

Received April 19, 2022, accepted June 1, 2022, date of publication June 13, 2022, date of current version June 16, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3181989

TVENet: Transformer-Based Visual Exploration Network for Mobile Robot in Unseen Environment

TIANYAO ZHANG^{1,2}, XIAO GUANG HU¹, JIN XIAO¹, AND GUOFENG ZHANG¹

¹School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China

²ShenYuan Honors College, Beihang University, Beijing 100191, China

Corresponding author: Jin Xiao (xiaojin@buaa.edu.cn)


This work was supported in part by the National Natural Science Foundation of China under Grant KZ73096202.

ABSTRACT This paper presents a Transformer-based Visual Exploration Network (TVENet) that capably serves as a solution for active perception problems, especially the visual exploration problem: How could a robot that is equipped with a camera explore an unknown 3D environment? The TVENet consists of a Mapper, a Global Policy and a Local Policy. The mapper is trained by supervised learning to take the visual observation as input and generate an occupancy grid map for the explored environment. The Global Policy and the Local Policy are trained by reinforcement learning in order to make navigation decision. Most state-of-the-art methods in visual exploration domain use ResNet as feature extractor, and few of them pay attention to the extraction capability of the extractor. Therefore, this paper focuses on enhancing the extraction capability, and proposes a Transformer-based Feature Pyramid Module (TFPM). Moreover, two tricks for training process are introduced to improve the performance (M.F. and Aux.) Our experiments in photo-realistic simulated environment (Habitat) demonstrate the higher-accuracy mapping of TVENet. Experimental results prove that the TFPM and tricks have positive impacts on the mapping accuracy of the visual exploration and increase it by 5.31% compared with the state-of-the-art. Most importantly, the TVENet is deployed on a real robot (NVIDIA Jetbot) to prove the feasibility of Embodied AI approaches. To the authors' knowledge, this paper is the first one that proves the viability of the Embodied AI style approach for visual exploration tasks and deploys the pre-trained model on the NVIDIA Jetson robot.

INDEX TERMS Active perception, embodied AI, learning for navigation, visual exploration, visual navigation.

I. INTRODUCTION

Visual exploration is a stem in active perception, which is actively capturing the necessary visual observations [1]. As pointed out by Chaplot *et al.* [2], the purpose of exploration is to efficiently visit as much of the environment as possible on a limited time budget. Ramakrishnan *et al.* [3] define this task as learning to look around: How can an agent learn to acquire informative visual observation skills? Visual exploration is an important part of navigation since it can actively gather useful information in the new environment for unspecified downstream tasks [1]. These tasks could be PointGoal Navigation, ObjectGoal Navigation, AreaGoal Navigation [4], Visual-Language Navigation [5], etc.

The associate editor coordinating the review of this manuscript and approving it for publication was Abdullah Iliyasa .

The traditional methods of visual exploration could be traced back to the 1980s [6], [7]. Traditional methods of visual exploration attempt to utilize purely geometric representations, such as simultaneous localization and mapping (SLAM) [8], visual odometry [9], visual landmark [10], appearance-based methods [11], and optical flow [12]. However, they entirely ignore the semantic information [13]. Using the ORB-SLAM3 [8] as an example, VSLAM focuses on keyframe and point matching, updating, and tracking. Furthermore, these methods mainly rely on sensors' readings to build maps and localize them, making them highly susceptible to measurement noise [14] and need more computational resources [9].

Therefore, traditional methods are generally hard to accomplish embodied AI tasks such as Navigation from Dialog History (NDH) [15], Visual Language Navigation (VLN) [16], Embodied Question Answering (EQA) [17],

Interactive Question Answering (IQA) [18], Scenario Oriented Object Navigation (SOON) [19], and etc. As computer vision technology has improved in recent years, visual exploration approaches have altered drastically, particularly with the appearance of the CVPR Embodied AI workshop in 2020.¹ We called the method of deploying the learning-based model, which was trained in a photo-realistic simulation environment, on a real robot as the “Embodied AI style approach.”

Under these conditions, learning-based methods stand out and catch the attention of the researchers [20]. Firstly, they argue that the optimization of the overall navigation system is more important than intermediate task goals like accurate localization and metric maps [21]. Secondly, they require much cheaper sensors like cameras, not the expensive Lidars or depth sensors [5], [22]. Thirdly, they extract the semantic information for navigation [23]. Finally, they can generalize robustly in previously unseen environments [24], [25]. Savva *et al.* [26] demonstrate that learning-based approaches outperform classic approaches (ORB-SLAM2 [27]) when the agent has more learning steps and data. A representative work of this style is conducted by Ramakrishnan *et al.* [28] in Facebook AI Research. Therefore, our TVENet follows their research path. Compared with their work, this paper enhanced the ability to perceive visual information and deployed TVENet on a real robot to prove the feasibility of the Embodied AI style approaches. Fig. 1 depicts a bird view of our work.

The learning-based methods operate directly on pixels and/or depth as input [28]. It could be an end-to-end paradigm or a modular architecture paradigm. The end-to-end paradigm means designing a network which directly maps the observation into action without any internal feature space for representing the environment (e.g., [22], [29]). The modular architecture paradigm means using several modules to perceive the environment from observation and then compute the action (e.g., [25], [30]). Our TVENet is categorized as a modular architecture paradigm.

Most researchers use ResNet [31] as the basic visual feature encoder for visual exploration. They pay more attention to the post-processing of the extracted features, but tend to ignore the extraction capability of the visual encoder. In this situation, the Visual Transformer technology [32] attracted our attention with its strong representation capabilities.

Therefore, this paper proposes a novel Transformer-based Feature Pyramid Module (TFPM) that takes advantage of Transformer [33] to enhance the extraction capability. Inspired by the work of Gordon *et al.* [34], this paper introduces two tricks: Training Mapper First (TMF) and Auxiliary Task (Aux) for the training process. This paper validates our TVENet, TMF, and Aux on Habitat with Gibson datasets [35] and Materport3D datasets [36]. Habitat is a photo-realistic simulator introduced by Savva *et al.* [26] to render photo-realistic observation of an agent in it.

¹<https://embodied-ai.org/>

Our main contributions are as follows:

- 1) Proved the viability of the Embodied AI style approach for visual exploration tasks. This manuscript successfully deployed the learning-based perception-decision model (TVENet), which was trained in a photo-realistic simulation environment, on a monocular real robot.
- 2) Proposed a novel module (TFPM) to enhance the ability to perceive visual information. TFPM combines Transformer in the NLP task and FPN in the CV task to extract and merge the multi-scale features. Compared with the SOTA method [28], the proposed TFPM increases mapping accuracy by +5.31%.
- 3) Introduced two efficient training tricks that significantly improve training performance. The mapping accuracy is improved by +9.66% with these tricks.
- 4) Conduct a practice-oriented discussion for the embodied AI style approach. Based on the experiments in the simulation environment and the real world, this paper carefully discusses the things that would be helpful for practicability and the potential future work direction.

II. RELATED WORKS

A. VISUAL EXPLORATION

Navigation has been well studied in classical robotics, and exploration is a basic part of navigation, especially exploring in new and unknown environments. Practically, the robot (agent) only obtains a partial observation of the environment at a certain time step. As a result, researchers define exploration as a partially observed Markov Decision Process (POMDP) [1]. The purpose of POMDP is to figure out the distribution of probability over each action at the current observation. This distribution is also called “policy.” Chen *et al.* [13] formulate the estimation of optimal policy π^* as a learning problem. The policy π is trained on a set of environments ε_{train} and tested on a held-out set of environments ε_{test} . Li *et al.* [37] design a Deep Reinforcement Learning method to automatically explore an unknown environment, but they use laser and odometry data as input. Luong *et al.* [38] use range finder laser sensors and online deep reinforcement learning to generate the navigation policy. However, their inputs are sensitive to the noise of sensors’ readings.

In this context, computer vision plays a more important role [39]. Some researchers focus on the functional components of exploration, like obstacle avoidance. Ren *et al.* [40] formulate the motion planning problem of an AGV with static and dynamic obstacles as a nonlinear optimal control problem (OCP). Zhang *et al.* [41] create an actively obstacle-avoiding algorithm for aircraft swarms while exploring. Narayan *et al.* [42] propose a dynamic color perception system for visual exploration on an unmarked road.

However, they utilize the color information and ignore the other visual information like semantic and category information. Moreover, their method only works well in a structured environment like a road.

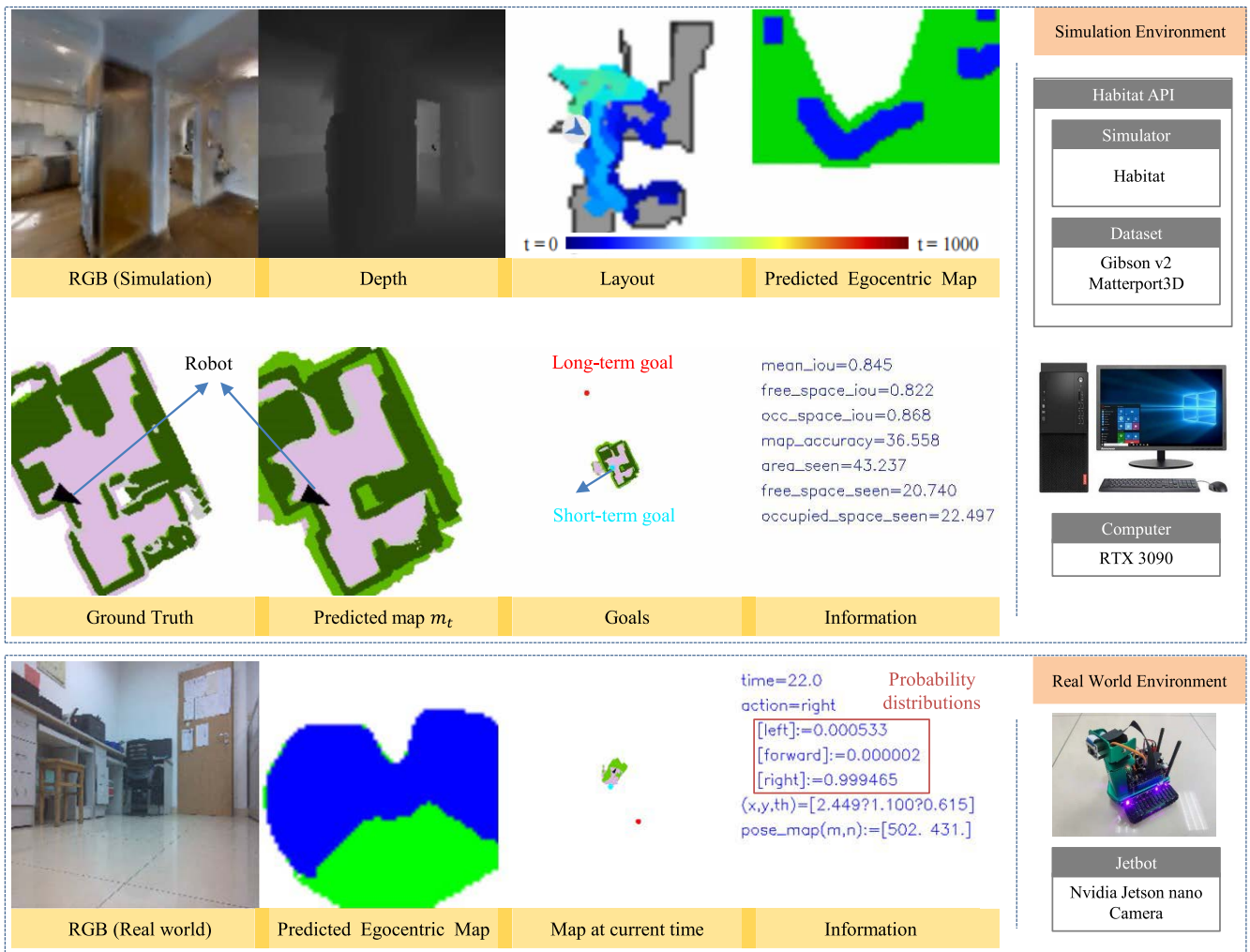


FIGURE 1. Exploration task and the bird view of the work in this paper. The proposed network takes a sequence of RGB images as the input and outputs the predicted map. The proposed network is trained in a simulation environment and deployed on a real robot to test in a real-world environment. The simulation is based on a photo-realistic simulator (Habitat) and several 3D environment datasets (Gibson-v2 and Matterport3D). The real robot is equipped with a powerful processor (Jetson nano) and a camera. The light (dark) gray area in “Layout” is the explored (unexplored) region of the unseen environment. The colored line in “Layout” represents the path of the robot.

Differently, Chaplot *et al.* [43] combine the classic processing pipeline and a learning-based method for a more complex environment, and propose an Active Neural Network (ANS). Their method won the AI Habitat challenge at CVPR 2019 across all tracks. Ramakrishnan *et al.* [28] formulate the exploration task as a pixel-wise classification task and add anticipation mechanism. They won the 2020 Habitat PointNav Challenge. Our TVENet is compared with the work by Ramakrishnan *et al.* [28].

B. RESNET AND TRANSFORMER

ResNet is an excellent backbone for extracting multi-scale features in the computer vision domain [31]. It is the dominant visual encoder in the visual exploration domain. Zhu *et al.* [25] utilize pretrained ResNet-50 as a main part of their actor-critic model. Savinov *et al.* [44] use ResNet-18 in their retrieval network to process the two input

observations. Gupta *et al.* [45] use ResNet-50 to extract features for their Cognitive Mapping and Planning (CMP) network. Chaplot *et al.* [46] use shared ResNet-18 to encode a source image and a goal image. Anderson *et al.* [16] use ResNet-152 to extract a mean-pooled feature for each image observation. Chen *et al.* [13] use ResNet-18 to process the RGB images and train the exploration policy. Henriques *et al.* [47] use ResNet-50 to design their MapNet. Their method outperforms a learned LSTM policy without a map in previously unseen environments. Li *et al.* [48] encode visual information through ResNet-18 for their meta-learning.

In addition to ResNet, Transformer is another choice for extracting the visual feature. Transformer is a type of deep neural network mainly based on the self-attention mechanism and pose embedding method [32]. At first, it is applied to the field of natural language processing (NLP). Recently,

researchers applied the Transformer to computer vision tasks and got state-of-the-art performance.

III. EMBODIED TASK SETUP

The exploration task could be formulated as the objective to maximize the coverage on a fixed time budget, as proposed by Chen *et al.* [13]. This paper follows the definition of coverage proposed by Chaplot *et al.* [43] where the coverage is the total area of the map known to be traversable. The main purpose of TVENet is to train a navigation policy that takes visual observation o_t and pose p_t at time step t from the statement s_t of an unseen environment and outputs a navigational action a_t to maximize the coverage.

A. PROBLEM FORMULATION

Mathematically, the navigation task in an unseen environment could be formulated as a *Partially Observable Markov Decision Process* (POMDP):

$$\{S, A(s_t), O, P(s_{t+1}|s_t, a_t), R(s_t, a_t)\} \quad (1)$$

where S represents the state space of the surrounding environment and the agent, $A(s_t)$ represents action space, O represents the information space that the agent could observe, $P(s_{t+1}|s_t, a_t)$ represents the state transition probabilities, and $R(s_t, a_t)$ represents the received reward after taking action $a_t \in A(s_t)$ at state $s_t \in S$. The policy π is a mapping from states to probabilities of selecting each possible action [49]. A neural network with parameters θ is typically used to represent the navigation policy, which predicts the distribution $\pi_\theta(\cdot|o_t, p_t)$ over action $a_t \in A(s_t)$ for the observation $(o_t, p_t) \in O$ along the policy π_θ . And the training process is to find the optimal policy π^* so as to maximize the cumulative expected future reward:

$$\begin{aligned} \pi^* &= \arg \max_{\pi} E[R(\tau)|\pi] \\ &= \arg \max_{\pi} E \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | \pi \right], \end{aligned} \quad (2)$$

where $\tau = (s_0, a_0, s_1, a_1, \dots)$ is a sequence of states and actions. It is also frequently called ‘‘trajectory’’ or ‘‘rollouts.’’ Return $R(\tau)$ represents the cumulative reward over a trajectory τ . The $s_t \in S$ is the state of the environment at time t . The γ is the discount factor. The action $a_t \in A$ is determined by the policy π with parameters θ : $a_t \sim \pi_\theta(\cdot|o_t, p_t)$. The next state s_{t+1} follows the dynamics of the environment: $s_{t+1} \sim P(\cdot|s_t, a_t)$.

The *On-Policy Value Function* $V^{\pi_\phi}(s)$, the *On-Policy Action-Value Function* $Q^{\pi_\phi}(s, a)$, and the advantage $A^\pi(s_t, a_t)$ in our training process are defined as follows:

$$V^{\pi_\phi}(s) \doteq E_{\tau \sim \pi_\phi} [R(\tau)|s_0 = s] \quad (3)$$

$$Q^{\pi_\phi}(s, a) \doteq E_{\tau \sim \pi_\phi} [R(\tau)|s_0 = s, a_0 = a.] \quad (4)$$

$$A^\pi(s_t, a_t) \doteq Q^\pi(s_t, a_t) - V^\pi(s_t). \quad (5)$$

Our objective is to design a network π with a parameter θ, ϕ to represent the distribution $\pi_\theta(\cdot|o_t, p_t)$ and the On-Policy value V^{π_ϕ} , respectively. Then, training the network to approximate optimal policy π^* in (2). Finally, mapping the observation into an accurate occupancy grid map based on the optimized networks.

B. ACTION SPACE

The action space A consists of four actions: *FORWARD*: move forward by 25cm; *TURN-RIGHT*: on the spot rotation clockwise by 10 degrees; *TURN-LEFT*: on the spot rotation counter-clockwise by 10 degrees; *STOP*: task completed. These configurations are the same as in the work by Ramakrishnan *et al.* [28], so the comparison is meaningful. It can also be set to any value in the simulation environment. The state space S consists all of the information about the surrounding environment, such as semantic information, geometric information, and so on. This paper denotes $o_t \in O$ as the visual observation, especially the RGB frame, and denote $p_t \in O$ as the pose of the agent, where O is the observation space.

C. REPRESENTATION OF THE ENVIRONMENT

To represent the environment, this paper uses an allocentric metric map m_t . The spatial map m_t is a $2 \times M \times M$ matrix where M represents the map’s size. The first channel of m_t is a 2D map ($M \times M$) that provides the probability of an obstacle in each grid. The second channel contains the probability of the grids being explored or not. The allocentric metric map is updated by the egocentric map $m_t^{ego} \in [0, 1]^{2 \times V \times V}$ at time t , where V is the vision range.

There are three main coordinate systems: the world coordinate system, the allocentric map coordinate system, and the egocentric map coordinate system. The allocentric map coordinate system is the discretization of the world coordinate system. The egocentric map coordinate system could be viewed as the body’s coordinate system. This paper uses the mark (\prime) to represent the egocentric map coordinate.

Considering the observation noise and the action noise, this paper uses the Pose Estimator $f_{PE}(\cdot)$ in the Mapper to correct the sensors’ reading value of the pose. The noisy value of pose in the world coordinate system at time t is represented as $p_t = [x, y, \theta]$, where $(x, y) \in \mathbb{R}^2, \theta \in (-\pi, \pi)$. This paper denotes \hat{p}_t as the corrected value of p_t which approximates the true value of the pose \tilde{p}_t :

$$\tilde{p}_t \approx \hat{p}_t = f_{PE}(m_t^{ego}, m_{t-1}^{ego}, p_t, p_{t-1}). \quad (6)$$

The camera’s shaking is not taken into account in this paper. We consider that using physical dampening (e.g., gimbals) rather than limited computer resources to solve the vibration problem is more economical.

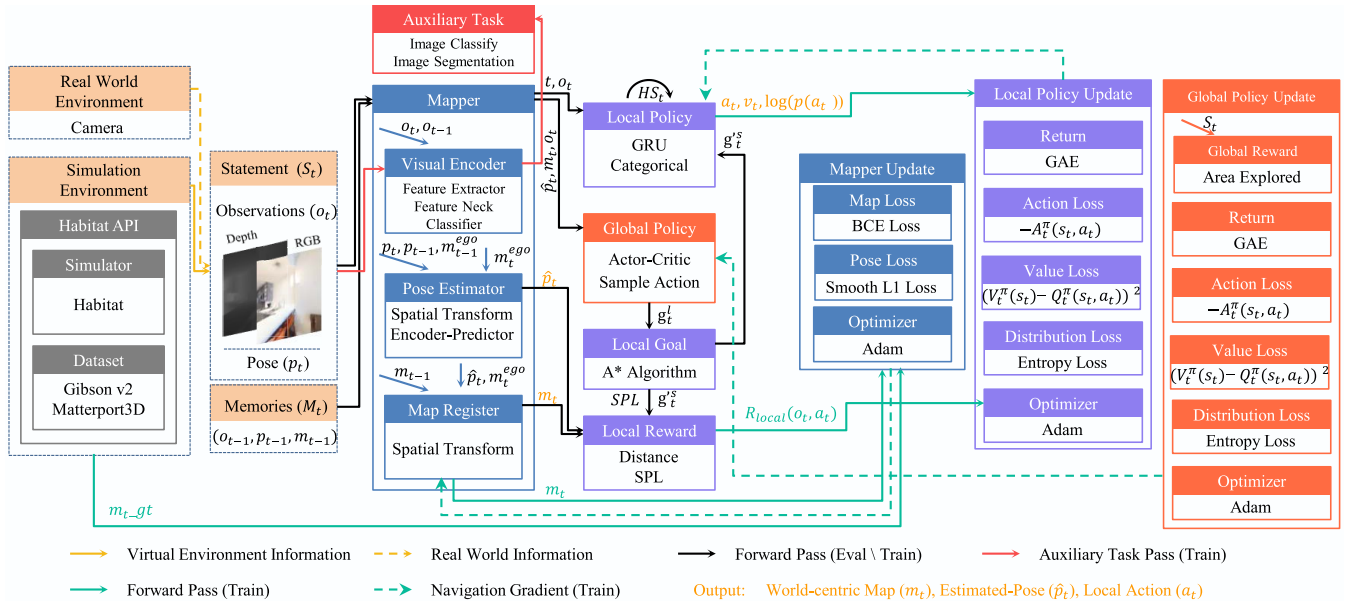


FIGURE 2. Model overview. Our TVNet incorporates two adjacent observations $(o_t, p_t, o_{t-1}, p_{t-1})$ of the environment to learn a grid representation (m_t) of the environment and two policy for deciding an action (a_t) . Based on these perceptions (s_t, M_t) , the agent decides the action to maximize coverage of the explored area on a fixed time budget. The Global Policy figures out a long-term goal (g_t^l) on the allocentric map coordinate system based on the estimated pose (\hat{p}_t) and grid map (m_t) . The g_t^l is then processed into a short-term goal (allocentric coordinate: g_t^s , egocentric coordinate: g_t^e) and fed into the Local Policy, which determines the motion action (a_t) and action-value $(v_t = Q^x(s_t, a_t))$.

IV. METHODS

A. NETWORK ARCHITECTURE

The overview of the model is shown in Fig.2. Inspired by Chaplot *et al.* [43], our TVNet consists of three main parts: Mapper, Global Policy, and Local Policy.

The *Mapper* takes current observations (o_t, p_t) and memory $M_t = (o_{t-1}, p_{t-1}, m_{t-1})$ as input and updates the new map m_t and the corrected pose \hat{p}_t :

$$m_t, \hat{p}_t = f_M(o_t, p_t, o_{t-1}, p_{t-1}, m_{t-1}), \quad (7)$$

where the Mapper network $f_M = \{f_{VE}, f_{PE}, f_{MR}\}$ contains a visual encoder f_{VE} , a pose estimator f_{PE} and a map register f_{MR} . As shown in Fig.3, the observation is fed into f_{VE} (feature extractor, neck, and task head) to get the egocentric map m_t^{ego} . Similar to Chaplot *et al.* [43], the Spatial Transformation (affine transformation) is applied to the last egocentric map m_{t-1}^{ego} with the sensors reading pose (p_t, p_{t-1}) as the input, i.e., $\tilde{m}_{t-1}^{ego} = f_{ST}(m_{t-1}^{ego}, p_t, p_{t-1})$. Then, the estimated pose \hat{p}_t is calculated by f_{PE} , i.e., $\hat{p}_t = f_{PE}(\tilde{m}_{t-1}^{ego}, m_t^{ego})$. Finally, the allocentric map m_{t-1} would be updated by the Map Register with \hat{p}_t and m_t^{ego} .

The *Global Policy* takes the spatial map m_t , the current position \hat{p}_t , and visited locations as inputs h_t to learn a policy (network) π^l with parameters θ and predict an *On-Policy Value* $\bar{v}_t = V^{\pi^l}(s_t)$ with parameters ϕ . The long-term goal (g_t^l) is sampled based on the distribution $\pi_\theta^l(h_t): g_t^l \sim \pi_\theta^l(\cdot|h_t)$. This Global Policy is an actor-critic model with multi-convolution layers [25]. The actor part and the critic part consist of five convolution layers with parameters θ and ϕ respectively.

The *Planner* calculates a short-term goal g_t^s based on the long-term goal g_t^l , the current map m_t , and an estimated pose \hat{x}_t . It uses the A-star algorithm to figure out the shortest path based on current observation instead of the fast-marching [51] used in [43]. The unexplored area is considered free space. A short-term goal g_t^s is sampled with a fixed length from the planned path between the current position and the goal g_t^s .

The *Local Policy* takes an observation o_t and a short-term goal g_t^s as inputs and predicts an action $a_t \sim \pi_\theta^s(\cdot|o_t, g_t^s)$, where θ is the parameter of the Local Policy. To begin, ResNet-18 is used to encode the RGB (D) of the observation o_t into 256-dimensional features F_{rgb} . The word embedding method is used to embed the scalars of position ρ , angle φ , and time t into 32-dimensional features F_ρ, F_φ , and F_t . Then, the concatenation of these features is fed into the Gate Recurrent Unit (GRU) [52]. Finally, the outputs of GRU are divided into two parts: the action distribution $\pi_\theta^s(\cdot|o_t, g_t^s)$ and the expected return $v_t = V^{\pi_\theta^s}(s_t)$.

One trick for training is Training Mapper First (TMF), which is inspired by the fact that decision-making is meaningful only when it is based on an accurate map. This paper implements TMF by changing the training epoch size based on the loss of the Mapper. At the beginning, the epoch size is set largely, say 400. Then, it decreases with the reduction of the Mapper loss.

As pointed out by Zhu *et al.* [5], the auxiliary task would enhance the performance and increase the generalization ability. Inspired by Gordon *et al.* [34], this paper introduces one trick called the Auxiliary Task (Aux) before the training

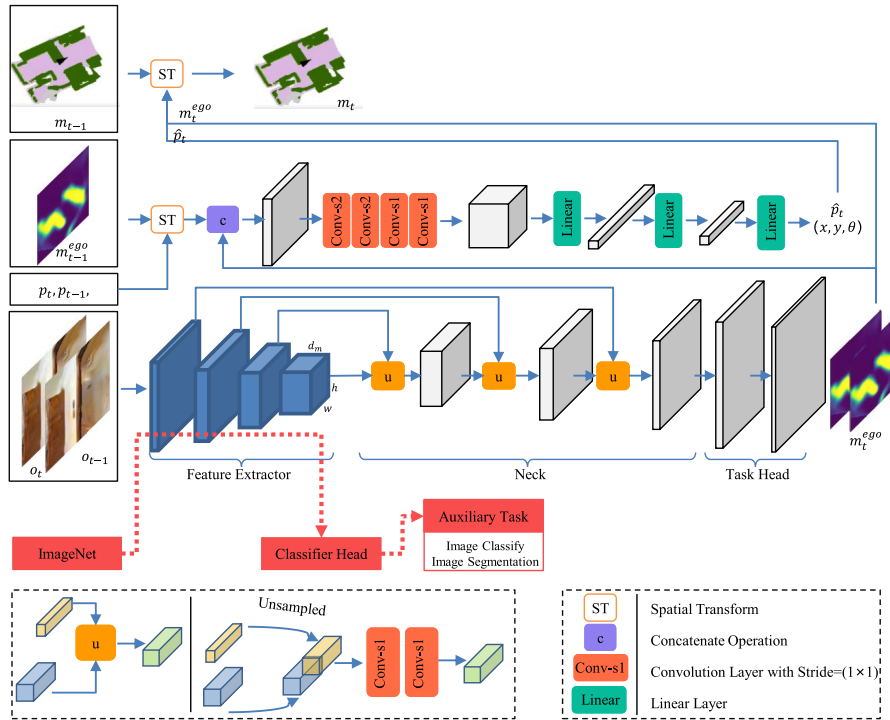


FIGURE 3. Mapper. The mapper takes the RGB frames as input and figures out the representation m_t of environment. In the pipeline of processing the visual information, the Visual Encoder (feature extractor, neck and task head) plays an important role. The auxiliary task (classification and segmentation) is applied on the Feature Extractor with the ImageNet [50] as the datasets to initialize the distribution of the initial parameters. The pose of agent is corrected by the pose estimator with egocentric map (m_t^{ego}, m_{t-1}^{ego}) and sensors reading pose (p_t, p_{t-1}). Finally, the allocentric map m_{t-1} is updated.

process to enable the visual encoder with the ability to extract semantic information. During the task, the parameters of the Global Policy and the Local Policy would be frozen. Intuitively, Aux could be viewed as a method to initialize the parameters of the network before formal training.

B. VISUAL ENCODER: TFPM

Compared to others, this paper pays more attention to the capability of extracting visual information. This paper presents our Transformer-based Feature Pyramid Module (TFPM) to apply Transformer technology in the visual encoder to extract semantic and geometric information from RGB images. Our TFPM consists of a feature extractor, a neck, and a task head. The experiments show that the TFPM outperforms the ResNet-based visual encoder by a significant margin.

1) FEATURE EXTRACTOR

The feature extractor is based on ResT [33], the current state-of-the-art network in the image recognition domain. It is an efficient multi-scale vision Transformer that can serve as a general-purpose backbone to extract multi-scale features. There are two main modifications to the standard ResT for the exploration task as shown in Fig.4. Firstly, this paper removes the classifier layer (the last pooling layer and fully-connected

layer) because it is designed for the image classification task. Secondly, this paper uses multi-scale features from different stages, as inspired by Feature Pyramid Network (FPN) [53] and U-Net [54]. Compared with the standard Transformer network, the feature extractor is only the encoder part of the Transformer.

The stem and patch embed layer in Fig.4 are used to decrease the spatial resolution and increase the channel dimension. Here this paper uses several convolution layers with stride 2 to achieve this purpose.

The position encoding exploits the order of the sequence. Here, this paper utilizes dot-product attention since it is much faster and more space-efficient than additive attention [55]. This paper follows the work by Zhang *et al.* [33] and apply a depth-wise convolution operation to compute the weight for each element of the features:

$$y = x \cdot DWConv(x). \quad (8)$$

In the Block part, to reduce computation and compress memory, the depth-wise convolution layers with stride 2 are used to down-sample input feature $x \in \mathbb{R}^{w \times h \times d_m}$ into $x' \in \mathbb{R}^{\frac{w}{s} \times \frac{h}{s} \times d_m}$, where s is the scale factor. In Efficient Multi-Head Self-Attention (EMSA), x and x' are flattened into $x_1 \in \mathbb{R}^{n \times d_m}, x'_1 \in \mathbb{R}^{n' \times d_m}$ respectively, where $n = w \times h$ and $n' = \frac{w}{s} \times \frac{h}{s}$. Natural Language Processing (NLP) considers x_1 to be n features of dimension d_m . The Linear layers map

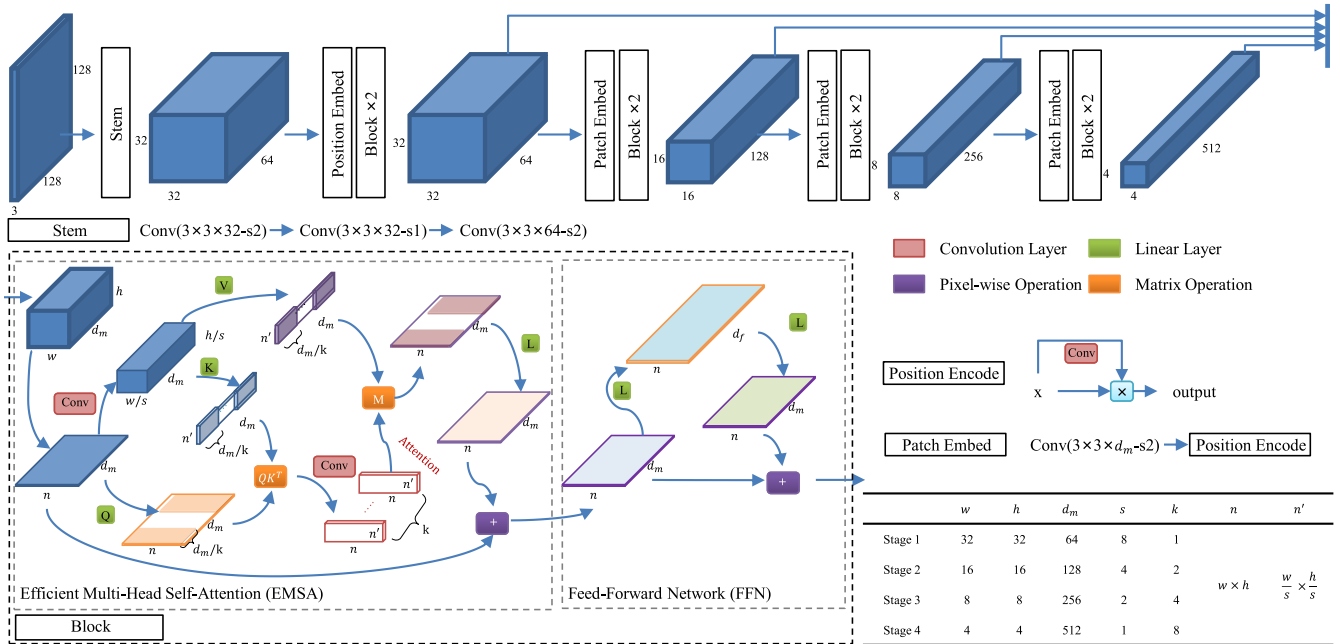


FIGURE 4. TFPM Feature Extractor. As with ResNet, there are four stages to extract four features with different scales. This network uses a self-attention mechanism to merge the information from the different channels. The input (RGB or features) is embedded by convolution layers. To encode the position of different channels, a convolution layer is used to compute the weight of each element of the input feature. For simplification, normalization layers and active layers are not drawn.

x_1 to k query matrix $Q : Q \in \mathbb{R}^{n \times \frac{d_m}{k}}$, where k is the number of EMSA heads. Similarly, x'_1 is translated into k key-value pairs $(K, V) \in \mathbb{R}^{n' \times \frac{d_m}{k}}$. The output of EMSA is a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key:

$$EMSA(Q, K, V) = \text{Softmax}(\text{Conv}(\frac{QK^T}{\sqrt{d_k}}))V, \quad (9)$$

where, $\text{Conv}(\cdot)$ is a 1×1 convolution layer used to combine information from multiple attention heads. For each block in the Feature Extractor, the output y is as follows:

$$\begin{cases} y = y' + \text{FFN}(\text{LN}(y')) \\ y' = x + \text{EMSA}(\text{LN}(x)), \end{cases} \quad (10)$$

where $\text{LN}(\cdot)$ represents the Layer Normalization [56], and $\text{FFN}(x) = \sigma(xW_1 + b_1)W_2 + b_2$ is the Feed-Forward Network(FFN). The $W_1 \in \mathbb{R}^{d_m \times d_f}$, $W_2 \in \mathbb{R}^{d_f \times d_m}$, $b_1 \in \mathbb{R}^{d_f}$ and $b_2 \in \mathbb{R}^{d_m}$ are the parameters of the Linear layers in FFN. The $\sigma(\cdot)$ is the activation layer GELU [57].

2) NECK

As illustrated in Fig.3, The Neck is intended to progressively merge the multi-scale features in order to enhance the lower-level features using the higher-level semantic information, as inspired by the decoder of the U-Net [54]. The multi-scale features $\{f_1, f_2, f_3, f_4\}$ are extracted by the Feature Extractor in corresponding stages. The highest resolution

of the feature f_1 contains more detailed pixel level (low-level) information (e.g., edges and corners), while the lowest resolution of the feature f_4 contains more semantic level (high-level) information (e.g., categories).

During the merging process, function $U(\cdot)$ up-samples the higher level feature f_{i+1} to the same resolution as another feature f_i :

$$u(f_i, f_{i+1}) = \text{Conv}(\text{Conc}(f_i, U(f_{i+1}))), \quad i = 1, 2, 3 \quad (11)$$

where $\text{Conc}(\cdot)$ is the concatenate layer and $\text{Conv}(\cdot)$ represents the convolution layers with stride 1. This process is shown at the bottom of Fig.3.

3) TASK HEAD

Inspired by the detection head in the object detection domain, this paper adds the task head after the neck to accomplish a certain task. The purpose of TFPM in the exploration task is to figure out the egocentric map m_t^{ego} . Thus, the task head is designed to map the merged feature f_m into m_t^{ego} . The task head in this case is made up of two 3×3 convolution layers with stride 1 to compress the 32-dimensional feature f_m into 2-dimensional map m_t^{ego} :

$$m_t^{ego} = \sigma(\text{Convs}(U(f_m))) \quad (12)$$

where $\text{Convs}(\cdot)$ represents convolution layers, $\sigma(\cdot)$ represents the Sigmoid function, and $U(\cdot)$ represents the up-sampled function to extend the resolution.

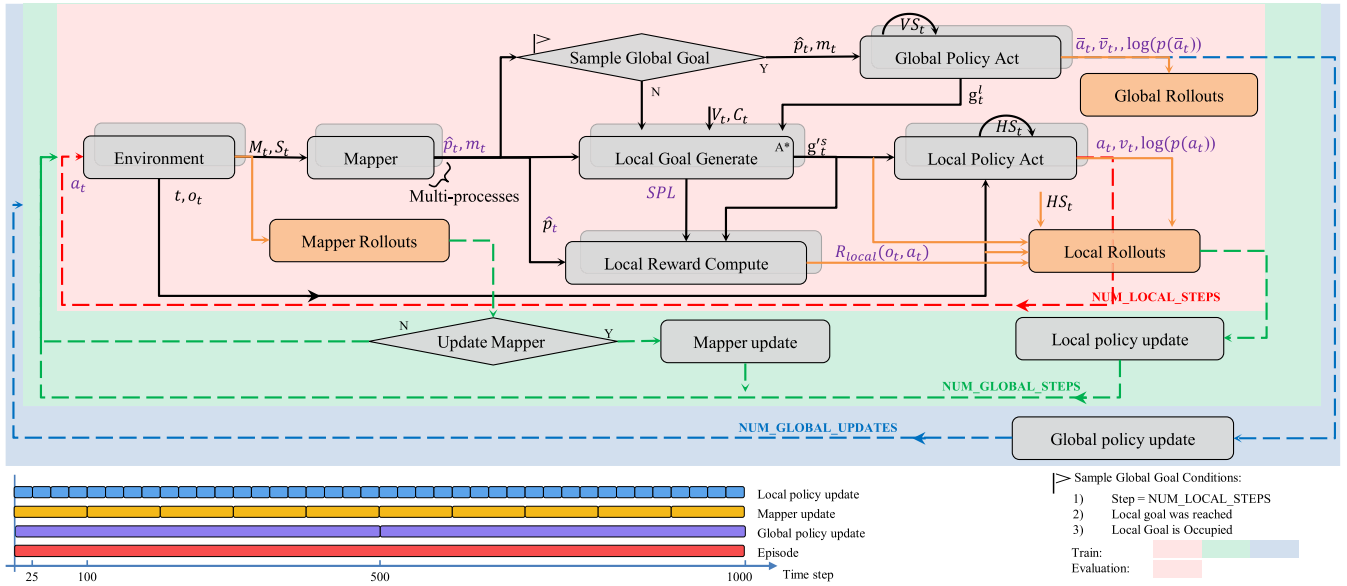


FIGURE 5. Training details. There are three groups of parameters that need to be optimized: the Mapper, the Global Policy (GP), and the Local Policy (LP). The Mapper is trained by the standard supervised learning. The GP and the LP are optimized by the Proximal Policy Optimization (PPO). Thus, three Rollouts are used to gather sufficient input (and rewards) for training. One time step denotes a single interaction with the environment (take action a_t and get the feedback observation o_{t+1}). The LP, Mapper, and GP are updated every 25, 100 and 500 time steps, respectively. Each episode, which begins with a random scene and location, contains 1000 time steps for training.

C. TRAINING DETAILS

The training and the evaluation process are under the platform with 2 GPUs, GeForce RTX 3090 and GeForce GTX 1080. The memory capacity is 24 GB and 8 GB, respectively. The former is used to load the network and optimize the parameters. The latter is used to render the environmental characteristics which are fed into the network.

The training details are shown in Fig. 5. There are three loops to gather the training data for the Mapper, the Global Policy, and the Local Policy. Each loop outputs a rollout τ , a sequence of states and actions: $\tau = (s_0, a_0, s_1, a_1, \dots, s_T, a_T)$.

1) REINFORCEMENT LEARNING

The first loop is for updating the Local Policy, as indicated by the red dot line in Fig.5. Here this paper sets the length of the local policy trajectory to 25 ($\text{NUM_LOCAL_STEPS} \sim \$ = \$T = 25$), which means updating Local Policy every 25 time steps. The reward r_t^s of the action just taken (a_t) is defined as follows:

$$r_t^s = \text{Dist}(g_{t-1}^s, \hat{p}_{t-1}) - \text{Dist}(g_{t-1}^s, \hat{p}_t) + C_0 + F_{succ} \frac{l}{\max(l, p)} + [F_{collision} \cdot C_1], \quad (13)$$

where C_0 is the slack rewards (time information) for the local policy, which means the rewards that are given to an agent as time goes on. C_1 represents the collision reward, and $F_{collision}$ represents a binary indicator of collision (1 for colliding and 0 otherwise). This paper lets $C_0 = -0.3$, $C_1 = -1.0$ in our TVENet network. The $\text{Dist}(g_t^s, \hat{p}_t)$ is the shortest path length

in the egocentric map m_t^{ego} between the short-term goal g_t^s and currently estimated pose \hat{p}_t . This path is computed by the A-star algorithm. The $F_{succ} \frac{l}{\max(l, p)}$ is the Success weighted by Path Length (SPL), where l is the shortest derivable path length from the starting point to g_t^s in this trajectory. p is the length of the actual path. F_{succ} is a binary indicator of success (1 for reaching the short-term goal g_{t-1}^s and 0 otherwise).

The second loop is for updating the Global Policy, as indicated by the green dot line in Fig.5. This paper sets the global policy trajectory length to 20 ($\text{NUM_GLOBAL_STEPS} \sim \$ = \$T = 20$). The Global Policy trajectory is collected only when Local Policy finishes one updating process. Thus, Global Policy is updated every 500 time steps. The reward r_t^g of the action just taken (a_t) is defined as follows:

$$r_t^g = C_2 \cdot \sum m_t^{gt}, \quad (14)$$

where C_2 is the coefficient, m_t^{gt} is the ground truth visited map at time t (1 for being visited and 0 otherwise). This reward r_t^g represents the coverage area of exploration.

Considering the fact that the rewards obtained before taking an action have no bearing on how good that action was, this paper uses the infinite-horizon discounted reward-to-go return $\hat{R}_t(\tau)$ to replace $R(\tau)$ in (2):

$$\hat{R}_t(\tau) = \gamma^{T-t} v_T + \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}, \quad t \in [0, T-1], \quad (15)$$

where $v_T = V^{\pi_\phi}(s_T)$ represents *On-Policy Value* at state s_T for the policy π with parameters ϕ . This paper uses the $\hat{R}_t(\tau)$ to approximate the *On-Policy Action-Value* $Q^{\pi_\phi}(s_t, a_t)$.

The advantage $A^{\pi\phi}(s_t, a_t)$ is estimated based on (15), (5), and the output v_t of the network:

$$A^{\pi\phi}(s_t, a_t) \doteq Q^{\pi\phi}(s_t, a_t) - V^{\pi\phi}(s_t) \approx \hat{A}_t = \hat{R}_t(\tau) - v_t. \quad (16)$$

Following the work of Schulman *et al.* [58], this paper optimizes the policy π in (2) by loss $\mathcal{L}^{a_t}(\theta)$:

$$\mathcal{L}^{a_t}(\theta) = -\min \left(\frac{\pi_\theta(a_t|o_t, p_t)}{\pi_{\theta'}(a_t|o_t, p_t)} \hat{A}_t, f_{1-\epsilon}^{1+\epsilon} \left(\frac{\pi_\theta(a_t|o_t, p_t)}{\pi_{\theta'}(a_t|o_t, p_t)} \right) \hat{A}_t \right), \quad (17)$$

where $\pi_{\theta'}(a_t|o_t, p_t)$ denotes the probability of action a_t at state (o_t, p_t) following the old policy $\pi_{\theta'}$. It comes from rollouts and stays still during the training process. The $\pi_\theta(a_t|o_t, p_t)$ is a new probability under the new policy π with parameters θ . The π_θ is updated once during one training epoch. It is possible to demonstrate that $E[-\mathcal{L}^{a_t}(\theta)]$ and (2) have the same gradient [58].

The output v_t (\bar{v}_t) of the Local Policy (Global Policy) is intended to estimate the *On-Policy Value* of current state. As a result, this paper includes a squared-error loss $\mathcal{L}^{VF}(\phi)$:

$$\mathcal{L}^{VF} = \left(v_t - V^{\pi\phi}(s_t) \right)^2 \approx \left(v_t - \hat{R}_t(\tau) \right)^2, \quad (18)$$

where $\hat{R}_t(\tau)$ is calculated using (15) from the collected data in rollouts. It should be noted that calculating the value of $V^{\pi\phi}(s_t)$ from the collected data is difficult and impractical. Achiam *et al.* [59] demonstrated that $V^{\pi^*}(s) = \max_a Q^{\pi^*}(s, a)$. Thus, in (18), this paper uses $\hat{R}_t(\tau)$ to approximate the $V^{\pi\phi}(s_t)$.

To decrease the uncertainty of the predicted distribution $\pi_\theta(\cdot|o_t, p_t)$, this paper adds an entropy loss $\mathcal{L}^{\pi_\theta}(s_t)$:

$$\mathcal{L}^{\pi_\theta}(s_t) = -E_{a_t \in A(s_t)} [\pi_\theta(a_t|o_t, p_t) \log(\pi_\theta(a_t|o_t, p_t))]. \quad (19)$$

Summarily, loss for policy π (Global Policy π^l or Local Policy π^s) with parameters θ is defined based on (17), (18), and (19):

$$\mathcal{L}_{LP}(\theta, t) = \mathcal{L}^{a_t}(\theta) + c_1 \mathcal{L}^{VF}(\theta) + c_2 \mathcal{L}^{\pi}(s_t|\theta), \quad (20)$$

where c_1, c_2 are coefficients, say $c_1 = 0.5, c_2 = 0.001$.

2) SUPERVISED LEARNING

The Mapper is optimized by supervised learning every 100 time steps. The ground truth is generated from the depth information provided by the virtual environment. Firstly, the world coordinate (x, y, z) of a point is computed from the depth image using the standard camera model. The agent is at the origin of this coordinate system, facing the z axis with x axis leftward. The y axis represents the height. Secondly, the ground truth egocentric map m_t^{ego-st} is generated by:

$$m[x', y'] = \begin{cases} 2, & \exists y \text{ s.t. } H > y > L \text{ if } x = x', z = y' \\ 1, & \exists y \text{ s.t. } L > y \text{ if } x = x', z = y' \\ 0, & \text{otherwise} \end{cases}$$

$$m_t^{ego-st}[x', y'] = \text{stack} \{m[x', y'] = 2, m[x', y'] > 0\}$$

where H and L are the thresholds of the observation height, say $H = 1.5, L = 0.2$. It should be noted that the computed world coordinates (x, y, z) describe the observed surface of obstacles. $m[x', y'] = 2$ means the grid (x', y') in an egocentric map is occupied by obstacles. $m[x', y'] = 1$ means the free area. Thus, the $m[x', y'] > 0$ means the explored area. Finally, $m_t^{ego-st}[x', y']$ consists of two channels, one for the occupied area and the other for the explored area.

This paper uses Binary Cross Entropy (BCE) loss for training the Mapper, as Ramakrishnan *et al.* [28]:

$$\mathcal{L}_M = \frac{1}{\|m_t\|} \sum_{x \in m_t, y \in m_t^{gt}} [y \cdot \log(x) + (1 - y) \cdot \log(1 - x)].$$

D. TRANSFERRING THE TRAINED MODEL ONTO THE REAL ROBOT

To examine the performance of the proposed network on a real robot, this paper took three steps: 1) train the model with the observation height set to 21 cm (the height of the camera on the real robot); 2) build the same network on the Jetson Nano (the microprocessor unit of the real robot) and load the trained model; 3) run the customized code to make use of the loaded network and control the real robot to move.

V. EXPERIMENT

A. EXPERIMENTAL SETUP

1) DATASETS

This paper uses the Habitat [26] simulator along with Gibson [35] datasets and Matterport3D datasets [36]. Gibson contains large-scale photo-realistic 3D indoor environments and simulates the first-person observations and actions of a robotic agent embodied in these environments. Our observation space consists of a 128×128 RGB-D observation (considering the real-time performance) and a position sensor. The sensor reading denotes the change in the agents' pose x, y, θ . This paper simulates noisy action and sensor reading for realistic evaluation, as with Ramakrishnan *et al.* [28] and Chaplot *et al.* [43]. The datasets consist of 72 scenes with 4,932,479 episodes. The evaluation datasets contain 944 episodes. This paper used 495 test episodes from Matterport3D datasets. Note that the scenes in Matterport3D did not appear during the training procedure.

2) METRICS

This paper uses Intersection over Union (IoU) as the main metric to measure exploration performance. The Mapper outputs an occupancy grid map. Each grid cell has three states: unknown, free (passable) and occupied. Thus, this paper uses free IoU to measure the accuracy of the free area, occupied IoU to measure that of the occupied area, and mean IoU to measure overall performance. Note that the unknown area is not included in the computation. For the experiment that is conducted in the real world, there is no layout of

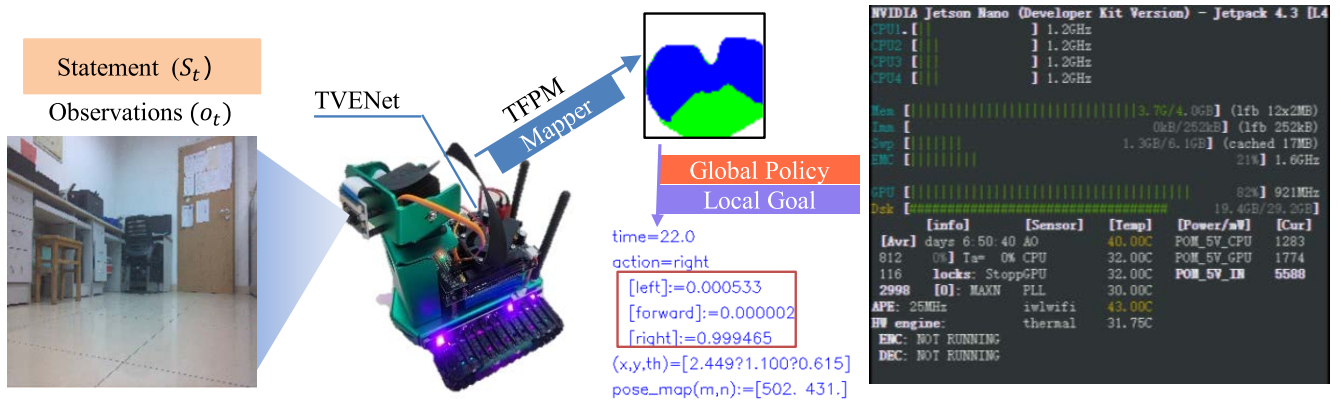


FIGURE 6. Experiment illustration for Nvidia-Jetbot. It should be noted that the value in the red box is the probability distribution of the action, and the final action is sampled based on that. This would be helpful for the algorithm to balance exploration and exploitation.

the environment and no ground truth for the global map. Thus, this paper shows the time consumption of the inference procedure on the real robot.

3) ROBOT

This paper used Nvidia Jetbot² to deploy the proposed TVENet. The Jetbot is a kind of crawler robot with one RGB camera. The process unit is the Nvidia Jetson nano developer kit.³ It has 4GB of RAM and is enough for TVENet. The architecture of the Jetson nano is ARMv8-AArch64, so installing the Pytorch with CUDA support would take some effort. The system information of the Jetbot is illustrated on the right side of Fig. 6.

B. BASELINES

This paper defines baselines based on prior work. None of these baselines conducted experiments on a real robot to prove the feasibility of the Embodied AI approach.

- **OccAnt-re**: the baseline that is reported in the paper [28]. It is the winning entry in the 2020 Habitat PointNav Challenge.⁴
- **OccAnt-ckpt**: the baseline network that is evaluated with the released pre-trained checkpoint [28]. This is to analyze the impact of a hardware platform on the evaluation procedure.

This paper implements the baseline on top of the ANS [43] framework, which is the same as Ramakrishnan *et al.* [28]. Our goal is to show the impact of our TVENet model while fixing the policy learning approach across methods for a fair comparison. This paper considers two new trained models:

- **TVENet**: our model based on the backbone of TFPM.
- **OccAnt-train**: the model that is trained on our hardware platform with the architecture and training datasets the

²<https://www.nvidia.cn/autonomous-machines/embedded-systems/jetbot-ai-robot-kit/>

³<https://www.nvidia.cn/autonomous-machines/embedded-systems/jetson-nano/>

⁴<https://aihabitat.org/challenge/2020/>

TABLE 1. The time consumption for the real world experiment.

	Time Consume (s)			
	necessary		visualization	
	prepare input	inference	record	image write
Computer (RTX 3090)	0.0026	0.1148	0.0391	0.2088
Jetbot (Jetson Nano)	0.0066	1.5100	0.2040	2.2500

same as Ramakrishnan *et al.* [28]. This is to analyze the impact of a hardware platform and training scale on training procedure.

C. EXPERIMENTAL RESULTS

1) SIMULATION EXPERIMENT: PROPOSED NETWORK WORKS

The evaluation of the proposed network is conducted in a simulation environment. The results are shown in Fig. 7. The TVENet takes a sequence of RGB images as inputs and outputs predicted maps. These predicted maps include the egocentric map and the allocentric map. The findings demonstrate that the TVENet can extract geometry information from RGB images and make decisions based on the current observation and navigation history.

2) REAL WORLD EXPERIMENT: EMBODIED AI METHOD WORKS

To prove the feasibility of the Embodied AI approach, this paper deployed the TVENet on a real robot. The illustration of an experiment conducted in the real environment is shown in Fig. 6. The robot captures the RGB image as the current observation o_t , feeds the current statement s_t into TVENet, samples the action a_t based on the probability distribution of the action space $\pi_{\theta}^s(\cdot|o_t, g_t^s)$, and conducts the action. The statements s_t contains two images o_t, o_{t-1} and the last action a_{t-1} . The experimental results are shown in Fig. 8. The first column shows the RGB input at time t , the second column shows the egocentric map predicted by the on-board

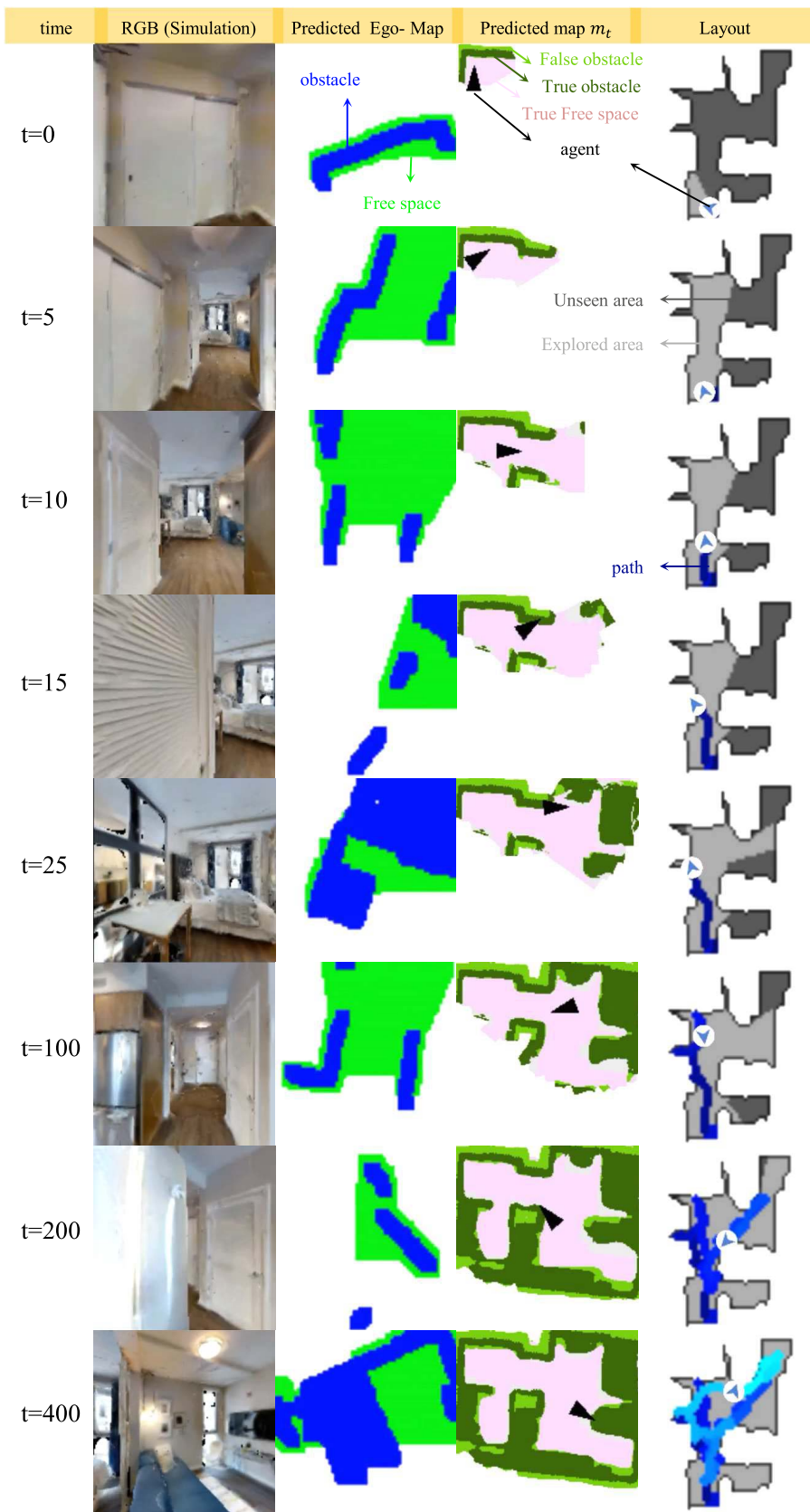


FIGURE 7. An illustration of the updating process of the predicted map. As time goes by, the predicted map is updated by the last allocentric map m_{t-1} and the current egocentric map m_t^{ego} . The processing flow is shown in Fig. 3.



FIGURE 8. Experimental results in the real world. Each experiment conducted 500 time steps (one time step is about 1.5 seconds, which is the processing time of the TVENet on the robot.) This figure shows five time steps in different experiments.

TVENet, and the third column shows the current output of the TVENet. The tuple (x, y, th) represents the estimated position (x, y) in the world coordinate system and the relative orientation θ with respect to the initial orientation. The tuple (m, n) represents the estimated position in the map coordinate system. This network (TVENet) is trained in a photo-realistic simulation environment and tested in real world scenarios that the network has never seen before. The results prove the feasibility of deploying a learning-based perception-decision network on a real robot for exploration tasks. The time consumption is listed in Table 1.

3) PERFORMANCE COMPARISON: TVENet IS BETTER

The comparison between the baselines and the trained models is shown in Table 2. The length of each episode for training and evaluation are set to 1000 time steps ($T_{EXP}=1000$) and 500 time steps, respectively. The comparison illustrates that our TVENet achieves higher-precision mapping performance than the baselines.

When comparing the performance of OccAnt-ckpt and OccAnt-train, this paper finds that the performance

could be better despite using less training scale (nearly 40% of the replay size, mapper batch size, and process number). They have the same network architecture and training process. The only differences are training scale and platform characteristics (version of GPUs and third-party libraries). From the Table 4, this paper notices that too few training scales would destroy the performance of OccAnt-train. Thus, this paper thinks that the training scale of OccAnt-train in Table 2 is enough for training the network to achieve similar performance to the baseline. The platform characteristic further increases performance a little.

Therefore, better network architecture plays a more important role than larger training scale (or GPU memory capability) to further improve the performance.

When comparing the performance of OccAnt-ckpt and TVENet, this paper discovered that the proposed network, TVENet, could outperform the baseline by 5.31%(+0.032). Even compared with OccAnt-train, our TVENet could enhance the performance and predict a more accurate map.

The qualitative comparison of TVENet, OccAnt-train, and OccAnt-ckpt is shown in Fig.9. Notice that they are evaluated over the full evaluation episodes (944) on the same hardware platform. Some snapshots of experimental results are shown in Fig.10. Similar to the configuration mentioned above, Fig. 10 shows the RGB inputs, the ground truth environment layout with the robot trajectory (visited point sequence, not the rollout mentioned above), and the predicted global maps. Note that scenes in Matterport3D are much larger than in Gibson, and the layout in the figure is scaled to the same size for display.

4) EXCHANGE STUDY: TFPM IS EFFECTIVE

The exchange study was conducted to analyze the impact of the proposed TFPM, which is the main part of Mapper. As was mentioned above, there are three group parameters: Global Policy, Local Policy, and Mapper. They are trained simultaneously, so exchange study is significant. This paper exchanges the Mapper part of OccAnt-ckpt and TVENet to get Mix 1 and Mix 2. These four networks are evaluated on the same evaluation datasets and the results are listed in Table 3. The qualitative comparison of the TFPM is shown in Fig.11. To illustrate clearly, Fig. 11 shows the IoU of 50 (first 50 of 944 episodes) episodes on the bottom. They are enough for case analysis. At the top of the Fig.11, each row represents an episode which is tested by four models. They are selected to illustrate the fact that each model has its benefits, and the essential differences are marked in a red circle. The mean IoU of each model is ranked and listed on the right side of each episode (each row).

Comparing the performance of Mix 1 and OccAnt-ckpt, this paper finds that TFPM could enhance the map accuracy by 9.55% (+0.063). Intuitively, equipped with the proposed TFPM, the network could perceive a more precise boundary of the obstacle as shown in the first row in Fig.11. The only difference between these two methods is the Mapper. That

TABLE 2. Performance comparison. The first three rows are the baselines reported by Ramakrishnan et al. [28]. The OccAnt-train is trained on our platform with the architecture staying still. The TVENet is our new proposed model. Training details include: size of the episode for training (T_EXP), replay size (RS), Mapper batch size (MBS), and process number (PN).

Method	training details				evaluation (IoU)					
	Episode length	replay size	map batch	processes	Gibson			Matterport3D		
					mean	free	occ	mean	free	occ
OccAnt(rgb)-re	1000	100,000	420	36	46.1	44.4	47.9	22.0	-	-
OccAnt(rgb)-re	1000	100,000	420	36	56.5	51.5	61.5	23.0	-	-
OccAnt-ckpt	1000	100,000	420	36	60.0	58.6	61.4	-	-	-
OccAnt-train	1000	40,000	180	16	62.7	62.4	63.1	-	-	-
TVENet	1000	40,000	180	16	63.2	62.9	63.5	27.9	28.7	27.1

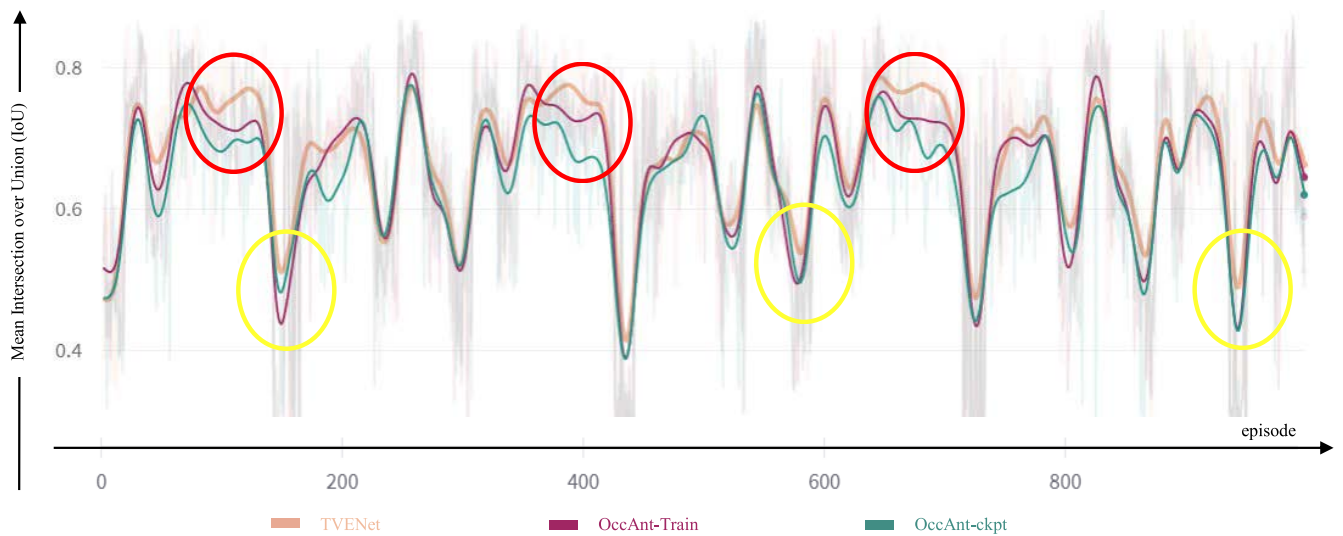


FIGURE 9. The Mean Intersection over Union (IoU) of the maps that are generated by TVENet, OccAnt-train, and OccAnt-ckpt. The horizontal ordinate represents the index of the test episode. The initial location is different in each test episode to evaluate the performance of the network. The yellow (red) circles mark the hard (easy) test episode. They occurred because of the randomized initial location. TVENet has the largest mean IoU of all marked episodes. Therefore, the performance of OccAnt-train is worse than our TVENet and better than OccAnt-ckpt.

TABLE 3. Exchange study. Quantitative analysis of the TFPM in TVENet.

Method	training details		evaluation (IoU)		
	Policy	Mapper	mean	free	occ.
OccAnt-ckpt	OccAnt-ckpt	OccAnt-ckpt	0.600	0.586	0.614
TVENet	TFPM	TFPM	0.632	0.629	0.635
Mix 2	TFPM	OccAnt-ckpt	0.584	0.575	0.593
Mix 1	OccAnt-ckpt	TFPM	0.663	0.656	0.671

means the proposed TFPM is effective for visual exploration. This conclusion is consistent when comparing TVENet and Mix 2.

From Fig. 11, this paper found that the rank of performance for these four methods varies among different episodes. Certainly, the rank of average performance for these methods is: Mix1 > TVENet > OccAnt-ckpt > Mix 2.

5) ABLATION STUDY: TRICKS ARE USEFUL

The ablation study was conducted to analyze the impact of tricks (Training Mapper First (TMF) and Auxiliary

Task (Aux)). There are three versions of OccAnt-train and TVENet. Each model is trained to get more than 11 checkpoints. Each checkpoint is evaluated over the same evaluation datasets. The best result of each version is listed in Table 4.

Comparing three versions of OccAnt-train, this paper finds that a larger training scale leads to better performance. In particular, Process Number (PN) enhances the accuracy of the map by a large margin. For example, $PN = 16$ means that there are 16 processes on the GPU. Each process conducts a different environment rendering mission. Therefore, there are 16 visual observations fed into the network at the same time. Abundant samples reduce the random error during gradient computation.

Comparing three versions of TVENet, this paper finds that tricks (TMF and Aux) both have a positive impact on the performance. They enhance the accuracy by 9.66% (+0.56). Our Auxiliary Task provides the model with the ability to perceive semantic information. The results of this

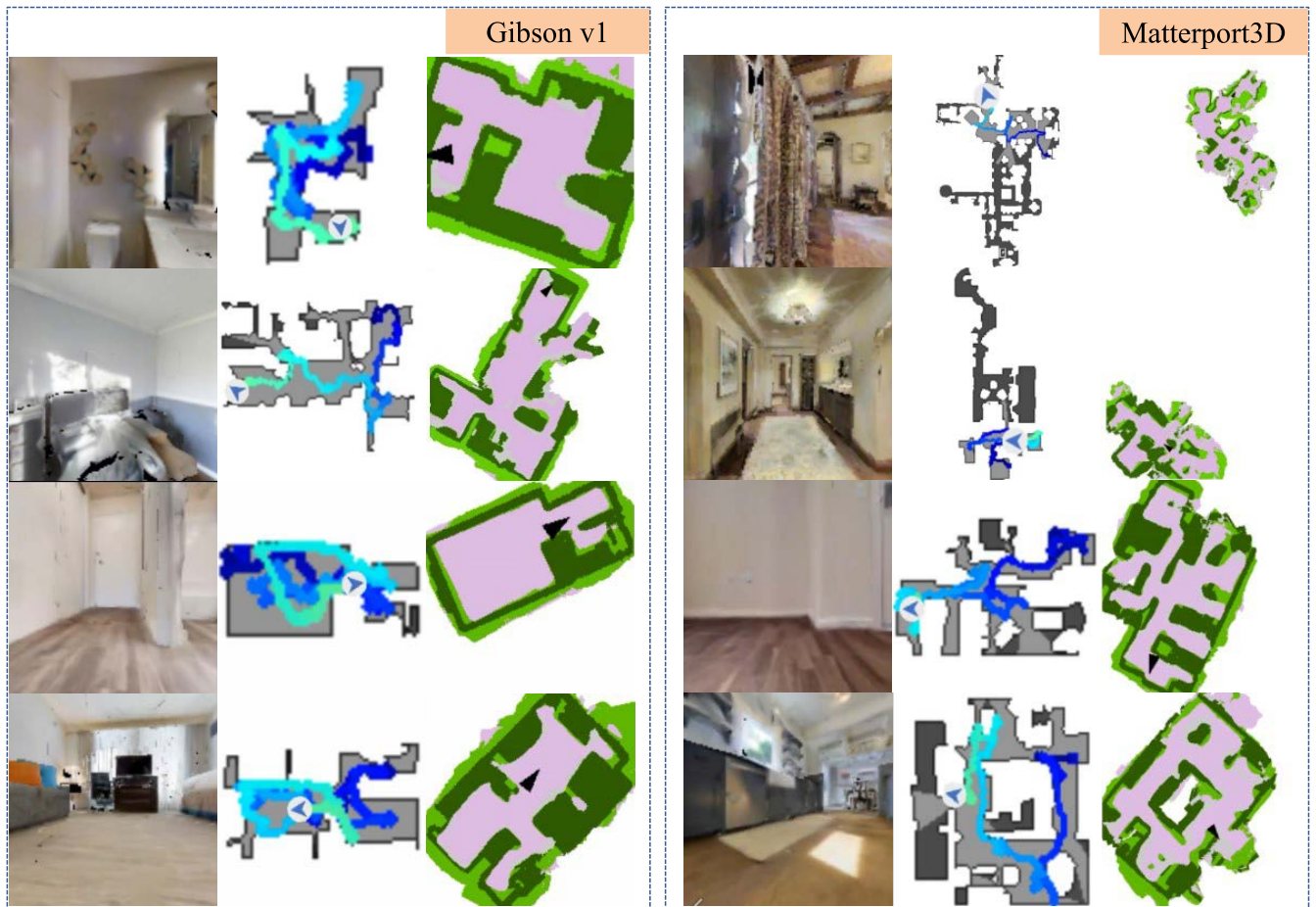


FIGURE 10. Experimental results of TVENet in the photo-realistic simulation environment.

TABLE 4. Ablation study. The best results are marked in bold. It shows that Training Mapper First (TMF) and Auxiliary Task (Aux) both improve the performance of TVENet. The bottom three rows demonstrate that increasing the Replay Size(RS), Mapper Batch Size (MBS), and Processes Number (PN) results in better performance.

Method	training details						evaluation (IOU)		
	M.F.	Aux.	T_EXP	replay size	map batch	processes	mean	free	occ.
TVENet	N	N	1000	40,000	180	16	0.576	0.577	0.575
	Y	N	1000	40,000	180	16	0.595	0.587	0.602
	Y	Y	1000	40,000	180	16	0.632	0.629	0.635
OccAnt-train			1000	40,000	180	1	0.385	0.372	0.398
			1000	500	42	1	0.364	0.385	0.344

experiment illustrate that semantic information enhances visual exploration performance.

VI. DISCUSSION AND FUTURE WORK

The auxiliary task only enables the TVENet with the ability of semantic information perception. The depth information is also important. Therefore, the future work is to add more auxiliary tasks to enable the TVENet with more ability. Current excellent work in monocular depth estimation could be the key.

The proposed TVENet is trained in indoor environments. The ground truth is generated based on the distance to the walls. However, this paradigm has potential. Using other approaches to generate the ground truth in outdoor environments, the trained model may have the ability to work in an unstructured field environment.

Besides the combination of multi-scale features, the Transformer technology may also work well with the combination of multi-temporal features. TVENet used GRU based on the visual observation o_t and the old memory (hidden recurrent

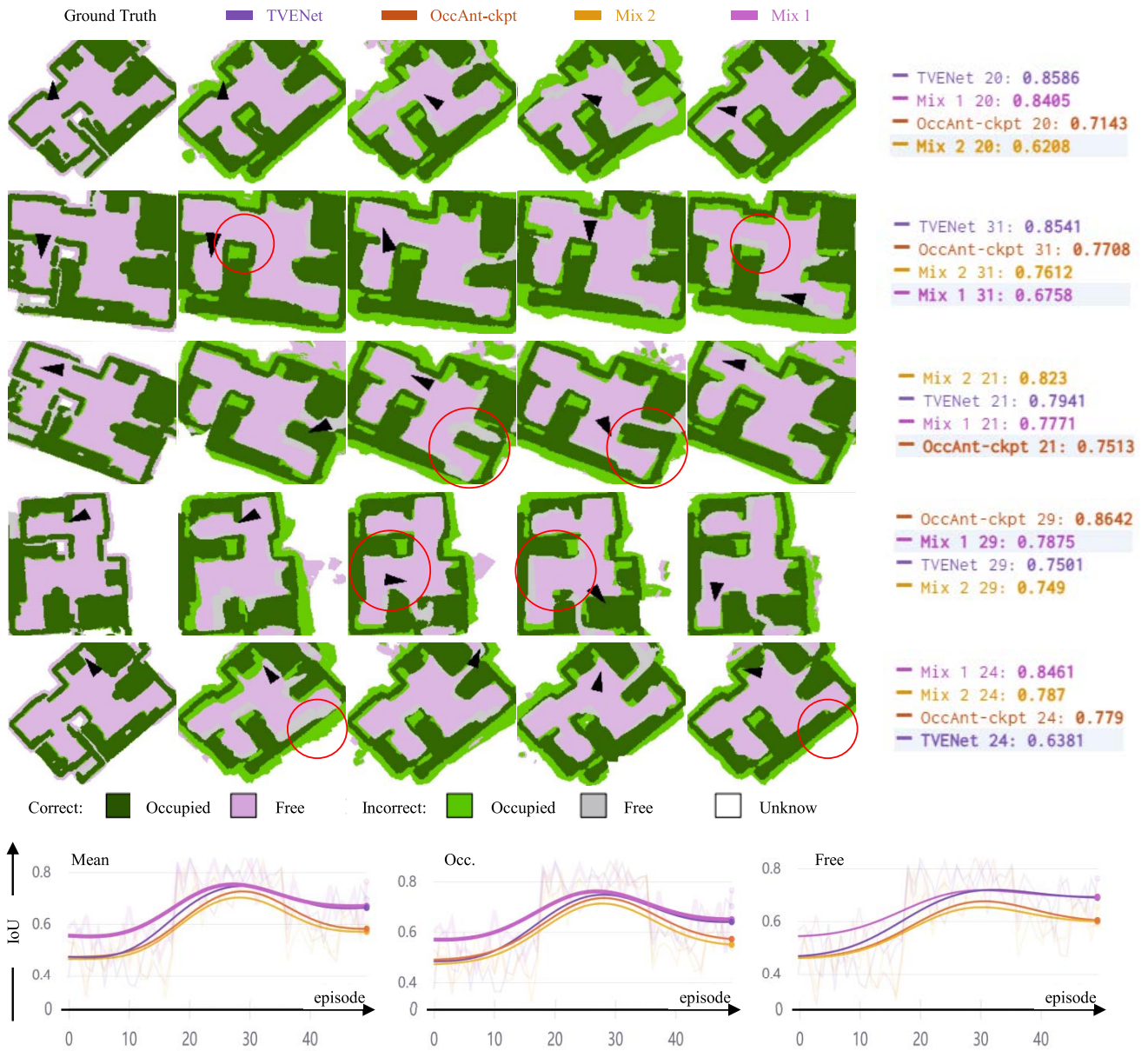


FIGURE 11. Exchange study. Qualitative analysis of the TFPM in TVENet.

states) HS_t at time step t to achieve the new memory HS_{t+1} . Besides the hidden recurrent states, the memory could also be represented by a list of observations that contains more raw information. These multi-temporal features could be handled by Transformer.

From simulation into reality is the main direction of our future work. This paper finds the OccAnt, ANS, and TVENet are all sensitive to the height of the camera. Thus, this paper needs to train the model at the corresponding height before deploying it on a real robot. If there is a network that is insensitive to the height of the camera, or if there is a height-invariant visual feature, the deployment will be simplified.

VII. CONCLUSION

This paper proposes a modular network (TVENet) for visual exploration of mobile robots. The novel module (TFPM) in TVENet leverages the strength of Transformer-based architecture and self-attention mechanism. It improves robotic perception ability and generates a more accurate occupancy grid map in visual exploration. Moreover, this paper introduces two tricks for the training process: Training Mapper First (TMF) and Auxiliary Task (Aux). Through these two tricks, the TVENet increases the accuracy of the occupancy map by a large margin. The benefits and effectiveness of TVENet, as well as two training tricks, are clearly demonstrated in our three experiments.

REFERENCES

- [1] S. K. Ramakrishnan, D. Jayaraman, and K. Grauman, "An exploration of embodied visual exploration," *Int. J. Comput. Vis.*, vol. 129, no. 5, pp. 1616–1649, May 2021, doi: [10.1007/s11263-021-01437-z](https://doi.org/10.1007/s11263-021-01437-z).
- [2] D. S. Chaplot, E. Parisotto, and R. Salakhutdinov, "Active neural localization," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018. [Online]. Available: https://openreview.net/forum?id=ry6-G_66b
- [3] S. K. Ramakrishnan, D. Jayaraman, and K. Grauman, "Emergence of exploratory look-around behaviors through active observation completion," *Sci. Robot.*, vol. 4, no. 30, May 2019. [Online]. Available: <https://robotics.sciencemag.org/content/4/30/eaaw6326>
- [4] P. Anderson, A. Chang, D. Singh Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," 2018, *arXiv:1807.06757*.
- [5] F. Zhu, Y. Zhu, X. Chang, and X. Liang, "Vision-language navigation with self-supervised auxiliary reasoning tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10012–10022.
- [6] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 237–267, Feb. 2002.
- [7] G. Wells, C. Venaille, and C. Torras, "Vision-based robot positioning using neural networks," *Image Vis. Comput.*, vol. 14, no. 10, pp. 715–732, Dec. 1996. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0262885696890226>
- [8] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/9440682/>
- [9] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011. [Online]. Available: <https://ieeexplore.ieee.org/document/6096039/>
- [10] J. B. Hayet, F. Lerasle, and M. Devy, "A visual landmark framework for mobile robot navigation," *Image Vis. Comput.*, vol. 25, no. 8, pp. 1341–1351, Aug. 2007. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S026288560600268X>
- [11] N. Ohnishi and A. Imiya, "Appearance-based navigation and homing for autonomous mobile robot," *Image Vis. Comput.*, vol. 31, nos. 6–7, pp. 511–532, Jun. 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0262885612002120>
- [12] K. McGuires, G. de Croon, C. D. Wagner, K. Tuyls, and H. Kappen, "Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 1070–1076, Apr. 2017. [Online]. Available: <http://ieeexplore.ieee.org/document/7833065/>
- [13] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation," 2019, *arXiv:1903.01959*.
- [14] J. Duan, S. Yu, H. Li Tan, H. Zhu, and C. Tan, "A survey of embodied AI: From simulators to research tasks," 2021, *arXiv:2103.04918*.
- [15] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer, "Vision-and-dialog navigation," in *Proc. Conf. Robot Learn.*, May 2020, pp. 394–406. [Online]. Available: <http://proceedings.mlr.press/v100/thomason20a.html>
- [16] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sunderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3674–3683. [Online]. Available: <https://ieeexplore.ieee.org/document/8578485/>
- [17] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1–10.
- [18] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "IQA: Visual question answering in interactive environments," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4089–4098. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Gordon_IQA_Visual_Question_CVPR_2018_paper.html
- [19] F. Zhu, X. Liang, Y. Zhu, Q. Yu, X. Chang, and X. Liang, "SOON: Scenario oriented object navigation with graph-based exploration," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 12689–12699. [Online]. Available: https://openaccess.thecvf.com/content_CVPR2021/html/Zhu_SOON_Scenario_Oriented_Object_Navigation_With_Graph-Based_Exploration_CVPR_2021_paper.html
- [20] P. Roy and C. Chowdhury, "A survey of machine learning techniques for indoor localization and navigation systems," *J. Intell. Robot. Syst.*, vol. 101, no. 3, p. 63, Mar. 2021. [Online]. Available: <http://link.springer.com/10.1007/s10846-021-01327-z>
- [21] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," in *Proc. Conf. Robot Learn.*, May 2020, pp. 420–429. [Online]. Available: <http://proceedings.mlr.press/v100/bansal20a.html>
- [22] P. Mirowski, M. Koichi Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell, "Learning to navigate in cities without a map," 2018, *arXiv:1804.00168*.
- [23] T. Zhang, X. Hu, J. Xiao, and G. Zhang, "A machine learning method for vision-based unmanned aerial vehicle systems to understand unknown environments," *Sensors*, vol. 20, no. 11, p. 3245, Jun. 2020.
- [24] D. Mishkin, A. Dosovitskiy, and V. Koltun, "Benchmarking classic and learned navigation in complex 3D environments," 2019, *arXiv:1901.10915*.
- [25] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3357–3364.
- [26] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied AI research," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 9338–9346. [Online]. Available: <https://ieeexplore.ieee.org/document/9010745/>
- [27] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017. [Online]. Available: <http://ieeexplore.ieee.org/document/7946260/>
- [28] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman, "Occupancy anticipation for efficient exploration and navigation," in *Proc. Comput. Vis. (ECCV) (Lecture Notes in Computer Science)*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer, 2020, pp. 400–418.
- [29] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhou, and K. Zieba, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.
- [30] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," 2018, *arXiv:1803.00653*.
- [31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Feb. 2016, pp. 770–778. [Online]. Available: <http://ieeexplore.ieee.org/document/7780459/>
- [32] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," 2020, *arXiv:2012.12556*.
- [33] Q. Zhang and Y. Yang, "ResT: An efficient transformer for visual recognition," 2021, *arXiv:2105.13677*.
- [34] D. Gordon, A. Kadian, D. Parikh, J. Hoffman, and D. Batra, "SplitNet: Sim2Sim and Task2Task transfer for embodied visual navigation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1022–1031.
- [35] F. Xia, A. Zamir, Z. He, S. Sax, J. Malik, and S. Savarese, "Gibson ENV: Real-world perception for embodied agents," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9068–9079. [Online]. Available: <https://ieeexplore.ieee.org/document/8579043/>
- [36] 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," 2017, *arXiv:1709.06158*.
- [37] H. Li, Z. Qichao, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2019.
- [38] M. Luong and C. Pham, "Incremental learning for autonomous navigation of mobile robots based on deep reinforcement learning," *J. Intell. Robot. Syst.*, vol. 101, no. 1, p. 1, 2021, doi: [10.1007/s10846-020-01262-5](https://doi.org/10.1007/s10846-020-01262-5).
- [39] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for UAVs: Current developments and trends," *J. Intell. Robot. Syst.*, vol. 87, no. 1, pp. 141–168, Jul. 2017. [Online]. Available: <http://link.springer.com/10.1007/s10846-017-0483-z>

- [40] Z. Ren, J. Lai, Z. Wu, and S. Xie, "Deep neural networks-based real-time optimal navigation for an automatic guided vehicle with static and dynamic obstacles," *Neurocomputing*, vol. 443, pp. 329–344, Jul. 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231221002800>
- [41] T. Zhang, X. Hu, J. Xiao, G. Zhang, and L. Fu, "An implementation of non-electronic human-swarm interface for multi-agent system in cooperative searching," in *Proc. IEEE 15th Int. Conf. Control Autom. (ICCA)*, Jul. 2019, pp. 1355–1360.
- [42] A. Narayan, E. Tuci, F. Labrosse, and M. H. Mohammed Alkilabi, "A dynamic colour perception system for autonomous robot navigation on unmarked roads," *Neurocomputing*, vol. 275, pp. 2251–2263, Jan. 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231217317228>
- [43] D. Singh Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural SLAM," 2020, *arXiv:2004.05155*.
- [44] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," 2018, *arXiv:1803.00653*.
- [45] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 7272–7281. [Online]. Available: <http://ieeexplore.ieee.org/document/8100252/>
- [46] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological SLAM for visual navigation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12875–12884. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Chaplot_Neural_Topological_SLAM_for_Visual_Navigation_CVPR_2020_paper.html
- [47] J. F. Henriques and A. Vedaldi, "MapNet: An allocentric spatial memory for mapping environments," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8476–8484. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Henriques_MapNet_An_Allocentric_CVPR_2018_paper
- [48] F. Li, C. Guo, B. Luo, and H. Zhang, "Multi goals and multi scenes visual mapless navigation in indoor using meta-learning and scene priors," *Neurocomputing*, vol. 449, pp. 368–377, Aug. 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231221004707>
- [49] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (Adaptive Computation and Machine Learning Series), 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
- [50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
- [51] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. Nat. Acad. Sci. USA*, vol. 93, no. 4, pp. 1591–1595, 1996. [Online]. Available: <https://www.pnas.org/content/93/4/1591>
- [52] K. Cho, B. Van Merriënboer, C. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar: ACL, Oct. 2014, pp. 1724–1734. [Online]. Available: <https://doi.org/10.3115/v1/d14-1179>, doi: 10.3115/v1/d14-1179.
- [53] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, Honolulu, HI, USA, Jul. 2017, pp. 936–944. [Online]. Available: <http://ieeexplore.ieee.org/document/8099589/>
- [54] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," 2015, *arXiv:1505.04597*.
- [55] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "OpenNMT: Open-source toolkit for neural machine translation," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, M. Bansal and H. Ji, Eds. Vancouver, BC, Canada: Association for Computational Linguistics, Jul./Aug. 2017, pp. 67–72. [Online]. Available: <https://doi.org/10.18653/v1/P17-4012>, doi: 10.18653/v1/P17-4012.
- [56] J. Lei Ba, J. Ryan Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
- [57] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.
- [58] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
- [59] J. Achiam, "Spinning up in deep reinforcement learning," 2018. [Online]. Available: <https://spinningup.openai.com/en/latest/user/introduction.html> and <https://github.com/openai/spinningup>



TIANYAO ZHANG received the B.S. degree from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, in 2018, where he is currently pursuing the Ph.D. degree with the School of Automation Science and Electrical Engineering. His research interests include computer vision, unmanned aerial vehicles, object detection, and deep learning algorithm.



XIAOGUANG HU received the B.S. degree from Northeast Dianli University, Jilin, China, in 1983, the M.S. degree from the Wuhan University of Hydraulic and Electric Engineering, Wuhan, China, in 1997, and the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2003. Currently, she is a Professor as the School Party Secretary with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. Her current research interests include image processing, embedded test systems, and smart grid.



JIN XIAO received the B.S. degree from Hunan University, Changsha, China, in 2005, and the M.S. and Ph.D. degrees in detection technology and automation devices from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, in 2008 and 2015, respectively. Currently, she is an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University. She has authored over 40 journals and conference papers. Her current research interests include image processing, pattern recognition, and embedded test systems. She has presided over 20 research projects of the National Natural Science Foundation of China, Aerospace Science and Technology Foundation, and Aviation Science Foundation.



GUOFENG ZHANG received the B.S., M.S., and Ph.D. degrees from the Harbin Institute of Technology, Harbin, China, in 1984, 1987, and 1996, respectively. Currently, he is an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His current research interests include robust control, virtual reality, computer control, and simulation.

...