2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

LITE: A Learning-Integrated Topological Explorer for Multi-Floor Indoor Environments

Junhao Chen[1], Zhen Zhang[1], Chengrui Zhu[1], Xiaojun Hou[1], Tianyang Hu[1], Huifeng Wu[2], Yong Liu[1,*]

***Abstract*— This work focuses on multi-floor indoor exploration, which remains an open area of research. Compared to traditional methods, recent learning-based explorers have demonstrated significant potential due to their robust environmental learning and modeling capabilities, but most are restricted to 2D environments. In this paper, we proposed a learning-integrated topological explorer, LITE, for multi-floor indoor environments. LITE decomposes the environment into a floor-stair topology, enabling seamless integration of learning or non-learning-based 2D exploration methods for 3D exploration. As we incrementally build floor-stair topology in exploration using YOLO11-based instance segmentation model, the agent can transition between floors through a finite state machine. Additionally, we implement an attention-based 2D exploration policy that utilizes an attention mechanism to capture spatial dependencies between different regions, thereby determining the next global goal for more efficient exploration. Extensive comparison and ablation studies conducted on the HM3D and MP3D datasets demonstrate that our proposed 2D exploration policy significantly outperforms all baseline explorers in terms of exploration efficiency. Furthermore, experiments in several 3D multi-floor environments indicate that our framework is compatible with various 2D exploration methods, facilitating effective multi-floor indoor exploration. Finally, we validate our method in the real world with a quadruped robot, highlighting its strong generalization capabilities.**

I. INTRODUCTION

Autonomous exploration is a fundamental problem in the development of embodied intelligence and plays a crucial role in uncertain scenarios such as search and rescue [1], scene reconstruction [2], and extraterrestrial planetary exploration [3]. An exploration task requires an autonomous agent to make long-term decisions and efficiently construct an environment representation without prior information. This task remains particularly challenging in indoor environments, which are often cluttered and contain multiple floors, making autonomous exploration difficult for robots.

In traditional exploration paradigms [4]–[6], autonomous exploration is achieved by iterative online viewpoint sampling and path generation. During each iteration, the identified frontiers or sampled viewpoints are evaluated by a cost function to select the most informative one for exploration [7] [8]. However, viewpoints determined by manually calculated cost functions often lead to locally optimal but short-sighted decisions. Consequently, the robot may exhibit zigzag movements and leave certain areas unexplored, *e.g.*, narrow corridors or small rooms, rendering the exploration procedure inefficient.

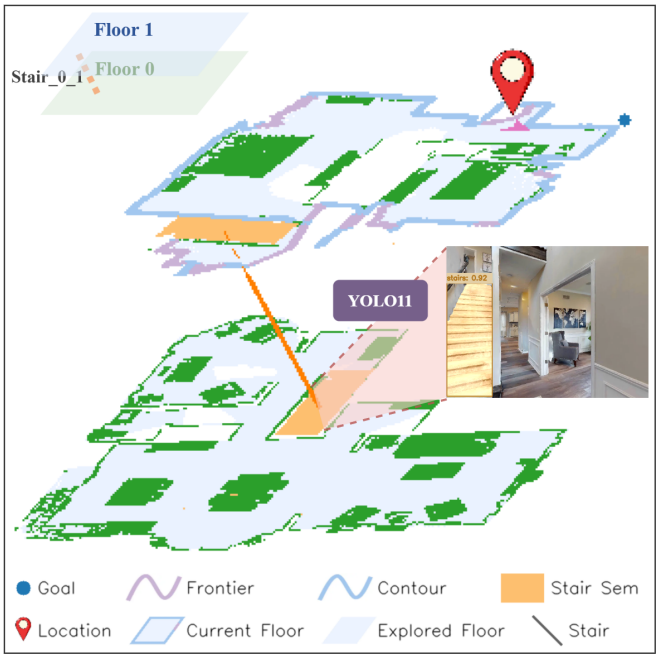

Fig. 1: Illustration of LITE. Right: In single-floor exploration, a YOLO11 segmentation model is applied to detect stairs. Left: The stair information is utilized to build floor-stair topology incrementally, which allows transition between multiple floors. Middle: LITE starts exploration from the first floor using an attention-based 2D exploration policy and then guides the agent upstairs to achieve multi-floor exploration.

Recently, several learning-based methods, particularly those using deep reinforcement learning, have been proposed to achieve efficient and non-myopic exploration [9]–[11]. These methods learn from extensive interactions with the environment, enabling the trained policy network to extract features from observations and make sequential decisions without exhibiting myopic behavior. However, current learning-based approaches are still mostly restricted to 2D environments to simplify exploration modeling, yet they fail to address the challenges of multi-floor indoor environments.

In this work, we propose a novel **L**earning-**I**ntergrated **T**opological **E**xplorer, named **LITE**, to learn an efficient attention-based 2D exploration policy and integrate it into multi-floor indoor environments (see Fig. 1). LITE decomposes the multi-floor environments into a topological graph, where each node represents a floor, and each edge corresponds to a stair connecting two floors. Our key insight is

∗ Corresponding author. E-mail: yongliu@iipc.zju.edu.cn
[1] are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China.
[2] is with the Hangzhou Dianzi University, Hangzhou 310018, China.
This work is supposed by Natural Science Foundation U21A20484.

that the information in indoor 3D environments is dense within a floor but sparse between floors. Consequently, a floor-stair topological representation is sufficient and *lite* for exploration and downstream tasks. Furthermore, it can be easily integrated with both learning and non-learning methods. LITE utilizes a finite state machine (FSM) to manage the exploration process within the floor-stair topology, dividing it into four distinct states, while a YOLO11-based [12] instance segmentation model is used for real-time stair prediction and topology building. This design facilitates the integration of various 2D exploration policies for exploring multi-floor indoor environments. In this work, we first focus on enhancing sustained efficiency of our attention-based 2D exploration policy. This policy captures spatial information using a vision transformer, generating near-optimal goals that direct the agent toward high-information regions. We then integrate this 2D exploration policy into the floor-stair topology, enabling multi-floor indoor exploration without requiring high-dimensional feature inputs.

Our contributions are summarized as follows:

- We propose a novel method called **LITE** to explore indoor multi-floor environments using floor-stair topology. We develop a new paradigm for seamlessly integrating different 2D exploration methods into 3D exploration, leveraging the advantages of learning-based approaches.
- We design an attention-based 2D exploration policy in LITE to select global navigation goals effectively. Extensive experiments demonstrate that our algorithm achieves state-of-the-art 2D exploration performance.
- We validate our LITE on a quadruped robot in the real world, showing its strong generalizability of our method.

II. Related Works

A. Traditional Exploration methods

Traditional exploration methods often begin with Simultaneous Localization and Mapping (SLAM), followed by selecting navigation viewpoints for further exploration. Frontier-based exploration, introduced by Yamauchi in [4], identifies frontiers (*i.e.*, the points between explored and unexplored areas) in the partially built map and greedily selects the closest one as the next goal. Improvements include cost functions that balance gain and distance [7], [8] or multi-resolution maps [13] to improve exploration efficiency. In contrast, sampling-based methods [5], [14], [15] maintain a Rapid-Exploration Random Tree (RRT) to sample stochastic viewpoints and select the best viewpoint with the highest gain. However, the process is usually computationally intensive due to extensive samplings and gain computation, and has a limited capacity for environmental comprehension.

B. Learning-based methods

Recently, deep learning has become a promising approach in autonomous robot exploration. While frontier-based and sampling-based methods are inherently short-sighted, deep learning methods enable the modeling of complex patterns from massive data, leading to more robust and adaptive exploration strategies. Different learning-based exploration approaches include graph neural networks [16], generative models [11], and imitation learning [17], with reinforcement learning being the most prevalent [18], [19]. For instance, [1] integrates deep reinforcement learning with frontier-based methods for optimal frontier selection. Neural-SLAM [9] utilizes an end-to-end neural network to directly generate an action for the agent, though the training process is challenging due to simultaneous optimization. ARiADNE [10] exploits an attention-based network to select goals in a topological graph, which avoids the agent's myopic action. Subsequent research [20] in large-scale environments once again emphasizes the high effectiveness of attention-based methods in capturing long-range spatial dependencies. However, no one has integrated these approaches into 3D exploration tasks. Intuitively, extending reinforcement learning observations to the 3D domain introduces excessive complexity, so it is essential to decompose 3D spaces to lower dimensions for effective learning.

C. Exploration in multi-floor indoor environments

Exploration in multi-floor indoor environments remains challenging, as the robot must consider both horizontal and vertical regions simultaneously. Approaches designed for 3D spaces like RH-NBV [5], GBP [6], and A-TARE [21] are capable of handling such tasks but must be equipped with aerial robots, which are not suitable for indoor environments. In the navigation field, [22] designs a multi-level navigator for robot delivery, which uses a floor topology to guide multi-floor navigation. We note that the transition between floors does not require a highly accurate 3D map, as they are typically connected only by elevators or stairs. Consequently, we model the multi-floor indoor environment as a floor-stair topology to leverage the advantages of 2D reinforcement learning algorithms for exploration on each floor. To the best of our knowledge, we are the first to integrate reinforcement learning-based exploration algorithms into multi-floor indoor exploration using a topological approach.

III. Task Formulation

Consider an autonomous robot placed in an unknown and bounded 2D environment represented by a grid map $\mathcal{Q} \subset \mathbb{R}^2$. The partial map $\mathcal{M} \subset \mathcal{Q}$ is incrementally constructed during exploration using sensor data, comprising free area $\mathcal{M}_f$, occupied area $\mathcal{M}_o$ and unexplored area $\mathcal{M}_u$. When exploration begins, the covered area is denoted as $\mathcal{A} = \mathcal{M}_f \cup \mathcal{M}_o$. We follow the task setup in [9] to define the 2D exploration problem as follows:

Problem 1: Given an unknown 2D environment and a time budget $\mathcal{T}$, find the optimal goal selection policy π_{θ^*} to maximize the covered area $\mathcal{A}$ within $\mathcal{T}$:

$$\theta^* = \arg\max_{\theta} \sum_{t=0}^{\mathcal{T}} \mathcal{A}(a_t|s_t, \pi_\theta), \tag{1}$$

where a_t and s_t are the action and state at the time $t \leq \mathcal{T}$, θ^* is the parameters of the optimal policy neural network. For a multi-floor indoor environment, we expect the robot to

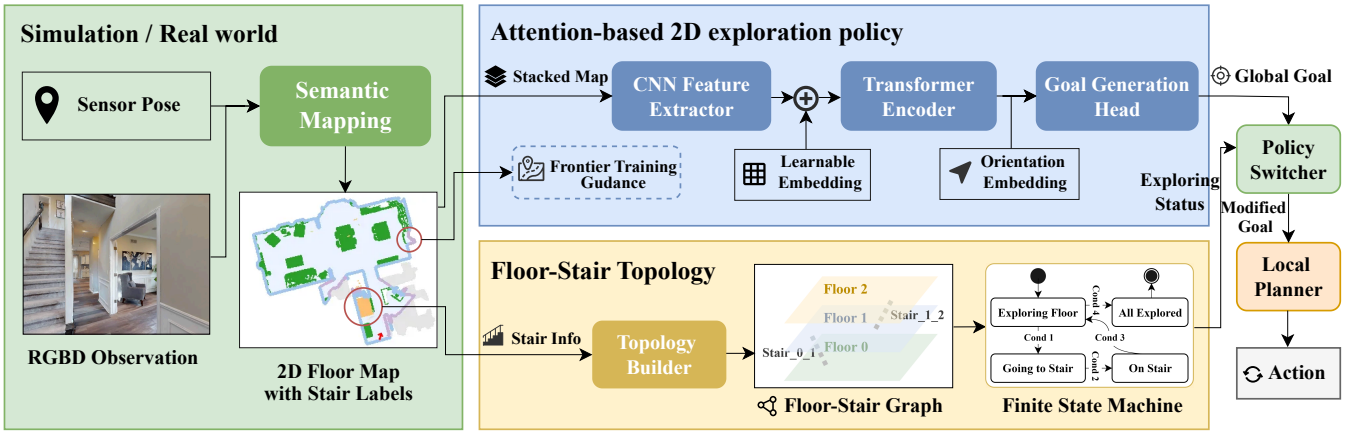

Fig. 2: The overall architecture of LITE. The semantic mapping module utilizes RGB-D observations to generate the current floor map with stair labels. Our attention-based 2D exploration policy then extracts long-range features and determines the next informative global goal for single-floor exploration. Simultaneously, stair information is employed to build a floor-stair topology and select an exploration status within a finite state machine, allowing the agent to transition between multi-floors.

cover areas $\mathcal{A}_i$ on all floors given time budget $\mathcal{T}$ using our floor-stair topology and 2D exploration policy.

Problem 2: Given an unknown multi-floor indoor environment and a time budget $\mathcal{T}$, find a floor-stair topology $\mathcal{G}$ to cover the areas $\mathcal{A}_i$ of all floors using the optimal 2D exploration policy θ^* from *Problem 1*.

IV. Methodology

In this section, we outline the overall architecture of LITE and provide a detailed methodology for each module.

A. Overall Architecture

The overall architecture is illustrated in Fig. 2. Firstly, the semantic mapping module utilizes RGB-D observations to create the current floor map, including stair semantic labels generated by YOLO11 [12] model. Secondly, our attention-based 2D exploration policy employs a transformer encoder to extract long-range features and determine the next global goal for single-floor exploration. The policy leverages frontiers from the floor map and additional embedding information during training to efficiently identify informative areas and minimize redundant exploration. Concurrently, a floor-stair topology is built incrementally to decompose the multi-floor environment and select an exploration state within a finite state machine, allowing the agent to transition between different floors. Once the global goal and exploration status are determined, the policy switcher and local planner module directs the agent's subsequent actions.

B. Map Representation

We represent each floor as a 2D grid map with additional stair semantic information. On the one hand, we use a geometric method to generate self-centered point clouds from depth images and camera poses, which are then flattened into a 2D local grid map. The grid has two channels, $2 \times L \times L$, representing the obstacle map and the explored map. By applying pose transformations over time, we merge local maps into a global grid map with dimensions of $2 \times M \times M$. Here, L and M denote the width or height of grid map, which contains $L \times L$ or $M \times M$ grid cells at a certain resolution. On the other hand, we employ a semantic mapping module similar to [23] to generate an additional stair information map. Specifically, we train a fine-tuned YOLO11 model, leveraging self-labeled stair datasets, to perform instance segmentation of stairs and generate corresponding semantic masks. With the stair masks, the corresponding point clouds are marked as stair spaces and projected into a stair semantic map with the same dimensions of $1 \times M \times M$.

C. Attention-based 2D Exploration Policy

To achieve efficient exploration on each floor, one significant component of our LITE is an attention-based 2D exploration policy using Proximal Policy Optimization (PPO) algorithm [24]. This policy, denoted as π, receives map observations and determines the next best goal as an action to maximize exploration rewards.

1) Observation: To understand the environment comprehensively, we preprocess the 2D grid map, similar to [9], by combining current and historical poses into both local and global maps, expanding their dimensions to $4 \times L \times L$ and $4 \times G \times G$, respectively. We then apply max pooling to the global map to match the local map's size, resulting in a final stacked map of $8 \times L \times L$. This stacked map $m_t \in [0,1]^{8 \times L \times L}$ serves as the policy's observation o_t at time t. We avoid using raw RGB-D images as observation since it hinders network convergence and exacerbates the sim-to-real gap.

2) Action: We define the action of attention-based policy as the global goal (x, y) within the local map $L \times L$ at time t: $a_t = (x, y) \sim \pi_\theta(a_t \mid o_t)$. Here, θ represents the policy network's parameters. This goal is referred to as a global goal, while a local planner is employed to navigate towards it. We utilize the Fast March Method [25] as our local planner to plan a path from the current agent's pose to the goal, determining the final action of the agent.

3) Policy Network Architecture: A significant challenge in exploration is achieving global spatial awareness and

avoiding local optima. To address this, we employ a vision transformer [26] paradigm, as illustrated in the right top part of Fig. 2. This transformer captures long-range dependencies across the map observation using its self-attention mechanism. A CNN feature extractor first processes the stacked map to produce high-level features with $C \times P \times P$ dimensions. Then, we split the features into $P^2 \times C$ and combine them with a learnable positional embedding, after which an attention layer learns dependencies between different spatial patches. This attention-based encoder enables the policy to focus on the most relevant parts of the input, leading to more accurate decision-making. Finally, we design a goal generation head as a decoder for generating goal action a_t within the local map. This head utilizes fully connected layers and a normal distribution to generate the final goal (x, y). Additionally, we incorporate an orientation embedding for efficient movement during exploration.

Given that we use PPO for training, the goal-generation head is designed with two branches: one predicts the goal action a_t, while the other estimates the state-value (*i.e.* $V(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, where γ is the discount factor and r is the reward function). Such an architecture is straightforward to train and converge, and it quickly learns informative goals from interactions with the environment.

4) Rewards with Frontier Guidance: To address Problem 1, we design a reward function with two components to facilitate efficient exploration: $r_t(o_t, a_t) = r_g + r_f \cdot \lambda$, where λ is a weight hyper-parameter for balancing the components. The first component $r_g = \mathcal{A}_t - \mathcal{A}_{t-1}$ is area information gain navigating to the global goal a_t. The second part r_f measures the distance between the global goal and frontier edges $D_{f,g}$:

$$r_f = \begin{cases} 0, & D_{f,g} \geq D_{max} \\ 0.05/D_{f,g}, & D_{min} \leq D_{f,g} < D_{max} \\ 0.005, & D_{f,g} < D_{min} \end{cases} \tag{2}$$

where D_{max} and D_{min} are the maximum and minimum thresholds distance between the global goal and frontier edges. The frontier reward incorporates the benefits of traditional frontier-based methods, guiding the global goal to be generated near frontiers, which represent potentially unexplored regions. Consequently, the trained policy is likely to explore the environment more efficiently.

D. Floor-Stair Topology

Another key component of LITE is the floor-stair topology, which decomposes a multi-floor indoor environment into a topological graph containing 2D floor-maps and stairs (see Fig. 1, and Fig. 2). The transition within the floor graph is conducted by a finite state machine, including different states: {`ExploringFloor`, `GoingToStair`, `OnStair`, `AllExplored`}. This structure allows us to seamlessly integrate the learning-based 2D exploration algorithm into the multi-floor exploration without much effort. The overall algorithm, detailed in Algorithm 1, puts the agent in a completely unknown environment $\mathcal{M}$ and incrementally builds a floor-stair topology for environment representation. The key components of the algorithm will be introduced in the following paragraphs.

Algorithm 1: Floor-Stair Topology Builder

Input: Unexplored multi-floor environment $\mathcal{M}$
Output: Floor-stair topology graph $\mathcal{G}$

1 Initialize the agent pose x_t on the first floor of the environment $\mathcal{M}$, and obtain the initial RGB-D observation $\mathcal{O}_t$;
2 $\mathcal{S} \leftarrow$ `ExploringFloor`; $\mathcal{G} \leftarrow \varnothing$;
3 **while** $\mathcal{S}$ is not `AllExplored` **do**
4 Obtain the stair instances $ins_t \leftarrow$ **YOLO11**$(\mathcal{O}_t)$;
5 Update stair map $l_t \leftarrow$ **SemanticMapping**(ins_t);
6 **if** $\mathcal{S}$ is `ExploringFloor` or `GoingToStair` **then**
7 **if** Arrive at a new floor i **then**
8 Create a new floor node v_i;
9 $\mathcal{G}$.**addVertex**(v_i);
10 Store the previous stair information;
11 **if** There is a new stair area in l_t **then**
12 Create a new stair edge e_{ij} based on l_t;
13 $\mathcal{G}$.**addEdge**(e_{ij});
14 Update V and E in the topology graph $\mathcal{G}$;
15 **else if** $\mathcal{S}$ is `GoingToStair` **then**
16 Update E, store the previous floor information;
17 **end**
18 $\mathcal{S} \leftarrow$ **UpdateExplorationStatus**$(\mathcal{G}, \mathcal{S}, x_t)$;
19 Select exploration policy $\pi \leftarrow$ **ChoosePolicy**$(\mathcal{S})$;
20 Generate exploration goal $a'_t \sim \pi$;
21 Navigate one step $\mathcal{O}_t, x_t \leftarrow \mathcal{M}$.step$(a'_t)$;
22 **end**
23 **return** The floor-stair topology graph $\mathcal{G}$;

1) Instance Segmentation Model based on YOLO11: To achieve fast and accurate stair detection (as described in Algorithm 1, Line 4), we developed a stair instance segmentation model leveraging YOLO11 [12]. Given the scarcity of labeled stair data in most public datasets, we randomly collected 1,511 stair images from the internet and performed semi-automatic annotation using the open-source labeling software Labelme and its integrated SAM model [27]. The annotations were then converted into YOLO format for training and validating, after which the model can be deployed to obtain the segmentation masks of stairs. As described in Section B, the semantic mapping approach is employed to project mask information into a stair mask map l_t in $1 \times M \times M$ (see Algorithm 1, Line 5), which is used for subsequent state transitions and policy switching.

2) Topology Builder: In a multi-floor indoor environment, to tackle *Problem 2*, we represent the entire environment as a topological graph $\mathcal{G} = (V, E)$, where $V = \{v_i \mid i = 0, 1, \ldots, n\}$ denotes floor maps and $E = \{(v_i, v_j), v_i, v_j \in V\}$ denotes edges connecting all floors. When exploring a new floor i, we add the 2D grid map v_i of size $2 \times M \times M$ to V and continuously update the floor map in exploration(Line

Algorithm 2: Exploring Status Transition with FSM

Input: Floor graph G_t, Previous exploration status S_{t-1}, Current pose x_t
Output: Current exploration status $\mathcal{S}_t$

$(V_t, E_t) \leftarrow G_t$;
$f_{done} \leftarrow \mathbf{IsCurrFloorDone}(V_t)$;
$g_{all\_visited} \leftarrow \mathbf{IsAllGraphExplored}(V_t, E_t)$;
$e_{on\_stair} \leftarrow \mathbf{IsOnStair}(E_t, x_t)$;
$S_t \leftarrow S_{t-1}$;
if $\mathcal{S}_{t-1} =$ `ExploringFloor` **and** f_{done} **then**
  if $g_{all\_visited}$ **then**
    $\mathcal{S}_t \leftarrow$ `AllExplored` ▷ Cond 4
  else
    $\mathcal{S}_t \leftarrow$ `GoingToStair` ▷ Cond 1
else if $\mathcal{S}_{t-1} =$ `GoingToStair` **and** e_{on_stair} **then**
  $\mathcal{S}_t \leftarrow$ `OnStair` ▷ Cond 2
else if $S_{t-1} =$ `OnStair` **and not** e_{on_stair} **then**
  $\mathcal{S}_t \leftarrow$ `ExploringFloor` ▷ Cond 3
end
return Current Exploration Status $\mathcal{S}_t$;

7-10, 14). The trained YOLO11 instance segmentation module is utilized for real-time stair detection and segmentation. Once we detect a new stair, the topology builder will extend a new edge v_i, v_{i+1} with stair information to V(Line 11-13). If a floor is explored or a stair is visited, their attributes in $\mathcal{G}$ (*e.g.* floor obstacle maps, stair semantics in two floors) are updated and stored for future exploration.

3) Exploration Status Transition: After updating topological graph G_t, a finite state machine (FSM) manages the transition between different floors (see Fig. 2 right bottom part), as shown in Algorithm 1, Line 18, **UpdateExplorationStatus**$(\mathcal{G}, \mathcal{S}, x_t)$. The transition conditions, Cond $1 \sim 4$, are described in Algorithm 2. We firstly extract floor nodes V_t and stair edges E_t from graph G_t and calculate three boolean variables f_{done}, $g_{all\_visited}$ and e_{on_stair}. f_{done} is true when the current exploring floor is fully explored (i.e., explore ratio $> 95\%$ in simulation or no new areas in last T_{thre} time steps) or time budget is reached; $g_{all\_visited}$ is true after all floors V_t and stairs E_t are visited and explored; e_{on_stair} is true if current pose x_t is within the range of a stair semantic map.

The exploration starts on the first floor with status `ExploringFloor`. When the current floor is done, the status transitions to `GoingToStair`, directing the agent to the nearest stair area. Upon reaching this stair, the status changes to `OnStair`, guiding the agent upstairs. After the stair is visited, the status reverts to `ExploringFloor` for further exploration. Once all floors are explored and no new stairs are found, the exploration procedure concludes with status `AllExplored`.

4) Policy Switcher: To handle diverse scenarios effectively, the agent must select an appropriate policy π to determine the action a'_t given different exploration status $\mathcal{S}$, as described in Algorithm 1, Line 19, **ChoosePolicy**$(\mathcal{S})$. When the agent is exploring a specific floor ($\mathcal{S}$ = ExploringFloor), our attention-based 2D exploration policy π_θ is employed, facilitating efficient single-floor exploration. During the process of navigating towards the stairs ($\mathcal{S}$ = GoingToStair), the center point c_k of each detected stair e_k on the current floor are calculated. Subsequently, the center point closest to the agent's current pose x_t is chosen as the goal point $a_t = c_{k^*}$:

$$c_k = \frac{1}{N}\left(\sum_{(x_i,y_i)\in l_k} x_i, \sum_{(x_i,y_i)\in l_k} y_i\right), N = \sum_{(x_i,y_i)\in l_k} 1$$
$$k^* = \arg\min_{e_k \in l_t} \sqrt{(x_t[0]-c_k[0])^2 + (x_t[1]-c_k[1])^2} \quad (3)$$

where l_k means the area of stair e_k on the stair map l_t, (x_i, y_i) denotes the corresponding coordinates in the area of l_k, while x_t means agent pose. This approach streamlines the navigation process, allowing the agent to reach the stairs in the most efficient way. For $\mathcal{S}$ = OnStair, since a stair typically has a ladder structure, it is projected as a rectangular area in the top-down map. In this map, the short side of the rectangle represents the entrance and exit of the stairs. To guide the agent upstairs, we compute the perpendicular bisector of the short side, and the end of the extension line of the center line on the side away from the entrance is selected as the goal. Once all stairs and floors have been explored ($\mathcal{S}$ = AllExplored), the policy stops exploration.

V. Experiments

A. Datasets

We conduct both 2D and 3D exploration experiments on visually realistic Habitat Simulator [28] and HM3D dataset [29] to achieve strong policy generalization capabilities. HM3d includes 800 training and 100 validation scenes, most of which are multi-floor scenes. For 2D exploration, we filtered out 57 training and 12 validation scenes, which are single-floor and well-reconstructed. We also generalize our trained policy to MP3D [30] validation datasets, including 11 large indoor scenes. For multi-floor exploration, we select 6 scenes in HM3D with semantic annotations that are navigable between floors.

B. 2D Exploration Results

1) Training details: We use the Habitat Simulator for training and evaluation, with 480×640 RGB-D observations. The local and global grid map sizes are $L = 240$ and $M = 480$, respectively, and the map resolution is 0.05m for each grid. The CNN feature is reshaped to $P^2 = 225$ patches. During training, each episode has 1000 steps, and if the explored ratio is larger than 95%, the episode is marked success, which will trigger a new episode. Each global step is 25 for generating a new goal. We train our policy on 2 NVIDIA A4000 16 GB GPU for over 15 million frames.

2) Metrics: We use four metrics to compare all methods: Coverage Ratio (CR), Coverage Area (CA), Area weighted by Path Length (APL), and Success Rate (SR). CR represents the ratio of the explored area to the total explorable area at the end of an episode, and CA is the corresponding explored

TABLE I: Results of Comparative Study for LITE-2D.

Method	HM3D				MP3D Generalization			
	CR↑	CA(m^2)↑	APL(m)↑	SR↑	CR↑	CA(m^2)↑	APL(m)↑	SR↑
Nearest Frontier	0.835(±0.125)	62.3(±10.6)	0.832(±0.212)	0.230	0.684(±0.264)	71.4(±33.8)	1.032(±0.412)	0.238
Utility Frontier	0.839(±0.108)	62.9(±11.6)	0.752(±0.189)	0.143	0.718(±0.251)	79.3(±43.6)	1.075(±0.482)	0.265
RH-NBV	0.896(±0.103)	67.3(±12.4)	1.032(±0.226)	0.403	0.745(±0.244)	84.8(±51.7)	1.355(±0.593)	0.321
ANS Global [9]	0.901(±0.097)	67.5(±11.6)	1.039(±0.236)	0.407	0.748(±0.228)	88.3(±56.7)	1.355(±0.574)	0.301
LITE-2D (Ours)	**0.918**(±0.071)	**69.1**(±11.5)	**1.054**(±0.248)	**0.473**	**0.765**(±0.216)	**90.5**(±56.1)	**1.364**(±0.544)	**0.323**

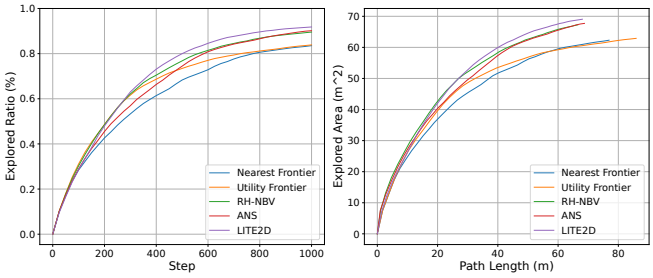

Fig. 3: The plot of CR and CA when exploration proceeds.

area. APL is defined as $\frac{CA}{L}$, where L is the episode path length. SR is defined as S_i, where S_i equals 1 if the coverage ratio exceeds 95%. All metrics are reported as the mean and standard deviation (except for SR, whose standard deviation is not applicable) across all episodes and scenes.

3) Comparison Analysis: We compare our attention-based 2D policy, namely LITE-2D, with several learning and non-learning baselines, including Nearest Frontier [4], Utility Frontier [8], RH-NBVP [5], and ANS global [9]. Nearest Frontier chooses the nearest frontier as the next goal, while Utility Frontier selects the frontier with the highest score: $S(f) = U(f) \cdot e^{-\lambda C(f)}$, where $U(f)$ is utility score of frontier f, $C(f)$ is the shortest path between the agent and the frontier. RH-NBVP instead maintains a Rapid-exploring Random Tree (RRT) and calculates a similar score to select the next best viewpoint (NBV) from the tree iteratively. ANS employs an end-to-end architecture for exploration, including Neural-SLAM, Global Policy, and Local Policy. We only use ANS Global policy with the same input and output for a fair comparison. Note that we transfer frontier-based methods and RH-NBV to the same framework of LITE-2D, and the best node of RH-NBV serves as the global goal.

The comparison results are reported in TABLE I. The table shows that frontier-based methods perform the worst as they always fall into local optima, leaving some regions unexplored. In contrast, sampling-based RH-NBV is more likely to acquire potential information(+6% CR in HM3D) as long as each regions are sampled. We utilize frontier edges instead of clustering them to avoid local optima while easily getting potential information from the map. Compared to the learning-based ANS Global, we achieve significantly better performance (e.g., +1.7% CR, +1.6 CA, +0.015 APL, and +6.6% SR). While ANS Global uses only CNN and fully connected layers as a global policy, our attention-based exploration policy can learn long-range spatial dependencies between different regions and determine a non-shortsighted goal. Generalization tests on MP3D val with larger scenes show that LITE-2D outperforms all baselines regarding exploration efficiency. We remark that the SR is much lower than expected since there are small reconstruction errors in most scenes and limitations on episode steps. In most scenes, LITE-2D can achieve a coverage ratio higher than 90% in 1000 steps. Fig. 3 shows the average exploration ratio and area when exploration in the exploration process, where LITE-2D achieves a more efficient exploration while other baselines experience unnecessary movements.

4) Ablation Study: We consider 3 different variants on HM3D to validate the effectiveness of each section.

- **LITE-2D w.o. Attention:** We remove the attention mechanism and directly flatten the features extracted from CNN to select a global goal.
- **LITE-2D w.o. Frontier:** We discard the frontier reward from training, with only information gain left.
- **LITE-2D w.o. Orientation:** The orientation embedding is removed in the goal generation head, and the policy has no information about the agent's orientation.

The importance of each part in LITE-2D is shown in TABLE II. Once the attention mechanism is removed, the metrics deteriorate to be similar to ANS Global, indicating that the model fails to focus on key areas effectively. In particular, we notice that the APL and SR of LITE-2D w.o. Frontier is much smaller than LITE-2D, which means the last few regions are ignored. We believe this guidance incorporates additional gain information to the policy so it can learn the underlying unexplored regions of the map. Finally, the LITE-2D w.o. Orientation also performs worse than LITE-2D, indicating that incorrect directions can cause the agent to make more turns when navigating to the goal.

TABLE II: Results of Ablation Study.

Method	CR↑	CA(m^2)↑	APL(m)↑	SR↑
LITE-2D w.o. Attention	0.902	67.7	1.015	0.380
LITE-2D w.o. Frontier	0.909	68.2	1.007	0.373
LITE-2D w.o. Orientation	0.907	68.1	1.021	0.410
LITE-2D	**0.918**	**69.1**	**1.054**	**0.473**

C. Multi-floor Indoor Exploration Results

We further perform experiments in 6 multi-floor indoor environments of HM3D to c the ability of floor-stair topology to integrate learning-based 2D exploration policy. As stated in Section IV-D, YOLO11 is employed for real-time instance segmentation of stairs. We trained a high-precision model

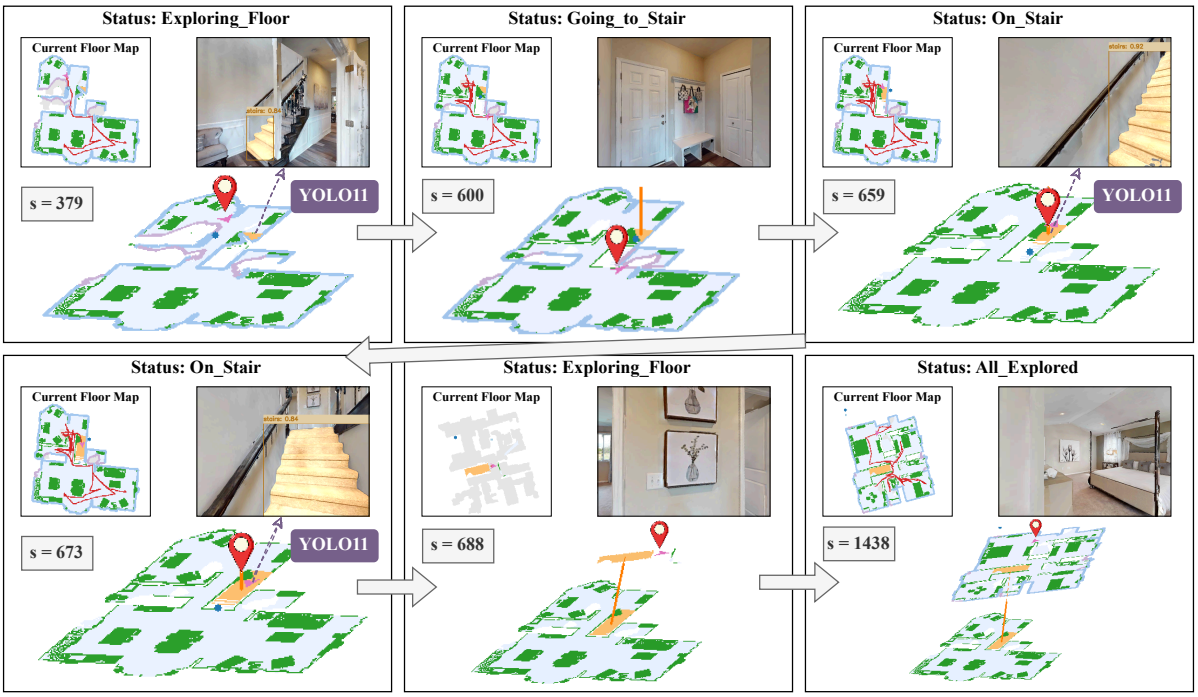

Fig. 4: Qualitative result of LITE for multi-floor indoor exploration in HM3D. All visualizations are the same as Fig. 1.

(up to 0.954 mAP^{50-95}) on our stair dataset using the pre-trained YOLO11n-seg model [12], and directly deploy it in our experiments. We always put the agent on the first floor and explore the environment with floor-stair topology. On each floor, we integrate our trained 2D exploration policy for efficient exploration.

The exploration process of one scene with two floors is shown in Fig. 4. In this scene, LITE detects a new stair when exploring the first floor and efficiently explores the first floor in 600 steps, reaching an explored ratio of 95%. Then, it extends floor-stair topology and guides the agent upstairs. No new stairs were detected on the second floor, so the entire exploration process concludes when the second floor is done at 1438 steps. This demonstrates that our framework effectively integrates 2D learning-based exploration methods in multi-floor environments by utilizing simple floor topology relationships. We also integrate all the explorers mentioned in Section V-B to our floor-topology, comparing the average CR of each floor and total steps when all floors are explored. The experimental results in Table III indicate that using our topological explorer, different methods can be applied to explore multi-floor indoor environments, and LITE-2D outperforms other explorers in terms of exploration efficiency.

TABLE III: Results of Multi-floor Exploration with LITE.

Method	Nearest	Utility	RH-NBV	ANS	**LITE**
CR↑	0.864	0.871	0.905	0.924	**0.929**
Steps↓	1884	1767	1736	1777	**1708**

D. Experimental Validation

We validate LITE in the real world with a quadruped robot, Unitree Aliengo (see Fig. 5). To achieve multi-floor exploration, we select a teaching building with four floors, and set the exploration goal to explore the first two floors, and stop the exploration after the second floor has been explored. Our robot is equipped with a RealSense D435i depth camera and two Livox lidars(for pose estimation). Fine-tuned YOLO11-based segmentation model is converted into ONNX format without extra training to predict stairs. We use [31] as the local planner and learning-based quadrupedal controller for going upstairs. The global goal outputted by LITE is re-planned once the goal is reached or the local planner fails. On each floor, we maintain a grid map with stair masks, and all floors and stairs are represented as a floor-stair topology (see Fig. 5). By utilizing an attention-based exploration policy and the floor-stair topology, our algorithm efficiently explores the entire environment despite significant differences in the environment, sensors, and robot models compared to the training scenarios. LITE ingeniously constructs the two-floor environment to a three-floor topology, this highlights the strong generalization capabilities of our approach.

VI. Conclusion

In this work, we propose the first leaning-integrated topological explorer (LITE) for multi-floor indoor exploration. LITE contains two key components: an attention-based 2D exploration policy and a floor-stair topology for multi-floor exploration. The attention-based 2D exploration policy leverages transformer encoders to capture long-range dependencies of spatial regions and use frontier training guidance to explore unknown areas efficiently. With a floor-stair topology, different 2D exploration policies can be seamlessly integrated into multi-floor exploration. Extensive experiments on both simulation and the real world show our approach's effectiveness and generalization ability. Future

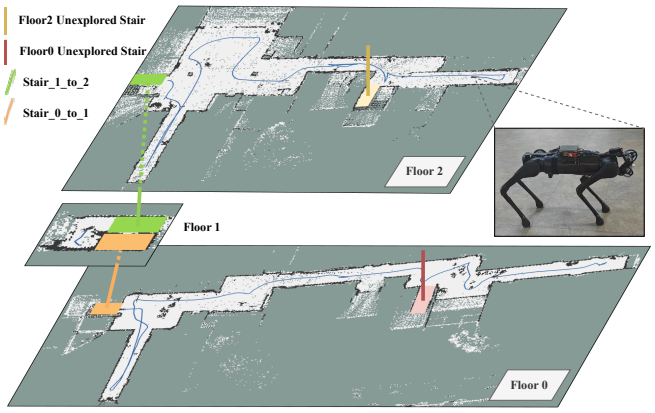

Fig. 5: Final floor-stair topology in the real world

work will consider more complex topologies for sophisticated multi-floor environments.

REFERENCES

[1] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, 2019.

[2] Y. Kompis, L. Bartolomei, R. Mascaro, L. Teixeira, and M. Chli, "Informed sampling exploration path planner for 3d reconstruction of large scenes," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7893–7900, 2021.

[3] A. Tagliabue, S. Schneider, M. Pavone, and A.-a. Agha-mohammadi, "Shapeshifter: A multi-agent, multi-modal robotic platform for exploration of titan," in *2020 IEEE aerospace conference*. IEEE, 2020, pp. 1–13.

[4] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 1997, pp. 146–151.

[5] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[6] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, vol. 37, no. 8, pp. 1363–1388, 2020.

[7] H. H. González-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[8] M. Kulich, J. Faigl, and L. Přeučil, "On distance utility in the exploration task," in *2011 ieee international conference on robotics and automation*. IEEE, 2011, pp. 4455–4460.

[9] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," *arXiv preprint arXiv:2004.05155*, 2020.

[10] Y. Cao, T. Hou, Y. Wang, X. Yi, and G. Sartoretti, "Ariadne: A reinforcement learning approach using attention-based deep networks for exploration," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 219–10 225.

[11] L. Schmid, C. Ni, Y. Zhong, R. Siegwart, and O. Andersson, "Fast and compute-efficient sampling-based local exploration planning via distribution learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7810–7817, 2022.

[12] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *arXiv preprint arXiv:2410.17725*, 2024.

[13] A. Batinovic, T. Petrovic, A. Ivanovic, F. Petric, and S. Bogdan, "A multi-resolution frontier-based planner for autonomous 3d exploration," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4528–4535, 2021.

[14] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. Siegwart, "History-aware autonomous exploration in confined environments using mavs," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[15] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.

[16] F. Chen, J. D. Martin, Y. Huang, J. Wang, and B. Englot, "Autonomous exploration under uncertainty via deep reinforcement learning on graphs," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6140–6147.

[17] R. Reinhart, T. Dang, E. Hand, C. Papachristos, and K. Alexis, "Learning-based path planning for autonomous exploration of subterranean environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1215–1221.

[18] M. Lodel, B. Brito, A. Serra-Gómez, L. Ferranti, R. Babuška, and J. Alonso-Mora, "Where to look next: Learning viewpoint recommendations for informative trajectory planning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4466–4472.

[19] S. Sivashangaran and A. Eskandarian, "Deep reinforcement learning for autonomous ground vehicle exploration without a-priori maps," *arXiv preprint arXiv:2301.04036*, 2023.

[20] Y. Cao, R. Zhao, Y. Wang, B. Xiang, and G. Sartoretti, "Deep reinforcement learning-based large-scale robot exploration," *IEEE Robotics and Automation Letters*, 2024.

[21] V. Krátký, P. Petráček, T. Báča, and M. Saska, "An autonomous unmanned aerial vehicle system for fast exploration of large complex indoor environments," *Journal of field robotics*, vol. 38, no. 8, pp. 1036–1058, 2021.

[22] T. Kim, G. Kang, D. Lee, and D. H. Shim, "Development of an indoor delivery mobile robot for a multi-floor environment," *IEEE Access*, 2024.

[23] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4247–4258, 2020.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts." *proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[26] A. DOSOVITSKIY, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[27] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

[28] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, "Habitat 2.0: Training home assistants to rearrange their habitat," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[29] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. M. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, M. Savva, Y. Zhao, and D. Batra, "Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. [Online]. Available: https://arxiv.org/abs/2109.08238

[30] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.

[31] Z. Zhang, J. Yan, X. Kong, G. Zhai, and Y. Liu, "Efficient motion planning based on kinodynamic model for quadruped robots following persons in confined spaces," *IEEE/ASME Transactions on Mechatronics*, vol. 26, pp. 1997–2006, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:236392521