map construction. Unlike these prior works, our *DeepExplorer* is completely metric-free and simple in experimental configuration (just RGB image, much smaller expert demonstration size).

Hallucinating Future Feature. The idea of hallucinating future latent features has been discussed in other application domains. Previous work has utilized this idea of visual anticipation in video prediction/human action prediction [24]–[28], and researchers have applied similar ideas to robot motion and path planning [29]–[32]. As stated in [24], [25], [28], visual features in the latent space provide an efficient way to encode semantic/high-level information of scenes, allowing us to do planning in the latent space, which is considered more computationally efficient when dealing with high-dimensional data as input [33], [34]. Different from previous robotics work, we take advantage of this efficient representation by adding deep supervision when anticipating the next visual feature, which was computationally intractable if we were to operate at the pixel level.

Deeply-Supervised Learning has been extensively explored [13], [35]–[37] during the past several years. The main idea is to add extra supervision to various intermediate layers of a deep neural network in order to more effectively train deeper neural networks. In our work, we adopt a similar idea to deeply supervise the training of feature hallucination and action generation.

Task and Motion Planning. Task and motion planning (TAMP) divides a robotic planning problem into high-level

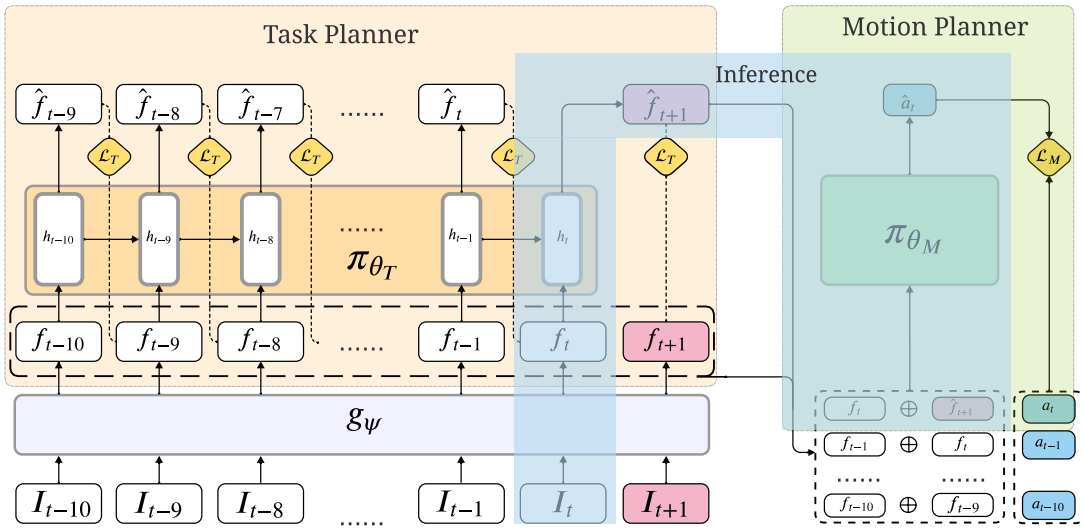


Fig. 3. **Training and inference for task and motion imitation.** Feature extractor g_ψ takes image I_t as input and generates the corresponding feature vector f_t. *TaskPlanner* π_{θ_T} is a recurrent neural network (RNN) consuming a sequence of features $\{f_{t-10}, \cdots, f_t\}$ to hallucinate the next best feature to visit $\hat{f}_{t+1}$. *MotionPlanner* π_{θ_M} consumes the concatenation (denoted by $\bigoplus$) of f_t and $\hat{f}_{t+1}$ and generates the action to move the agent towards the hallucinated feature. During training, we supervise all the intermediate outputs including the intermediate hallucinated features $\{\hat{f}_{t-9}, \cdots, \hat{f}_t\}$ and the intermediate actions $\{\hat{a}_{t-10}, \cdots, \hat{a}_{t-1}\}$, in addition to the final output $\hat{f}_{t+1}$ and $\hat{a}_t$. During inference, current observation I_t is firstly encoded and fed into π_{θ_T} to hallucinate $\hat{f}_{t+1}$, and then $\hat{f}_{t+1}$ combined with the f_t is fed into π_{θ_M} for motion planning. $\mathcal{L}_T$ is L_2 loss and $\mathcal{L}_M$ is cross entropy loss (the subscripts T and M denote **T**ask and **M**otion respectively). h_t denotes the hidden state of RNN.

task allocation (task planning) and low-level action for task execution (motion planning). This hierarchical framework is adopted in many robotic tasks such as manipulation [38], [39] exploration [40] and navigation [41], [42]. Such a framework allows us to leverage high-level information about the scenes to tackle challenges in local control techniques [43]. In this work, to perform active topological mapping of a novel environment, the agent firstly reasons at the highest level about the regions to navigate: hallucinate the next best feature point to visit. Afterward, the agent takes an action to get to the target feature. The whole procedure is totally implemented in feature space without any metric information.

Imitation Learning aims to mimic human behavior or expert demonstrations for a given specific task [11], [44], [45]. The agent is trained to perform tasks by directly observing demonstrations [44], [45]. In our work, the expert demonstration is a set of image-action pair sequences that an agent would observe along a route that efficiently, albeit sub-optimally, covers an environment. It is widely accessible in either real-world or simulated environments (e.g. from human experts or maps of environments). More details in Sec IV-A.

III. Topological Exploration in Feature Space

Our topological map is represented by a graph $\mathcal{G} = (\mathcal{I}, \mathcal{A})$, where the graph nodes denoted by $\mathcal{I}$ is a set of RGB panoramic image observations collected by the agent at different locations $\mathcal{I} = \{I_1, I_2, \cdots, I_N\}$ (where N denotes the number of nodes), and the edges denoted by $\mathcal{A}$ is composed of a set of actions $a_{(I_i, I_j)} \in \mathcal{A}$ which moves an agent between the two spatially adjacent observations I_i and I_j. Each RGB panoramic image is of size 256×512, and the action space consists of three basic actions: `move_forward`, `turn_left`, and `turn_right`.

Our visual exploration aims at maximizing the topological map coverage over an environment given a certain step budget N. The coverage of the topological map denoted by $\mathcal{C}$ is defined as the total area in the map that is known to be traversable or non-traversable. Mathematically, let π_θ denote the policy network parameterized by θ, a_t denote the action taken at step t, and $\Delta\mathcal{C}(a_t)$ denote the gain in coverage introduced by taking action a_t, the following objective function is optimized to obtain the optimal exploration policy π^*:

$$\pi^* = \arg\max_{\pi_\theta} \mathop{\mathbb{E}}_{a_t \sim \pi_\theta} \left(\sum_{t=1}^{N} \Delta\mathcal{C}(a_t) \right) . \tag{1}$$

Learning from expert demonstrations. In literature, most works solve Eq. (1) by reinforcement learning to maximize the reward [46], [47], such solutions are not only data-hungry but also require complicated training involving metric information. Differently, we adopt imitation learning [11] to let our policy network π_θ mimic the output of the expert policy $\tilde{\pi}$ which could come from either an *oracle policy* having full access to virtual environments or simply a *human expert* in real world (more discussion is in Sec. IV-A). Hence, our objective is to minimize the difference between our policy network and the expert policy:

$$\pi^* = \arg\min_{\pi_\theta} \mathcal{L}(\pi_\theta, \tilde{\pi}) , \tag{2}$$

where $\mathcal{L}$ measures the discrepancy between two policies. We propose the task and motion imitation in feature space to solve Eq. (2) which will be introduced in the following (see Fig. 3). We respectively introduce the feature extraction (III-A), the policy network π_θ composed of a *TaskPlanner* denoted by π_{θ_T} (III-B) as well as a *MotionPlanner* denoted by π_{θ_M} (III-C), and the deeply-supervised learning strategy (III-D).

A. Image Feature Extraction

We firstly encode each visual observation $I_t \in \mathcal{I}(t = 1, 2, ..., N)$ with a feature extractor g_ψ parameterized by ψ which uses the ImageNet [48] pre-trained ResNet18 backbone [49]. The feature embedding $f_t \in \mathbb{R}^d (d = 512)$ is obtained by $f_t = g_\psi(I_t)$, (see Fig. 3). Note that g_ψ is jointly optimized with the task planner π_{θ_T} as well as the *MotionPlanner* π_{θ_M} via imitation learning.

B. Task Planner for Next Best Feature Hallucination

TaskPlanner π_{θ_T} parameterized by θ_T takes the most recent m-step visual features $\mathcal{F} = \{f_{t-m}, \cdots, f_t\}$ as input, and learns to hallucinate the next best feature to visit which is denoted by $\hat{f}_{t+1}$, see Fig. 3. In specific, π_{θ_T} is a two-layer LSTM [12]:

$$\hat{f}_{t+1} = \pi_{\theta_T}(f_{t-m}, \cdots, f_t | \theta_T) \ . \tag{3}$$

To save computation, π_{θ_T} only takes the most recent m-step features as input and we empirically find that $m = 10$ achieves good performance. In other words, *TaskPlanner* is only equipped with a short-term scene memory, and it tries to extend the feature space as quickly as possible in order to guide the agent to perform efficient exploration. Essentially, *TaskPlanner* is planning in the feature space. This efficient representation of the environment enables us to deploy deep supervision strategy introduced in Section III-D.

C. Motion Planner for Action Generation

MotionPlanner π_{θ_M} parameterized by θ_M takes the hallucinated feature $\hat{f}_{t+1}$ and the current feature f_t as input, and outputs the action taking the agent towards the hallucinated goal (see Fig. 3). Specifically, π_{θ_M} is a multi-layer-perceptron (MLP) taking the concatenation of two features as input to classify the action:

$$\hat{a}_t = \pi_{\theta_M}(\hat{f}_{t+1}, f_t | \theta_M) \ . \tag{4}$$

D. Deeply-Supervised Imitation Learning Strategy

Our imitation pipeline is shown in Fig. 3. Given an expert exploration demonstration including a sequence of images and the corresponding expert actions $\mathcal{E} = \{\{I_1, a_1\}, \{I_2, a_2\}, \cdots, \{I_N, a_N\}\}$, we adopt the deeply-supervised learning strategy [13] to jointly optimize the feature extractor g_ψ, task planner π_{θ_T}, and *MotionPlanner* π_{θ_M}. Ultimately, our objective in Eq. (2) becomes,

$$\min_{\psi, \theta_T, \theta_M} \sum_{t=1}^{N-1} \mathcal{L}_T(\hat{f}_{t+1}, f_{t+1}) + \sum_{t=1}^{N} \mathcal{L}_M(\hat{a}_t, a_t) \ , \tag{5}$$

where $\mathcal{L}_T$ is L_2 loss to measure the discrepancy between two features, and $\mathcal{L}_M$ is cross-entropy loss to make the model imitate the expert action. The desired target feature f_{t+1} is obtained by $f_{t+1} = g_\psi(I_{t+1})$ (I_{t+1} is obtained from the expert demonstration $\mathcal{E}$), the desired action a_t is also read from $\mathcal{E}$, the hallucinated feature $\hat{f}_{t+1}$ is calculated by Eq. (3), and the generated action $\hat{a}_t$ is computed by Eq. (4). For each training iteration, we randomly clip $m+1$ observations and the corresponding m actions from an expert exploration ($m = 10$ and $N \gg m$), and feed them to g_ψ, π_{θ_T}, and π_{θ_M}. During exploration, we iteratively take the latest m image observations as input, after which we first call task planner π_{θ_T} to hallucinate the next best feature and then motion planner π_{θ_M} to predict the next best action taking the agent to the hallucinated feature accordingly. By constantly executing the predicted action, the agent efficiently explores an environment.

The whole pipeline is shown in Fig. 3, in which we deeply supervise all intermediate output. Specifically, in *TaskPlanner*, instead of simply hallucinating the next best feature, we simultaneously hallucinate all intermediate feature for each step and supervise all hallucinations by truly image observations. In *MotionPlanner*, we deeply supervise the action prediction in a similar fashion. We show by experiment that such deeply-supervised learning strategy [13] endows the agent with more powerful exploration capability.

TABLE I
TOTAL LOOP CLOSING TIME FOR TOPOLOGICAL MAPPING ON GIBSON 14 ROOM SCENES. HARDWARE: 10 CORES OF INTEL XEON PLATINUM 8268, 32GB RAM, AND AN SSD. SPTM REQUIRES AN NVIDIA A100 GPU.

Methods	Total Time Spent
SPTM [4]	$\approx$ 2.1 hrs
VLAD-Based VPR (Ours)	$\approx$ 0.2 hrs

E. VPR for Loop Closing

The topological map $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ initialized by the active exploration experience in Sec. III is unidirectionally connected in the temporal axis. Each node (a panoramic RGB image observation) is just connected with its preceding node and next node, failing to reflect the nodes' spatial adjacency. We propose to further complete the initial map $\mathcal{G}$ by adding edges to any two unconnected nodes if they possess a high visual similarity. In this work, we adopt VLAD-based visual place recognition (VPR) [14], [50] to measure the "visual similarity" between two nodes.

Specifically, given N unidirectionally connected image nodes collected during exploration in a room scene, we extract the local SIFT [51] feature for each image. Then we get the global VLAD [14] descriptor for each image by first clustering all SIFT features with K-Means [52] into k centroids (in our case $k = 16$), and then stacking the residuals between the local SIFT features and centroids. After VLAD descriptors construction, we store all VLAD features into a ball tree [43], [53] with leaf size 60. Then we can query each image's top-N "most visually similar" images from the corresponding ball tree, the node pairs whose similarity score is below a threshold (in our case 1.15) are added edges.

It is worth noting that our VLAD-based VPR is more efficient for loop closing than SPTM [4] which uses a binary classification network that requires exhaustive pairwise checking to detect loops. We report the average loop closing time of the two methods on all the 14 Gibson test rooms in Table I, showing VLAD-based VPR's speed advantage.

Apart from the VPR, we train a model named *ActionAssigner* to assign an action list to each new edge. The architecture of *ActionAssigner* is similar to *MotionPlanner*, except that *ActionAssigner* predicts a sequence of actions with two node features as input, while *MotionPlanner* is a one-step action predictor (predict just one action).

After topological mapping, the completed topological map represents a room scene through the edges between nodes and the actions corresponding to each edge. It reflects both spatial adjacency and traversability of the room scene so that it can be used for navigation tasks. Given the image observations for the start and goal positions, we localize them on a topological map via the same VPR procedure. Once localized, we apply Dijkstra's algorithm [54] to find the shortest path between the two nodes. We can then navigate the agent from the start position to the goal without metric information.

IV. Experiments

We test *DeepExplorer* on two datasets: Gibson [16] and Matterport3D (MP3D) [17] dataset on Habitat-lab platform [55]. The two datasets are collected in real indoor spaces by 3D scanning and reconstruction methods. The agents can be equipped with multi-modality sensors to perform various robotic tasks. The average room size of MP3D (100 m^2) is much larger than that of Gibson ($[14m^2, 85m^2]$).

We run experiments on two tasks: (1) **autonomous exploration** proposed by Chen *et al.* [46] and discussed in [59], in which the target is to maximize an environment coverage within a fixed step budget (1000-step budget following [8]), and (2) **image-goal navigation** where the agent uses the constructed topological map to navigate from current observation to target observation. Regarding the exploration, we employ two evaluation metrics: (1) coverage ratio which is the percentage of the covered area over all navigable area, and (2) absolute covered area (m^2). We exactly follow the setting by ANS [8] that a point is covered by the agent if it lies within the agent's field-of-view and is less than $3.2m$ away. Regarding the navigation, we adopt two evaluation metrics: shortest path length (SPL) and success rate (Succ. Rate) [56]. We again follow ANS [8] to train DeepExplorer on Gibson training dataset (72 scenes), and test DeepExplorer on Gibson validation dataset (14 scenes) and MP3D test dataset (18 scenes). Testing on the MP3D dataset helps to show DeepExplorer's generalizability.

A. Experiment Configuration

Exploration setup. In exploration, we independently explore each scene 71 times, each time assigning the agent a random starting point (created by a random seed number). We keep track of all the random seed numbers for result reproduction. We use Habitat-lab's sliding function so that the agent will not be forced to stop but instead continue the exploration when it collides with the wall. In order to generate the initial 10 steps required by *DeepExplorer*, we constantly let the agent execute `move_forward` action. Once it collides with the wall, it randomly chooses `turn_left` or `turn_right` action to continue to explore. Afterward, we iteratively call *TaskPlanner* and *MotionPlanner* to efficiently explore the environment. During *DeepExplorer*-guided exploration, we allow the agent to actively detect its distance with surrounding obstacles or walls (by using a distance sensor). When the agent's forward-looking distance to the closest obstacle or wall is less than 2-step distances and the *DeepExplorer* predicted action is `move_forward`, we constantly execute `turn_right` or `turn_left` actions to the direction which has the largest ego-centric range distance. Note that using such a *local* obstacle avoidance scheme does not lead to unfair comparisons for *global* exploration because the baseline methods internally preserve a metric map, which serves a similar purpose to help the agent avoid *local* obstacles.

We experiment with two locomotion setups: the first one is with step-size 0.25 m and turning angle 10°, which follows the same setting established in [8] for comparing with baseline methods in the exploration task. The second one is with step-size 0.30 m and turn-angle 30°. This setting helps us test *DeepExplorer*'s generalization capability under different locomotion configurations.

Navigation setup. In navigation, we encourage the agent to visit enough positions for each room scene. Specifically, the agent has collected 2,000 images per room on Gibson and 5,000 images per room on MP3D (2,000/5,000-step *DeepExplorer*-guided exploration).

Expert demonstration generation. For each room scene, we first sample multiple anchor points across the whole navigable area for each room scene. Then the agent starts at a random anchor point and iteratively walks to the next unvisited closest anchor point with minimal steps (by calling Habitat `PathFollower` API) until all anchor points are traversed. At each step, we record the agent's action and panoramic RGB image. Please refer to Fig. 4 for a visualization of this process. Note that our expert demonstration does not necessarily guarantee globally optimal exploration, which is intentional as this process can be easily automated and scaled.

Training details. The network architectures for both *TaskPlanner* and *MotionPlanner* are given in Table IV and V in Appendix . In our implementation, the local observation sequence length is 10 (m=10) due to its empirical good performance-memory trade-off. We experimentally tested $m = 20$ and got inferior performance. *DeepExplorer* network architecture is illustrated in Appendix (parameter size is just 16 M). We train *DeepExplorer* with PyTorch [60]. The optimizer is Adam [61] with an initial learning rate of 0.0005, but decays every 40 epochs with a decaying rate of 0.5. In total, we train 70 epochs. We train all the *DeepExplorer* variants with the same hyperparameter setting for a fair comparison.

B. Comparison Methods

For exploration task, we compare *DeepExplorer* with six RL-based methods: 1. **RL + 3LConv**: An RL Policy with 3 layer convolutional network [55]; 2. **RL + Res18**: RL Policy initialized with ResNet18 [49] and followed by GRU [62]; 3. **RL + Res18 + AuxDepth**: adapted from [57] which uses depth

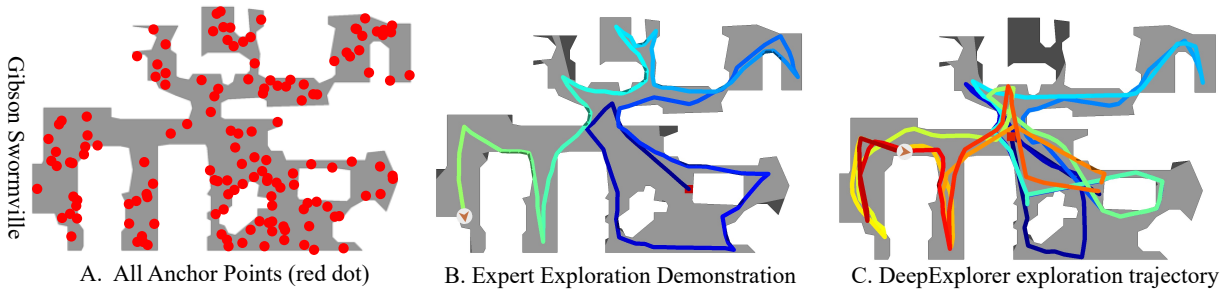


Fig. 4. **Expert Demonstration and *DeepExplorer* exploration visualization**. **A**: all potential anchor points in the Gibson Swormville scene. **B**: expert selectively traverses a subset of the anchor points (by merging spatial close anchor points) by iteratively reaching the next unvisited closest anchor points so as to create expert demonstrations. **C**: *DeepExplorer* exploration trajectory with a 1000-step budget, with the model learned from expert demonstrations.

TABLE II

COVERAGE RATIO OVER 1000-STEP BUDGET. TOP THREE PERFORMANCES ARE HIGHLIGHTED BY **RED**, **GREEN**, AND **BLUE** COLOR, RESPECTIVELY.

Method Description	Method	Sensor Used	#Train Imgs	Gibson Val		Domain Generalization MP3D Test	
				%Cov.	Cov. (m^2)	%Cov.	Cov. (m^2)
Non-learning Based	RandomWalk (used by SPTM [4])	No	No	0.501	22.268	0.301	40.121
RL w/ Metric Input/Estimates	RL + 3LConv [56]	RGB, Depth, Pose	10 M	0.737	22.838	0.332	47.758
	RL+ResNet18		10 M	0.747	23.188	0.341	49.175
	RL+ResNet18+AuxDepth [57]		10 M	0.779	24.467	0.356	51.959
	RL+ResNet18+ProjDepth [46]		10 M	0.789	24.863	0.378	54.775
	OccAnt [58]		1.5-2 M	**0.935**	**31.712**	0.500	71.121
	ANS [8]		10 M	**0.948**	**32.701**	0.521	73.281
DeepExplorer Model Variants	DeepExplorer_NoDeepSup	RGB only	0.45 M	0.768	26.671	0.292	37.163
	DeepExplorer_NoFeatDeepSup			0.912	35.151	0.620	104.499
	DeepExplorer_NoActDeepSup			0.900	33.922	0.600	102.122
	DeepExplorer_LSTMActRegu			0.914	35.238	0.610	101.734
	DeepExplorer_withHistory			0.917	35.331	**0.618**	**102.302**
	DeepExplorer_NoFeatHallu			0.907	34.563	0.589	99.091
Deeply Supervised Imitation	**DeepExplorer**			0.918	35.274	**0.642**	**109.057**
	DeepExplorer (0.30m/30°)			**0.927**	**37.731**	**0.656**	**117.993**

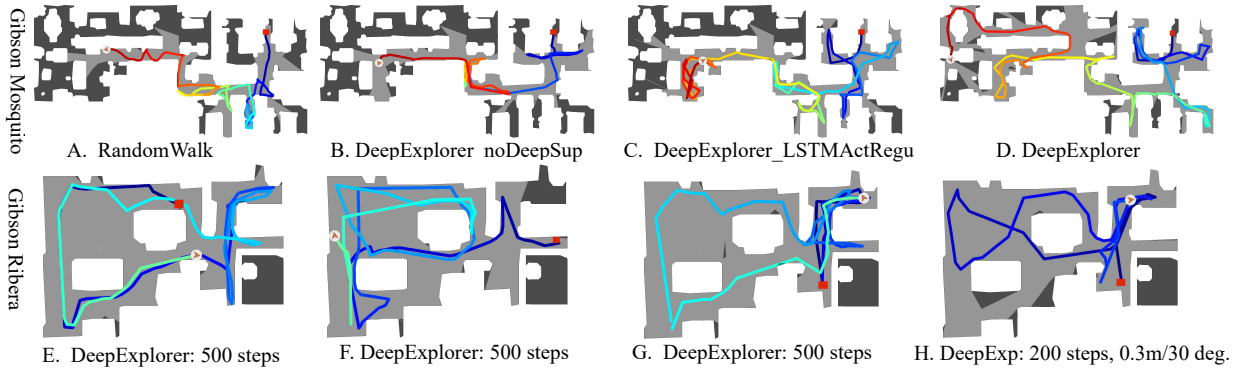


Fig. 5. **Exploration trajectories Visualization**. Top row: various *DeepExplorer* variants exploration result (1000-step budget) on Gibson `Mosquito` scene. Bottom row: exploration with different start positions (E, F, G, 500-step budget, with agent step-size 0.25 m and turn-angle 10°). An agent with larger step size and turn angle (0.3 m/30°) achieves a similar coverage ratio with much smaller steps (200 steps, F). The trajectory color evolving from cold (blue) to warm (yellow) indicates the exploration chronological order.

map prediction as an auxiliary task. The network architecture is the same as ANS [47] with one extra deconvolution layer for depth prediction; 4. **RL + Res18 + ProjDepth** adapted from Chen *et al.* [46] who project the depth image in an egocentric top-down in addition to the RGB image as input to the RL policy. 5. **ANS** (Active Neural SLAM [8]) jointly learns a local and global policy network to guide the agent to explore; 6. **OccAnt** [58]: takes RGB, depth, and camera as inputs to learn a 2D top-down occupancy map to help exploration. For ablation studies, we have following *DeepExplorer* variants:

1) **RandomWalk** The agent randomly chooses an action to execute at each step. It serves as a baseline and helps us to know agent exploration capability without any active learning process. Please note that RandomWalk is also the SPTM [4] exploration strategy.
2) **DeepExplorer_NoDeepSup.** *DeepExplorer* without deeply-supervised learning. We remove LSTM per-step feature supervision in *TaskPlanner* and neighboring frame action supervision in *MotionPlanner*. In other words, we just keep the feature prediction and action classification between the latest step and the future step. It helps to test the necessity of involving a deeply-supervised learning strategy.
3) **DeepExplorer_NoFeatDeepSup.** *DeepExplorer* with-

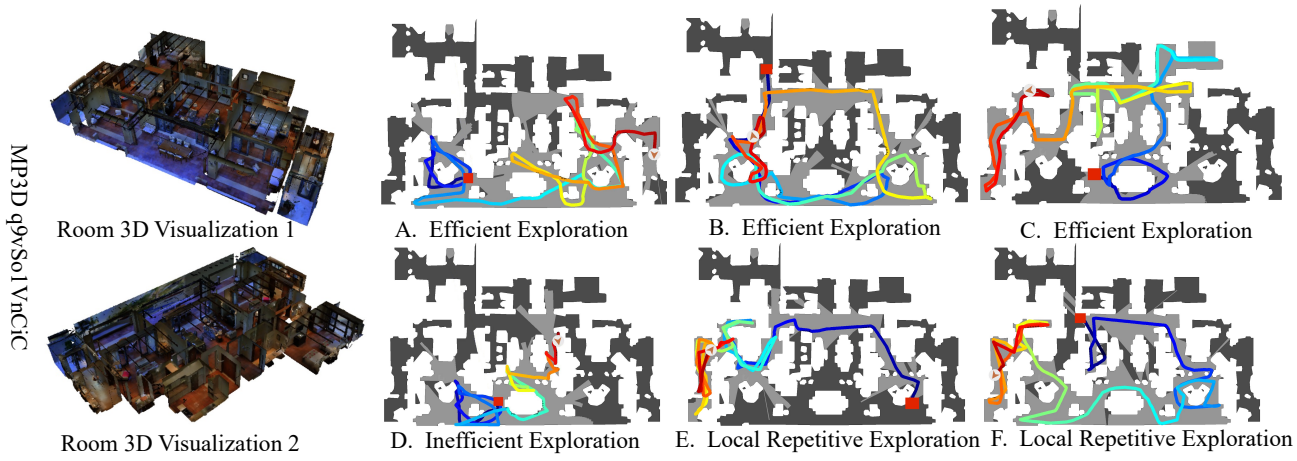


Fig. 6. DeepExplorer exploration result on MP3D [17] room scene `q9vSo1VnCiC`. We show both efficient exploration results (top row, sub-figure A, B, C). We also show relatively less-efficient exploration in sub-figure E and F, which are mainly due to local repetitive exploration. We further show an inefficient exploration example in sub-figure D. Two-room scene 3D visualization is given in the left-most subfigures. The agent exploration starting position is marked by a red rectangle patch.

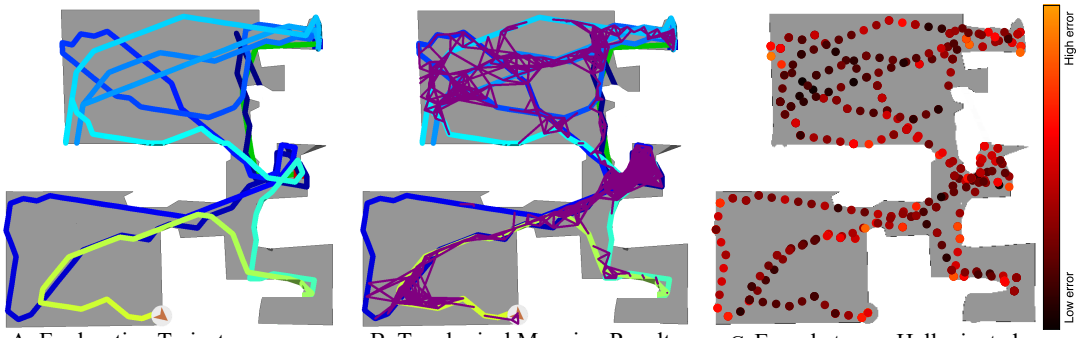
out deeply-supervised learning in the feature space. We remove LSTM per-step feature supervision in *TaskPlanner* but keep the neighboring frame action supervision in *MotionPlanner*. This means no $\mathcal{L}_T$ but only $\mathcal{L}_M$. Together with **DeepExplorer_NoDeepSup** and **DeepExplorer_NoActDeepSup**, it helps to test the necessity of deploying deep supervision in both task and motion planning.

4) **DeepExplorer_NoActDeepSup.** *DeepExplorer* without deeply-supervised learning regarding action prediction. We remove the neighboring frame action supervision in *MotionPlanner* but keep the LSTM per-frame feature supervision in *TaskPlanner*. In other words, there is no $\mathcal{L}_M$ but only $\mathcal{L}_T$. Together with **DeepExplorer_NoDeepSup** and **DeepExplorer_NoFeatDeepSup** It helps to test the necessity of deploying deep supervision in both task and motion planning.
5) **DeepExplorer_LSTMActRegu.** *TaskPlanner* hallucinates the next-best feature at each step to deeply supervise the whole framework in the feature space. As an alternative, we can instead predict action instead at each step in *TaskPlanner*. This DeepExplorer variant helps us to figure out whether supervising each step of *TaskPlanner* LSTM in feature space is helpful.
6) **DeepExplorer_withHistory.** *DeepExplorer* is trained with only a short-memory (the latest m steps observations). To validate the influence of long-term memory, we train a new *DeepExplorer* variant by adding extra historical information: we evenly extract 10 observations among all historically explored observations excluding the latest m steps. After feeding them to ResNet18 [49] to get their embedding, we simply use average pooling to get one 512-dimensional vector and feed it to *TaskPlanner* LSTM as the hidden state input.
7) **DeepExplorer_noFeatHallu.** We use the architecture of *TaskPlanner* to directly predict the next action. It discards task planning in feature space but instead plans directly in action space. Its performance helps us to understand if the hallucinated feature is truly necessary.
8) **DeepExplorer (0.30m/30°).** This variant adopts a different locomotion protocol than the one used in ANS [8] and all other variants to demonstrate *DeepExplorer*'s robustness under different locomotion setups.

Some visualizations of different *DeepExplorer* variants' exploration results can be found in Fig. 5.

C. Evaluation Results on Exploration

The quantitative results of the exploration task are shown in Table II. We can observe from this table that *DeepExplorer* achieves comparable performance on the Gibson dataset with the best RL-based methods and best-performing result on the MP3D dataset by outperforming all RL-based methods significantly (about 13% coverage ratio and $40m^2$ area improvement). Since the comparing RL-based methods [8], [46], [56], [57] build the map in metric space and requires millions of training images, DeepExplorer is desirable because (1) it provides a metric-free option for exploration, and (2) it is lightweight (in terms of parameter size 16 M) and requires much less training data (just about 0.45 million images, in contrast with 10 million images required by most RL-based methods). The room scenes in MP3D dataset are much more complex and larger than those in the Gibson dataset. They contain various physical impediments (e.g., complex layout, furniture), and some rooms contain outdoor scenarios. Hence, **DeepExplorer exhibits stronger zero-shot sim2sim generalizability in exploring novel scenes than RL-based methods**. Moreover, the performance gain is more obvious on both Gibson and MP3D datasets when we change the agent to a different locomotion setup (from 0.25/10° to 0.30/30°), which also shows DeepExplorer is robust to different locomotion setups.



A. Exploration Trajectory B. Topological Mapping Result C. Error between Hallucinated Feat. and Real-Observed Feat.

Fig. 7. **Feature Visualization A**. The exploration trajectory (blue to yellow, step size 0.25 m and turn-angle 30°) with a 500-step budget overlaid on top of the floor plan map. **B**. The spatially-adjacent panoramic images are connected (purple lines) via VPR. **C**. The difference (Euclidean distance in 512-d feature space) between real-observed features and hallucinated features. The darker the color, the lower the difference.

On the Gibson dataset, *DeepExplorer* achieves a slightly lower coverage ratio than ANS [8] but a higher average covered area. We find such performance difference is mainly caused by *DeepExplorer* stronger capability in exploring large areas than RL-based methods. In most cases, *DeepExplorer* actively reaches new areas within limited steps.

Comparison with random exploration. *RandomWalk* serves as the baseline for our framework. It is also adopted by SPTM [4] to build a topological map. It involves no learning procedure, and the agent randomly takes action at each step to explore an environment. From Table II, we can see that *RandomWalk* dramatically reduces the exploration performance in terms of both coverage ratio and average coverage area. The inferior performance of *RandomWalk* verifies the necessity of learning active exploration strategy in order to help the agent efficiently explore an environment. Figure 5 demonstrates the qualitative comparison between *RandomWalk* and *DeepExplorer* exploration result.

Feature regularization and with history memory. If we replace feature regularization involved in *TaskPlanner* with action regularization (*DeepExplorer_LSTMActRegu*), we have observed more performance drop on MP3D than on Gibson dataset (3% versus 0.2%), which shows adopting feature regularization improves the generalizability compared with action regularization. Moreover, introducing full history memory (*DeepExplorer_FullHistory*) to *TaskPlanner* (used as LSTM hidden state input) produces very similar results on the Gibson dataset, but significantly reduces the performance on MP3D dataset (more than 2% drop). It thus shows using historical memory tends to encourage *DeepExplorer* to overfit training data so that its generalizability is inevitably reduced. We argue that such generalizability drop might lie in our over-simplified history memory modeling because we just evenly sample 10 nodes (image observations) from all historically visited nodes, which might be too simple to represent the whole history memory, or even confuses *TaskPlanner* if the agent has already explored many steps. A more elegant long-term history memory model remains to be explored.

Deeply-supervised learning and joint task and motion imitation. Removing deeply-supervised learning (*DeepExplorer_noDeepSup*, *DeepExplorer_noFeatDeepSup*, *DeepExplorer_noActDeepSup*) leads to performance drop on both Gibson and MP3D dataset, especially when both deep supervisions in *TaskPlanner* and *MotionPlanner* are both dropped. In the MP3D dataset, it can even lead to worse performance than *RandomWalk*. It thus shows the necessity of deep supervision in both feature space (*TaskPlanner*) and action space (*MotionPlanner*). Meanwhile, *DeepExplorer_noFeatHallu* leads to a significant performance drop on both Gibson and MP3D datasets. It thus attests to the advantage of our feature-space task and motion imitation strategy which jointly optimize *TaskPlanner* for high-level task allocation and *MotionPlanner* for low-level motion control.

We also visualize the comparison between *DeepExplorer* hallucinated next-step future feature and truly observed feature in Fig. 7 (C). We see that the hallucinated feature is more similar to the observed real feature when the agent is walking through a spacious area (in other words, the agent mostly takes `move_forward` action) than when the agent is walking along a room corner, against the wall or through a narrow pathway. This may be due to the learned *TaskPlanner* most likely hallucinates feature moving the agent forward if the temporary egocentric environment allows. This also matches expert exploration experience because experts mostly prefer moving forward so as to explore as many areas as possible.

D. Evaluation Results on Navigation

For the visual navigation task, we compare *DeepExplorer* with most of the methods compared in the exploration task. CMP [63] builds up a top-down belief map for joint planning and mapping. For OccAnt [58], we just report its result with the model trained with RGB image (so as to be directly comparable with *DeepExplorer*). For SPTM [4], we train all its navigation-relevant models on data obtained by *DeepExplorer*. The navigation result is given in Table III. We can see that *DeepExplorer* outperforms all comparing methods on the two datasets, with the largest performance gain on the MP3D dataset (about 14% Succ. Rate, 12% SPL improvement). Hence, we can see that our *DeepExplorer*-built topological

TABLE III

NAVIGATION RESULT. TOP THREE PERFORMANCES ARE HIGHLIGHTED IN **RED**, **GREEN**, AND **BLUE** COLOR, RESPECTIVELY. WE COLLECT 2000 (GIBSON)/5000(MP3D) IMAGES WITH AGENT SETUP (0.25M/10°). 'N/A' MEANS 'NOT AVAILABLE.'

Method	Gibson Val		Domain Generalization on MP3D Testset	
	Succ. Rate (↑)	SPL (↑)	Succ. Rate (↑)	SPL (↑)
RandomWalk	0.027	0.021	0.010	0.010
RL + Blind	0.625	0.421	0.136	0.087
RL + 3LConv + GRU [56]	0.550	0.406	0.102	0.080
RL + ResNet18 + GRU	0.561	0.422	0.160	0.125
RL + ResNet18 + GRU + AuxDepth [57]	0.640	0.461	0.189	0.143
RL + ResNet18 + GRU + ProjDepth [46]	0.614	0.436	0.134	0.111
IL + ResNet18 + GRU	0.823	0.725	**0.365**	**0.318**
SPTM [4]	0.510	0.381	0.240	0.203
CMP [63]	0.827	**0.730**	0.320	0.270
OccAnt (RGB) [58]	**0.882**	0.712	N/A	N/A
ANS [8]	**0.951**	**0.848**	**0.593**	**0.496**
DeepExplorer	**0.957**	**0.859**	**0.733**	**0.619**

map can be used for image-goal-based visual navigation. More importantly, *DeepExplorer* shows satisfactory zero-shot sim2sim generalizability in navigation as well. In Fig. 7 (B), we can see VPR and *ActionAssigner* successfully add new edges (purple lines) for loop closing. The resulting topological map, after topological mapping, fully reflects environment connectivity and traversability.

E. Zero-Shot Sim2Real Real-World Exploration

DeepExplorer is deployed and verified on a customized real-world robot. We set an Insta360 Pro 2 camera[1] on an iRobot Create 2 robot[2] (the camera height is around 1.5 m). Nvidia Jetson TX2[3] platform is used to launch the *DeepExplorer* model and control the robot. We directly deploy the model (with step size $0.25m$ and turn-angle $10°$) trained on the Gibson simulation dataset without any fine-tuning on the real-world dataset. The robot's physical configuration is as close to that of the simulation as possible. We adopt a LiDAR scanner for obstacle avoidance. The experiment environment is a large indoor multi-functional office building.

We find that *DeepExplorer* demonstrates strong zero-shot sim2real exploration results: the robot is capable of identifying obstacles and actively reaching the open navigable areas. The robot can traverse the entire hallway and enter the only open door (marked with a star in Fig. 1) and manage to exit it through the door after exploring it. The exploration trajectory is shown in Fig. 1 and Fig. 9 (in Appendix). The corresponding video can be found on the Github repository.

F. Limitations

In our experiment on the simulation datasets, we find that *DeepExplorer* sometimes leads to inefficient exploration in complex room environments as is shown in Fig. 6 (bottom row), especially when the room layout is sophisticated and the navigable area is narrow. We hypothesize that this is partly due to the lack of full history memory of *DeepExplorer* that can steer the agent away from already-covered areas. Although we have tried one simple history memory mechanism (DeepExplorer_withHistory), it still remains a future research topic to design a better history memory framework.

In the zero-shot sim2real exploration experiment, we find that the agent often mistakes large glass walls for open doorways. We speculate that the lack of relevant data in the Gibson training dataset, which is entirely made of the household environment, leads to this failure. Extra measurement should be considered to handle such cases.

[1] https://www.insta360.com/cn/product/insta360-pro2

[2] https://edu.irobot.com/what-we-offer/create-robot

[3] https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-tx2/

V. CONCLUSION

Our proposed *DeepExplorer* is capable of efficiently building a topological map by metric-free exploration to represent an environment. It entirely works in an image feature space to explore a new environment by jointly hallucinating the next step feature and predicting the appropriate action that best moves the agent to the feature. It is simple and lightweight as it just requires RGB images and the model size is small. It is trained via deeply-supervised imitation learning where the expert demonstration is easy to acquire and scale up. We show its strong zero-shot sim2sim and sim2real generalization capability by experiments in both large-scale and photo-realistic simulation environments and real-world environments. Future works include designing more elaborate historic memory modules and involving multi-modality sensors to further improve the performance of visual exploration and navigation.

REFERENCES

[1] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 2016. 1

[2] Benjamin Kuipers and Yung-Tai Byun. A Robot Exploration and Mapping Strategy based on a Semantic Hierarchy of Spatial Representations. *Robotics and Autonomous Systems*, 1991. 1

[3] David Kortenkamp and Terry Waymouth. Topological Mapping for Mobile Robots Using a Combination of Sonar and Vision Sensing. In *Proceedings of the twelfth national conference on Artificial Intelligence (AAAI)*, 1994. 1

[4] Savinov Nikolay, Dosovitskiy Alexey, and Koltun Vladlen. Semi-Parametric topological memory for navigation. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2, 4, 6, 8, 9

[5] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural Topological SLAM for Visual Navigation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[6] Sabine Gillner and Hanspeter A. Mallot. Navigation and Acquisition of Spatial Knowledge in a Virtual Maze. *Journal of Cognitive Neuroscience*, 1998. 1

[7] Obin Kwon, Nuri Kim, Yunho Choi, Hwiyeon Yoo, Jeongho Park, and Songhwai Oh. Visual Graph Memory With Unsupervised Representation for Visual Navigation. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 1, 2

[8] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning To Explore Using Active Neural SLAM. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 5, 6, 7, 8, 9, 12

[9] Nuri Kim, Obin Kwon, Hwiyeon Yoo, Yunho Choi, Jeongho Park, and Songhawi Oh. Topological Semantic Graph Memory for Image Goal Navigation. In *Annual Conference on Robot Learning (CoRL)*, 2022. 1, 2

[10] Cindy Leung, Shoudong Huang, and Gamini Dissanayake. Active SLAM Using Model Predictive Control and Attractor based Exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5031. IEEE, 2006. 1

[11] Stephane Ross and Drew Bagnell. Efficient Reductions for Imitation Learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010. 1, 3

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997. 1, 4, 12

[13] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015. 2, 4

[14] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating Local Descriptors into a Compact Image Representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 2, 4

[15] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. Visual Place Recognition: A Survey. *IEEE Transactions on Robotics*, 2016. 2

[16] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: Real-World Perception for Embodied Agents. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 5, 12

[17] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D Data in Indoor Environments. *International Conference on 3D Vision (3DV)*, 2017. 2, 5, 7

[18] Edward C Tolman. Cognitive Maps in Rats and Men. *Psychological review*, 1948. 2

[19] Fan Yang, Dung-Han Lee, John Keller, and Sebastian Scherer. Graph-based Topological Exploration Planning in Large-scale 3D Environments. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 2

[20] Liz Murphy and Paul Newman. Using Incomplete Online Metric Maps for Topological Exploration with the Gap Navigation Tree. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2008. 2

[21] Edward Beeching, Jilles Dibangoye, Olivier Simonin, and Christian Wolf. Learning to Plan with Uncertain Topological Maps. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[22] Zhaoliang Zhang, Jincheng Yu, Jiahao Tang, Yuanfan Xu, and Yu Wang. MR-TopoMap: Multi-Robot Exploration Based on Topological Map in Communication Restricted Environment. *IEEE Robotics and Automation Letters*, 2022. 2

[23] Kevin Chen, Juan Pablo de Vicente, Gabriel Sepulveda, Fei Xia, Alvaro Soto, Marynel Vazquez, and Silvio Savarese. A Behavioral Approach to Visual Navigation with Graph Localization Networks. In *Proceedings of Robotics: Science and Systems*, 2019. 2

[24] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating Visual Representations from Unlabeled Video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[25] K. Zeng, W. B. Shen, D. Huang, M. Sun, and J. Niebles. Visual Forecasting by Imitating Dynamics in Natural Sequences. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[26] Chien-Yi Chang, De-An Huang, Danfei Xu, Ehsan Adeli, Li Fei-Fei, and Juan Carlos Niebles. Procedure Planning in Instructional Videos. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[27] B. Fernando and S. Herath. Anticipating Human Actions by Correlating Past with the Future with Jaccard Similarity Measures. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[28] Dídac Surís, Ruoshi Liu, and Carl Vondrick. Learning the predictability of the future. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[29] Ashesh Jain, Avi Singh, Hema Swetha Koppula, Shane Soh, and Ashutosh Saxena. Recurrent Neural Networks for Driver Activity Anticipation via Sensory-Fusion Architecture. *IEEE International Conference on Robotics and Automation (ICRA)*, 2016. 2

[30] Hema S. Koppula and Ashutosh Saxena. Anticipating Human Activities Using Object Affordances for Reactive Robotic Response. *IEEE Transaction on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016. 2

[31] Luca Carlone and Sertac Karaman. Attention and Anticipation in Fast Visual-Inertial Navigation. *Transaction on Robotics*, 2019. 2

[32] Hyun Soo Park, Jyh-Jing Hwang, Yedong Niu, and Jianbo Shi. Egocentric Future Localization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[33] Martina Lippi, Petra Poklukar, Michael C. Welle, Anastasiia Varava, Hang Yin, Alessandro Marino, and Danica Kragic. Latent Space Roadmap for Visual Action Planning of Deformable and Rigid Object Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. 2

[34] Brian Ichter and Marco Pavone. Robot Motion Planning in Learned Latent Spaces. *IEEE Robotics and Automation Letters*, 2019. 2

[35] Dawei Sun, Anbang Yao, Aojun Zhou, and Hao Zhao. Deeply-Supervised Knowledge Synergy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[36] Chi Li, M Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D Hager, and Manmohan Chandraker. Deep Supervision with Shape Concepts for Occlusion-Aware 3D Object Parsing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[37] Chi Li, M Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D Hager, and Manmohan Chandraker. Deep Supervision with Intermediate Concepts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018. 2

[38] Rohan Chitnis, Dylan Hadfield-Menell, Abhishek Gupta, Siddharth Srivastava, Edward Groshev, Christopher Lin, and Pieter Abbeel. Guided Search for Task and Motion Plans Using Learned Heuristics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016. 3

[39] Michael James McDonald and Dylan Hadfield-Menell. Guided Imitation of Task and Motion Planning. In *Annual Conference on Robot Learning (CoRL)*, pages 630–640. PMLR, 2022. 3

[40] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments. In *Proceedings of Robotics: Science and Systems*, 2021. 3

[41] Shih-Yun Lo, Shiqi Zhang, and Peter Stone. PETLON: Planning Efficiently for Task-Level-Optimal Navigation. In *Proceedings of International Conference on Autonomous Agents and MultiAgent Systems*, 2018. 3

[42] Antony Thomas, Fulvio Mastrogiovanni, and Marco Baglietto. MPTP: Motion-Planning-Aware Task Planning for Navigation in Belief Space. *Robotics and Autonomous Systems*, 2021. 3

[43] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining Optimal Control and Learning for Visual Navigation in Novel Environments. In *Annual Conference on Robot Learning (CoRL)*, 2019. 3, 4

[44] Xinlei Pan, Tingnan Zhang, Brian Ichter, Aleksandra Faust, Jie Tan, and Sehoon Ha. Zero-shot Imitation Learning from Demonstrations for Legged Robot Visual Navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 3

[45] Sha Luo, Hamidreza Kasaei, and Lambert Schomaker. Self-Imitation Learning by Planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 3

[46] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning Exploration Policies for Navigation. In *International Conference on Learning Representations (ICLR)*, 2019. 3, 5, 6, 7, 9

[47] Cindy Leung, Shoudong Huang, and Gamini Dissanayake. Active SLAM in Structured Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2008. 3, 6

[48] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large

Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015. 4

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4, 5, 7, 12

[50] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017. 4

[51] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal on Computer Vision (IJCV)*, 2004. 4

[52] S. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. 4

[53] Ting Liu, Andrew W. Moore, and Alexander Gray. New Algorithms for Efficient High-Dimensional Nonparametric Classification. *Journal of Machine Learning Research*, 2006. 4

[54] Edsger W Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 1959. 5

[55] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *International Conference on Computer Vision (ICCV)*, 2019. 5

[56] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On Evaluation of Embodied Navigation Agents. *CoRR*, 2018. 5, 6, 7, 9

[57] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to Navigate in Complex Environments. In *International Conference on Learning Representations (ICLR)*, 2017. 5, 6, 7, 9

[58] Ziad Al-Halah Santhosh Kumar Ramakrishnan and Kristen Grauman. Occupancy Anticipation for Efficient Exploration and Navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 6, 8, 9

[59] Santhosh K. Ramakrishnan, Dinesh Jayaraman, and Kristen Grauman. An Exploration of Embodied Visual Exploration. In *International Journal of Computer Vision (IJCV)*, 2021. 5

[60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 5

[61] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[62] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014. 5

[63] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive Mapping and Planning for Visual Navigation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 8, 9

APPENDIX

A. DeepExplorer Neural Network Architecture

DeepExplorer neural network architecture is given in Table IV, it consists of ResNet18 (for image observation embedding), LSTM layer [12] (for *TaskPlanner*) and multi-layer perceptron (MLP) (for *MotionPlanner*). Please note that *DeepExplorer* is lightweight, its parameter size is just 16 M (where M is million).

TABLE IV
DEEPEXPLORER NETWORK ARCHITECTURE ILLUSTRATION. THE NETWORK CONSISTS OF BASIC 2D IMAGE CONVOLUTION LAYERS, SUCH AS RESNET18, LSTM, AND FC. THE NETWORK IS LIGHTWEIGHT, THE PARAMETER SIZE IS JUST 16 M.

Layer Name	Filter Num	Output Size
Image Embedding Layer		
Input: [10, 3, 256, 512] Embedding Network: ResNet18 Embedding Size: [10, 512]		
Task Planner Network		
LSTM Feat Prediction FC	layers = 2, hidden size = 512 in feat = 512, out feat = 512	[10, 512] [10, 512]
Motion Planner Network		
Input: Feat [10, 1024], Action: [10]		
Feat Merge FC Action Classification FC	in feat = 1024, out feat = 512 in feat = 512, out feat = 3	[10, 512] [10, 3]

TABLE V
ActionAssigner NEURAL NETWORK ARCHITECTURE. THE WHOLE PARAMETER SIZE IS 13.5 M.

Layer Name	Filter Num	Output Size
Image Embedding Layer		
Input: [2, 3, 256, 512] Embedding Network: ResNet18		
Feat. Merge Layer		
Concat. Size: [1, 1024]		
FC	in feat = 1024, out feat = 512	[1, 512]
Action Predict Branch		
head1 FC head2 FC head3 FC head4 FC head5 FC head6 FC	in feat = 512, out feat = 128 in feat = 512, out feat = 128 in feat = 512, out feat = 128 in feat = 512, out feat = 128 in feat = 512, out feat = 128 in feat = 512, out feat = 128	[1, 128] [1, 128] [1, 128] [1, 128] [1, 128] [1, 128]
Action Predict		
Concat. Size: [1, 6, 128]		
BiLSTM Action Classify FC	layers = 1, out feat = 128 in feat = 128, out feat = 3	[1, 6, 128] [1, 6, 3]

B. ActionAssigner Network Architecture

The *ActionAssigner* neural network is given in Table V. The *ActionAssigner* also uses ResNet18 [49] as the image embedding module. Then it uses a sequence of multi-layer perceptron (MLP) to predict multi-step actions separately (step length is 6), each step independently predicts one action. So *ActionAssigner* is a multi-label classification neural network. Bidirectional LSTM is applied to model mutual action dependency among different steps. The parameter size is 13.5 M. We train *ActionAssigner* with the same parameter setting as of *TaskPlanner* and *MotionPlanner* (the network in Table IV). During training data preparation, if the action list length is smaller than 6, we pad `STOP` action to fill the length.

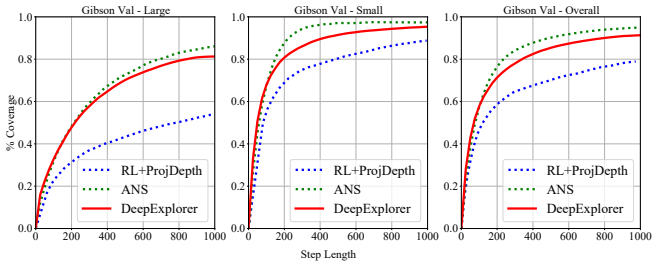


Fig. 8. Coverage ratio variation curve comparison over 1000-step budget over large $> 50m^2$, small $< 50m^2$ and all (average) room size, respectively.

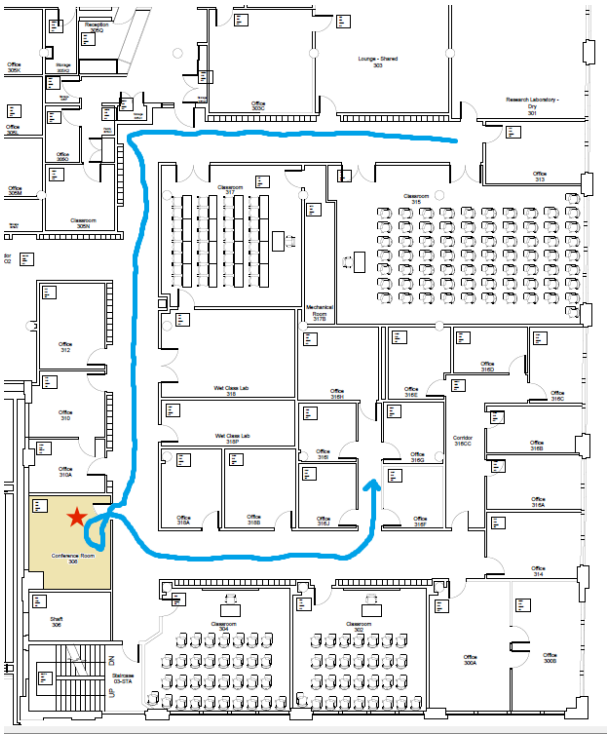


Fig. 9. DeepExplorer exploration trajectory on zero-shot sim2real experiment. We can see that the agent can successfully find the navigable area along the corridor to efficiently explore more areas. It can also manage to enter and exit one conference room (the area marked with a red star).

C. Coverage Ratio Progression Comparison

We further provide the coverage ratio progression variation w.r.t. exploring steps comparison between *DeepExplorer* and ANS [8], one RL-based method (RL+ProjDepth) in Fig. 8. The comparison is based on Gibson validation dataset [16], and we divide the room into *Large*, *Small*, and *Overall* according to the room size. From this table, we can see that *DeepExplorer* is capable of covering more area during the first 200 steps than ANS [8] on large rooms (which is verified by the more steep curve of *DeepExplorer* over ANS [8] and RL+ProjDepth, in the middle sub-figure).

D. Zero-Shot Sim2Real Exploration Discussion

We provide two exploration videos in the supplementary folder. One video demonstrates the successful exploration (with the exploration trajectory shown in Fig. 9), and

the other video shows one unsuccessful exploration case in which the agent mixes the glass walls with an open area.

The agent we used for real-world exploration contains large actuation noise, so the actual angle it has turned may be different from our configuration ($10°$). Sometimes its actual executed turn angle can be as large as $20\circ$ or even $30°$, especially in the conference room where the floor is overlaid with carpet (yellow color and red star marked area in Fig. 9). The existence of actuation noise explains the non-smoothness between adjacent frames in our provided video, especially when the agent entered the conference room.

Although the actuation noise is caused by the agent, we find our trained *DeepExplorer* model can predict the appropriate actions to mitigate the actuation noise impact. For example, when the agent has turned a larger angle than the configuration (e.g. turn left), *DeepExplorer* can predict a contrary action (e.g. turn right) to the correct agent.