2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Memory-based Exploration-value Evaluation Model for Visual Navigation

Yongquan Feng[†1], Liyang Xu[†1], Minglong Li[1], Ruochun Jin[1], Da Huang[⋆1], Shaowu Yang[1] and Wenjing Yang[1]

***Abstract*— We propose a hierarchical visual navigation solution, called Memory-based Exploration-value Evaluation Model (MEEM), to improve the agent's navigation performance. MEEM employs a hierarchical policy to tackle the challenge of sparse rewards, holds an episodic memory to store the historical information of the agent, and applies an Exploration-value Evaluation Model to calculate an exploration-value for action planning at each location in the observable area. We experimentally verify MEEM by navigation performance comparison on two datasets including the grid-map dataset and the 3D scenes Gibson dataset, where our approach achieves state-of-the-art performance on both. Specifically, the overall success rate of MEEM is 95% on the grid-map dataset while the best competitor reaches 68% only. As for the Gibson dataset, the success rate of ours and the best competitor SemExp are 69.8% and 54.4%, respectively. Ablation analysis on the tile-map dataset indicates that all three components of MEEM have positive effects.**

I. INTRODUCTION

Autonomous navigation is fundamental and essential for intelligent navigable agents in the real world [1]. As a popular navigation approach, visual navigation requires the agent to reach the target within a limited number of actions based on visual inputs only, where the policy that defines the agent's way of behaving decides the navigation performance. To optimize the policy, researchers have introduced Reinforcement Learning (RL) to address this challenge, where the policy is defined as a mapping between the environment state space and the action space based on rewards, and the training process maximizes the accumulated rewards obtained by the agent [2]. Although the rapid development of Deep Reinforcement Learning [3] enables agents to handle challenging tasks such as Atari Arcade Games [4] and 3D games [5], these policies trained in fully observable environments can hardly achieve satisfactory performance in most real-world tasks, such as visual navigation, where the agents behave in partially observable environments based on long-term planning.

The main challenge of visual navigation is that the agent can only capture localized information of its surroundings at each time step, and the agent will lose the goal if it is in an unobservable area. If the goal is missing, the agent cannot act reflexively based on its immediate percepts and fails to optimize the policy properly. Thus, it is crucial to find the goal during visual navigation, which requires the agent to reason and execute plans over an extended time interval.

As mentioned in [6], the ability of planning and reasoning in partially observable environments heavily depends on historical information. Following this approach, researchers have employed historical information to train RL-based visual navigation policies, which fall primarily into two categories. One relies on unstructured general-purpose memory such as LSTM [7], [8], and the other stores historical information in navigation-specific memory structures [9], [10], [11]. For the first category, the historical information stored in general-purpose memory is intrinsically unstable over long period of time due to the heavy recurrent calculation in LSTM [12]. In contrast, the latter can provide stability, and both its form and calculation can be specified, so as to offer more comprehensive and flexible support for the policy learning. However, introducing historical information to visual navigation increases the number of states in the search space of policy training, which induces higher complexity. Thus, it is critical to find a trade-off between RL training cost and effectiveness of historical information for visual navigation. Unfortunately, to the best of our knowledge, no prior work has proposed such trade-off between effectiveness and efficiency.

In order to apply historical information to visual navigation efficiently and effectively, we propose a Memory-based Exploration-value Evaluation Model (MEEM) to improve the agent performance in visual navigation, which includes three components. First, to overcome the challenge of sparse rewards [13], we propose a novel hierarchical policy, which defines a sub-goal space in all reachable locations within the observable area. Inspired by the landmark-based navigation of animals [14], we then design an episodic memory structure, which stores the agent's historical information by selectively saving the coordinates of the locations the agent passes by. Last, an Exploration-value Evaluation Model (EEM) is presented to calculate an exploration-value for each location within the observation area, whose input contains the historical information and the agent sensor map. EEM allows the agent to use historical information for efficient action planning.

Our MEEM makes three contributions to solve visual navigation tasks.

- Our hierarchical policy does not rely on any predefined information about the environment, and theoretically, it

*This work was partially supported by the National Natural Science Foundation of China(No. 91948303-1, No. 62106278, and No. 12002380), the National Key R&D Program of China(No. 2021ZD0140301), and the National University of Defense Technology Foundation(No. ZK20-52)

[†]Equal Contribution, [⋆]Corresponding author

[1]With Institute for Quantum Information State Key Laboratory of High Performance Computing, College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China {fengyongquan12, xuliyang08, liminglong10, jinrc}@nudt.edu.cn, huangda1109@163.com, {shaowu.yang, wenjing.yang}@nudt.edu.cn

is applicable for various scenes, even unfamiliar.
- We propose a novel way to store historical information in visual navigation, which is simple but effective.
- Exploration-value provides the agent another way to evaluate the environment, which makes it perform better action planning.

We demonstrate our approach both in grid-world domains and visually realistic simulation environments, which show that MEEM outperforms prior methods by a significant margin. Specifically, the overall success rate of MEEM is 95% with the others 68% at most on the 16×16 grid-map dataset, and 69.8% for ours, 54.4% for SemExp on the Gibson dataset with 3D scenes.

II. Related Work

A. Hierarchical Policy

Hierarchical policy has been introduced to tackle the challenge of sparse reward in visual navigation, which can further be categorized into two types. One can be referred to as time abstraction [15], which generates a high-level action space based on the features of action sequences, such as option-critic[16]. The other proposes a sub-goal space, where the upper layer plans a sub-goal and the lower layer enables the agent to reach the sub-goal until the overall goal is reached.

Methods using sub-goal-based hierarchical policy dominate visual navigation tasks. To be more specific, both h-DQN [17] and HRL-GRG [11] define sub-goal spaces on the space of entities and relations, and achieve desirable performance in the game of Montezuma's Revenge and visual navigation tasks respectively. However, these methods are limited to practical use in that (1) the location of entities has a huge impact on their navigation performance; (2) the agent must have already understood and identified all entities in the sub-goal space; and (3) the number of entities in the environment must reach a predefined threshold.

We propose a novel hierarchical policy, which defines a sub-goal space in all reachable positions within the observable area. Our method is different from previous work as follows: (1) it does not rely on any predefined information in the environment; (2) the sub-goal is required to be reachable for the agent.

B. Semantic Mapping

Learning visual navigation purely from data in an end-to-end manner is prohibitively expensive, in that it includes mapping, state-estimation and path-planning. Consequently, end-to-end learning work for visual navigation still performs poorly even with millions of frames of experience. Active Neural SLAM [18] builds a top-down explicit 2D obstacle map with a Simultaneous Localization and Mapping (SLAM) module, which performs mapping and state-estimation with SLAM. It not only leads to better sample efficiency but also provides flexibility for input modality. Based on Active Neural SLAM, SemExp[10] encodes semantics into obstacle maps to tackle visual navigation tasks and outperforms prior methods by a significant margin.

We employ semantic mapping to build top-down semantic maps for environments. In contrast to SemExp, MEEM preserves only two types of semantics: goal and non-goal, which can effectively simplify the semantic maps.

C. Memory-based Reinforcement Learning

Memory is crucial for intelligent agents to plan on an extended time interval in visual navigation. An addressable memory is used for first-person-view navigation in 3D environments in [19]. Khan et al. [9] propose a novel architecture named Memory Augmented Control Network (MACN), which combines Value Iteration Network (VIN) [20] with Differentiable Neural Computer (DNC) [21] to learn how to plan in partially observable environments. With a structured scene memory, Wang et al. [22] captures environment layouts and makes long-term planning successfully in Vision-and-Language Navigation. The data in memories of these methods is the metric representation of the historical information, which is processed by some neural networks, thus incomprehensible for humans.

HRL-GRG [11] records the probabilities of different entities being observed simultaneously in the sub-goal space through GRG, which indirectly stores the spatial correlation between different entities. An episodic memory is equipped by SemExp [10], and it records the explored area and the corresponding semantics of the environment in real-time, hence supporting more precise action planning. Although the data in these memories has pritical meaning, only spatial information is left, with temporal information ignored.

Our memory structure has the following characteristics: (1) it stores historical information by recording the agent's historical positions in each episode; (2) it keeps both temporal and spatial historical information; (3) only a tiny proportion of historical locations are recorded, not all of them, which greatly reduces the complexity of historical information.

III. Method

A. Preliminaries

A visual navigation task can be described with a 9-tuple $< S, A, G, O, r, o, \pi, N, \gamma >$. S is the state space and contains the whole environment. A is the action space, including all actions of the agent, and G is the goal state space which a subset of S and includes all states of the agent reaching the goal. o is the observation function, which depends on the observable range of the agent, and O is the observation state space determined by o and S: $O = o(S)$. r is the reward function, which returns 0 for non-goal states and 1 for goal states: $r(s) = 1$ iff $s \in G$. π is the policy of the agent taking action, N is the maximum number of actions, and γ is a discount factor. When the agent takes action according to π, it gives rise to a trajectory as follows

$$\{o(s_0), a_0, o(s_1), a_1, ..., o(s_{T-1}), a_{T-1}, o(s_T)\}, \quad (1)$$

where $\forall i, o(s_i) \in O$. The trajectory terminates when $o(s_T) \in o(G)$ or $T == N$, and returns a discounted reward as (2)

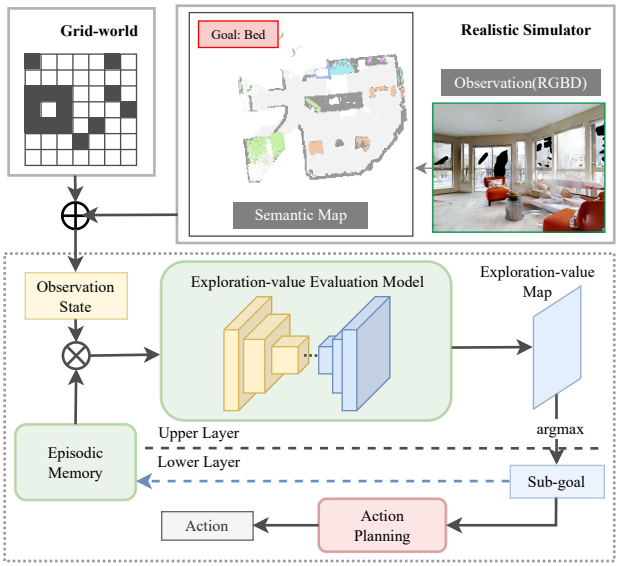

Fig. 1. Memory-based Exploration-value Evaluation Model Overview

$$\mathbb{E}_\pi = \gamma^T r(o(s_T)). \tag{2}$$

Therefore, when $\gamma < 1$, we can optimize the policy π by maximizing $\mathbb{E}_\pi$. q_π is introduced to denote the action-value function [2] for π. According to [2], for fully observable environments, $o(S) == S$, the action-value function of the optimal policy $\pi^\star$ can be expressed as

$$q_{\pi^\star}(s_t, a_t) = r(s_{t+1}) + \gamma \max_a q(s_{t+1}, a). \tag{3}$$

Through the value iteration algorithm[2], we can optimize policy π to approach $\pi^\star$. However, there are still two challenges in visual navigation tasks: sparse rewards and partially observable environments. Our MEEM proposes solutions to address both challenges. Fig. 1 depicts an overview of our method. MEEM makes action planning based on the top-down 2D maps, so when the agent can only get first-person pictures, semantic mapping is employed to generate top-down semantic maps.

B. Hierarchical Policy

Since the number of legal transitions grows exponentially with the distance to the goal, and the agent is only rewarded when reaching the goal, it leads to sparse rewards and causes difficulty for policy training. To mitigate this challenge, we propose a novel hierarchical policy for visual navigation tasks, which defines a sub-goal space Ω on all positions in observation states, as (4).

$$\Omega = \{\omega | \omega \in o(s), \forall s \in S\}. \tag{4}$$

Our hierarchical policy divides the visual navigation task into two stacked sub-tasks, as shown in Fig. 1. The upper layer predicts a sub-goal, and the lower plans a path to it. The trajectory of the upper can be described as

Fig. 2. Different states with the same observation states in grid-world

$$\{o(s_0), \omega_0, o(s_1), \omega_1, ..., o(s_{T'-1}), \omega_{T'-1}, o(s_{T'})\}. \tag{5}$$

The length of (5) can be much shorter than (1). MEEM makes two restrictions on the sub-goal selected by the upper layer: (1) it must be on an unblocked area; (2) it can be reached within a limited number of steps, denoted by N_s. The restrictions bring our MEEM two advantages: (1) they shorten the trajectory while ensuring the frequency of sub-goal planning; (2) both RL methods and planning methods can be used to plan a path to the sub-goal in the lower layer. In our experiments, The lower layer of MEEM employs different methods in different datasets.

C. Episodic Memory Structure

The environments in visual navigation are partially observable, which means $O \neq S$ and $o(s_t) \neq s_t$. Then the agent may get the same observation from different states. Fig. 2 illustrates several situations in grid maps where the agent gets the same observations due to the limitation of the observation range. It brings uncertainty to the optimal action policy $\pi_u^\star$ of the upper layer of MEEM and makes the RL methods in fully observable environments perform poorly in visual navigation. One effective way to reduce the uncertainty is by taking the historical observations of the agent into consideration and planning sub-goals over an extended time. We design an episodic memory structure, M, based on top-down maps of the environments, where MEEM stores the historical information by labeling part of visited positions of the agent.

The introduction of historical information inevitably increases the number of states in the state space, which can make policy training more difficult. It is necessary to simplify historical information while retaining valid information. Our memory does this through four measures:

- Record the location of the goal once the goal is observed. Then the agent can know where it is, even if it disappears from the observable area again.
- Inspired by the landmark-based navigation of animals, our memory only keeps the information of historical sub-goals, not all historical locations, in that the sub-goals selected by the upper of MEEM tend to play a more significant role in navigation than others.
- We simplify the semantic information for top-down maps and keep only two categories of semantics: goal and non-goal. The complexity of the semantic maps can be significantly reduced as a result.

- To encode the temporal information into our memory, we introduce a forgetting factor β to our memory. A negative value is assigned to each historical sub-goal to encourage the agent to explore unvisited areas, which is calculated as

$$M(\omega_i) = -\beta^{t-i}, \quad \forall i, i < t, \tag{6}$$

where t is the current index of the sub-goal.

Based on historical information and observation, MEEM calculates a value for each sub-goal, called exploration-value and denoted by e. Then, the optimal policy of the upper layer is described as

$$\pi_u^\star(o_t) = \arg\max_{\omega}(R_{t+1} + \gamma e(o_{t+1}, \omega)), \tag{7}$$

where $o_t = o([s|M]_t)$, $o_{t+1} = o([s|M]_{t+1})$, and R_{t+1} is the rewards received by the agent from ω_t to ω_{t+1}.

D. Exploration-value Evaluation Model

According to (7), the policy of the upper layer can be trained with the value iteration algorithm based on exploration-value. We propose a model to calculate exploration-values for all positions in observation space, named Exploration-value Evaluation Model (EEM). Our EEM has two characteristics. First, the input of EEM includes the top-down 2D maps centered at the agent and the corresponding historical information stored in our memory, and the output is an exploration-value map. Second, to ensure that each exploration-value has a large enough receptive field, it adopts an encoder-decoder structure for the complex maps, which is similar to U-net[23] but has fewer parameters due to fewer channels. We adopt the Q-learning techniques[4] to update the parameters of EEM θ by (8).

$$\theta \leftarrow \theta - \nabla_\theta(R_{t+1} + \gamma \max_{\omega} e_{t+1} - e_t), \tag{8}$$

where $e_{t+1} = e(o([s|M]_{t+1}), \omega)$ and $e_t = e(o([s|M]_t), \omega_t)$.

Two techniques are adopted to increase the number of transitions in training: (1) All unreachable sub-goals from the current agent position are randomly used to train our EEM, including the locations with obstacles and locations the agent cannot reach within N_s steps. Their target exploration-values are specified as 0. (2) The transition from the previous sub-goal to the current position is used for policy training, even if the current position is not selected by the upper layer. All transitions above can help EEM to calculate exploration-value more accurately.

IV. Experiment

Our experiments include three parts, which are conducted on three different datasets respectively. The first two datasets both belong to the grid world domain, with the difference that the size of maps in the first dataset is only 16×16, while 512×256 in the second. We compare the performance of h-DQN, MACN, and HRL-GRG with our MEEM on the first dataset and perform ablation studies for each component of MEEM on the second, a tile-map dataset which is generated based on the code of [24]. Finally, we train our method on the Gibson[25] dataset in the Habitat simulator[26], which consists of 3D scenes reconstructed from real-world environments. The agent's parameters in our MEEM are kept the same as in the SemExp[10], and we compare the performance of MEEM with SemExp. All codes are implemented using the Pytorch machine learning framework[27], and training is conducted on a server with four NVIDIA Tesla A100 GPUs.

A. Grid Maps

In this experiment, the grid-world maps have a size of 16×16 and come from [11]. There are 120 different grid maps, with 100 of them for training and the remaining 20 for testing. Each map contains 16 goals that simulate entities of the real-world, generated with specific rules, and only 12 of them are used for training. We assume the agent can only observe the window of size 7×7 centered at its position. In every episode, the maximum action number is fixed at 100, and the agent can take one action as moving up/down/left/right at each timestep with forbidden to the blocked area. Success is defined as the agent reaching the position of the designated goal.

Since the size of the observation map is only 7×7, EEM, the upper layer of our MEEM, employs a simple convolutional neural network to calculate exploration-values of all positions within the observable area. It consists of three 3×3 convolutions and two 1×1 convolutions, and each 3×3 convolution is followed by a rectified linear unit (ReLU). The input of EEM contains three channels: the observable environment map, the goal map including historical information provided by our episodic memory, and the agent map indicating the agent's position, and the output is a 7×7 exploration-value map. For the lower layer, we adopt a model based on VIN, whose input has three channels: the environment map, the sub-goal map, and the agent map. Both layers are trained with DQN techniques with ϵ-greedy exploration strategy. We set $\epsilon = 0.9$, $\gamma = 0.95$, $\beta = \sqrt{0.5}$ for our MEEM, and compare our method with the following baseline methods.

- **Random**: The agent always takes a random action.
- **h-DQN**: A widely adopted hierarchical method. It defines the sub-goal space on the 16 goals predefined in grid maps for the upper layer, and both layers are trained with the same DQN techniques.
- **MACN**: We implement MACN according to [19]. The input of this method is the whole environment map with the unobservable area masked. Different from other methods, MACN is trained with supervised learning.
- **HRL-GRG**: It also makes the 16 predefined goals the sub-goal space and records the correlations between different goals with a GRG. According to the open-source code in [11], we implement HRL-GRG on Pytorch and keep all parameters unchanged except for $\epsilon = 0.95$.

For all hierarchical methods, including h-DQN, HRL-GRG, and our MEEM, the lower layer can take at most 10 actions for each sub-goal. We train all methods for 100

TABLE I

THE RESULTS OF ALL METHODS ON THE 16×16 GRID MAPS

Method	Seen Goals			Unseen Goals			Overall		
	SR↑	AS / MS↓	SPL↑	SR↑	AS / MS↓	SPL↑	SR↑	AS / MS↓	SPL↑
Random	0.19	56.05 / 5.42	0.04	0.13	36.00 / 4.46	0.04	0.19	39.79 / 4.95	0.04
h-DQN	0.50	25.50 / 8.19	0.28	0.19	20.49 / 5.58	0.10	0.52	23.74 / 7.68	0.29
MACN	0.26	15.12 / 7.85	0.19	0.28	14.96 / 7.50	0.21	0.37	13.76 / 6.46	0.27
HRL-GRG	0.61	26.58 / 9.17	0.36	0.68	26.23 / 8.85	0.42	0.68	22.88 / 8.20	0.44
Ours	**0.89**	**23.66 / 11.56**	**0.63**	**0.90**	**23.11 / 10.56**	**0.63**	**0.95**	**22.33 / 10.00**	**0.64**

thousand episodes, and during training, the Adam optimizer is adopted with a learning rate of 0.0001 to optimize EEM. Then we evaluate them on the test set with the same starts and goals. The evaluation includes three parts: (1) Seen Goals: the goal is one of 12 goals used for training; (2) Unseen Goals: the goal is one of 4 goals not used in training; (3) Overall: the goal is selected randomly from all predefined goals. We use three metrics for comparing all methods: Success Rate (SR), the ratio of the Average Steps and Minimal Steps over all successful cases (AS/MS), and the Success weighted by inverse Path Length (SPL)[28]. Table I reports the results.

As is shown in Table I, our method outperforms all baseline methods by a significant margin. For h-DQN and HRL-GRG, defining sub-goal spaces on predefined goals simplifies sub-goal selection, but restricts the range of sub-goals, thus less flexibility. In contrast, our MEEM does not rely on any predefined entities or relationships of the environments, making all reachable positions within the observable area a sub-goal space. It enables MEEM choose the sub-goal according to the characteristics of the environment. Moreover, the historical information provided by our episodic memory can assist our method performs better sub-goal selection. MACN can achieve comparable performance to the MHEE during training, but its performance drops significantly in evaluation. In contrast, our approach maintains excellent performance during training and testing, which shows that MEEM has better generalization capabilities for visual navigation tasks.

B. Ablation Studies on Tile-map

In this section, we train our MEEM on tile-maps of size 512×256 and perform ablation studies for each component of our MEEM. Compared with 16×16 grid maps, although the tile-maps belong to the grid-world domain, they are more complex and more similar to real-world top-down 2D maps, as shown in Fig. 3. The tile-map dataset contains 120 maps of size 512×256, 100 of which are used for training our MEEM and the remaining 20 for evaluation.

We specify that the agent can observe a window size of 120×120 centered at its position, and the maximum number of actions is 500. Every step, the agent can move to any location in a 10×10 area centered on itself in a straight line, and if a collision occurs, it stays at the position of collision. When the distance to the goal is less than d_s, the success distance threshold, the agent is identified to reach the goal successfully. $d_s = 5$ in this experiment.

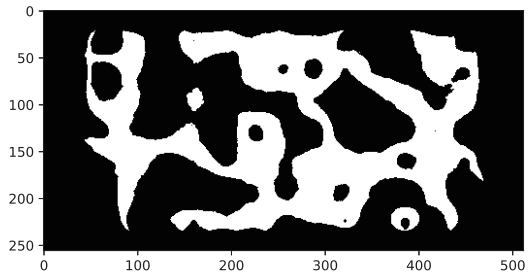

Fig. 3. One tilemap of size 512×256 with white indicates free areas and black obstacles

To calculate the exploration-value map with a size of 120×120, EEM adopts an encoder-decoder structure in this experiment, as illustrated in Fig. 1, which is similar to U-net but with fewer layers and channels. The input channels of EEM keep the same as the previous experiment, and the difference is that the lower layer employs the Fast Marching Method (FMM) [29] to reach the sub-goal, and 25 is the maximum number of movements for each sub-goal. 100 thousand episodes and the Adam optimizer with a learning rate of 0.0001 are used for training, and 1000 episodes for evaluation. Remarkably, due to no predefined sub-goal in tile-maps, h-DQN and HRL-GRG cannot be adopted to solve visual navigation tasks on these maps. Besides SPL and SR, Distance to Success (DTS) [10], the distance to the success threshold boundary when the episode ends, is used to evaluate MEEM.

To understand the importance of three components in MEEM, we consider ablations as follows: (1) Random: take all actions randomly; (2) Random+FMM: adopt our hierarchical policy with the random sub-goal selection in the upper layer and use FMM in the lower; (3) EEM+FMM: selects sub-goals with our EEM but with no memory structure. The quantitative results are shown in Table II.

TABLE II

THE RESULTS OF ABLATION EXPERIMENTS ON THE TILE-MAP DATASET

Method	SPL↑	SR↑	DTS↓
Random	0.103	0.011	162.041
Random+FMM	0.323	0.064	119.976
EEM+FMM	0.608	0.370	98.533
MEEM	**0.768**	**0.401**	**55.274**

The comparison of Random with Random+FMM proves that our hierarchical policy can be used with FMM in completely unfamiliar environments and improve the agent's performance of the agent in visual navigation tasks. The better performance of EEM+FMM than random+FMM infers that selecting sub-goals based on the exploration-value is feasible and can help the agent perform better sub-goal planning. The best performance obtained by MEEM indicates that our episodic memory structure stores valid historical information, based on which our EEM can choose better sub-goals. In conclusion, all three components of our approach have positive effects on the agent's performance in visual navigation tasks.

C. Gibson in Habitat Simulator

To validate the feasibility of our MEEM for visual navigation in the real world, we train and evaluate MEEM on the Gibson dataset in the Habitat simulator, which consists of 3D scenes reconstructed from real-world environments. As with SemExp in [10], the train and val splits of the Gibson tiny set are used for training and evaluation respectively. The goal of this experiment is different from the previous ones, which is given by the semantics. It means that the agent needs to identify the semantic target from the environment and approaches it then.

The agent obtains a first-person RGBD picture of the environment through sensors at each timestep and uses semantic mapping techniques to construct a top-down semantic map based on the observed pictures. Actions are planned according to the 2D semantic map to reach the goal, where the distance to the semantic target is less than the success distance threshold d_s. We use the same semantic mapping techniques as SemExp to generate semantic maps, and the agent parameters are the same as SemExp. The action space includes four actions: $move_forward$, $turn_left$, $turn_right$, $stop$. Success is defined when the agent takes $stop$ action with the distance to the semantic goal less than d_s, which is $d_s = 1m$ in this experiment.

EEM adopts the same structure as in the tile-map experiment, except for the input, which contains five channels: (1) the obstacle map: mark all reachable positions, with 1 indicating reachable and 0 unreachable; (2) the observed area map: mark all positions observed by the agent, with 1 indicating observed and 0 unobserved; (3) the agent map: the map with all zeros except for the location of the agent; (4) the semantic map: contain only two types of semantic information, 1 for the goal locations and -1 for the non-goal locations; (5) the historical information map: generated by our episodic memory structure, mark all positions stored in our memory according to (6). The agent only gets a reward when success, which is 1, and DQN techniques with ϵ-greedy strategy are adopted to train EEM, with $\epsilon = 0.9$, $\beta = 0.9$, $\gamma = 0.95$. All other settings remain the same as SemExp, which include that the method of the lower layer is FMM, the maximum episode length is 500 steps, and the maximum number of actions of the lower layer for each sub-goal is 25. We train our MEEM for 10 million frames and use the Adam optimizer with a learning rate of 0.000025. The results is shown in Table III.

TABLE III
THE PERFORMANCE OF SEMEXP AND MEEM ON THE GIBSON DATASET

Method	SR↑	SPL↑	DTS↓
SemExp	0.544	0.199	1.723
MEEM	**0.698**	**0.328**	**1.576**

TABLE IV
THE PROBABILITIES OF CASES FOR MEEM EVALUATION EPISODES

Method	TP	TN	FP	FN
MEEM	0.685	0.093	0.209	0.013

Our MEEM outperforms SemExp in all three metrics, which indicates that our method can achieve better performance for visual navigation in the real world. Moreover, since our MEEM does not rely on any predefined information in the environment, theoretically, it can be applied to visual navigation tasks in various scenes, not only indoors.

To improve our approach in future work, we classified the evaluation episodes of MEEM into four categories: (1) True Positive (TP): the agent reaches the semantic target successfully; (2) True Negative (TN): the agent fails to reach the goal; (3) False Positive (FP): the agent misidentifies other objects as semantic targets and reaches it successfully; (4) False Negative (FN): the agent happens to stop in the success zone without correctly identifying the semantic target. Table IV shows the probabilities of these four cases. According to the definitions, TN and FP make up all failed episodes, and the statistics in Table IV show that FP accounts for more than two-thirds of them. We can reduce the probability of FP by improving the semantic recognition and segmentation models, which implies better semantic segmentation can further improve our approach.

V. CONCLUSION

In this paper, we propose an exploration-value-based model to improve the agent's performance for visual navigation, named MEEM. Our approach consists of three components: (1) a hierarchical policy that does not rely on any predefined information about the environment; (2) an episodic memory structure that stores the spatial and temporal information of historical sub-goals; (3) an Exploration-value Evaluation Model, which calculates exploration values for all positions in the observable area based on the top-down 2D maps. The experiment results show that MEEM achieves state-of-the-art performance both on the grid-world map dataset and 3D scenes in the Habitat simulator. The ablation studies based on the tile-map dataset demonstrate that all three components of our method have positive effects on performance. Moreover, the analysis of the results on the Gibson dataset indicates that the performance of MEEM can be improved with better semantic segmentation.

REFERENCES

[1] F. Zhu, Y. Zhu, V. Lee, X. Liang, and X. Chang, "Deep learning for embodied vision navigation: A survey," *arXiv preprint arXiv:2108.04097*, 2021.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[3] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*. MIT press Cambridge, MA, USA, 2017, vol. 1.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[5] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[6] E. Parisotto and R. Salakhutdinov, "Neural map: Structured memory for deep reinforcement learning," *arXiv preprint arXiv:1702.08360*, 2017.

[7] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3d environment," *arXiv preprint arXiv:1801.02209*, 2018.

[8] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6629–6638.

[9] A. Khan, C. Zhang, N. Atanasov, K. Karydis, V. Kumar, and D. D. Lee, "Memory augmented control networks," *arXiv preprint arXiv:1709.05706*, 2017.

[10] 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.

[11] X. Ye and Y. Yang, "Hierarchical and partially observable goal-driven policy learning with goals relational graph," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 101–14 110.

[12] J. Zhao, F. Huang, J. Lv, Y. Duan, Z. Qin, G. Li, and G. Tian, "Do rnn and lstm have long memory?" in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 365–11 375.

[13] I. Osband, B. Van Roy, and Z. Wen, "Generalization and exploration via randomized value functions," in *International Conference on Machine Learning*. PMLR, 2016, pp. 2377–2386.

[14] P. Foo, W. H. Warren, A. Duchon, and M. J. Tarr, "Do humans integrate routes into a cognitive map? map-versus landmark-based navigation of novel shortcuts." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 31, no. 2, p. 195, 2005.

[15] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[16] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[17] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *Advances in neural information processing systems*, vol. 29, pp. 3675–3683, 2016.

[18] 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.

[19] J. Oh, V. Chockalingam, H. Lee *et al.*, "Control of memory, active perception, and action in minecraft," in *International conference on machine learning*. PMLR, 2016, pp. 2790–2799.

[20] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," *arXiv preprint arXiv:1602.02867*, 2016.

[21] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[22] H. Wang, W. Wang, W. Liang, C. Xiong, and J. Shen, "Structured scene memory for vision-language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8455–8464.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[24] F. D. T. Markus 'Notch' Persson, "Tilemapgenerator public source repository," https://github.com/Dentrax/TileMapGenerator, 2018.

[25] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9068–9079.

[26] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik *et al.*, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9339–9347.

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[28] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[29] 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.