Hierarchical Reinforcement Learning-based Mapless Navigation with Predictive Exploration Worthiness

Yan Gao, Ze Ji
School of Engineering
Cardiff University
Cardiff, United Kingdom
gaoy84, jiz1@cardiff.ac.uk

Jing Wu
School of Computer Science and Informatics
Cardiff University
Cardiff, United Kingdom
wuj11@cardiff.ac.uk

Changyun Wei
College of Mechanical and Electrical Engineering
Hohai University
Changzhou, China
c.wei@hhu.edu.cn

Raphael Grech
Spirent Communications
Paignton, United Kingdom
raphael.grech@spirent.com

***Abstract*—Hierarchical reinforcement learning (HRL) is a promising approach for complex mapless navigation tasks by decomposing the task into a hierarchy of subtasks. However, selecting appropriate subgoals is challenging. Existing methods predominantly rely on sensory inputs, which may contain inadequate information or excessive redundancy. Inspired by the cognitive processes underpinning human navigation, our aim is to enable the robot to leverage both 'intrinsic and extrinsic factors' to make informed decisions regarding subgoal selection. In this work, we propose a novel HRL-based mapless navigation framework. Specifically, we introduce a predictive module, named Predictive Exploration Worthiness (PEW), into the high-level (HL) decision-making policy. The hypothesis is that the worthiness of an area for further exploration is related to obstacle spatial distribution, such as the area of free space and the distribution of obstacles. The PEW is introduced as a compact representation for obstacle spatial distribution. Additionally, to incorporate 'intrinsic factors' in the subgoal selection process, a penalty element is introduced in the HL reward function, allowing the robot to take into account the capabilities of the low-level policy when selecting subgoals. Our method exhibits significant improvements in success rate when tested in unseen environments.**

***Index Terms*—Mapless navigation, Deep Reinforcement Learning, Motion Planning, Hierarchical Reinforcement Learning**

I. Introduction

Autonomous navigation in unknown environments is a necessary capability for mobile robots. Conventional approaches use mapping techniques, such as Simultaneous Localization and Mapping (SLAM) [1], to pre-build maps and then apply path planning algorithms to guide the robot through the task. However, relying on pre-built maps is not always practical or reliable. Mapless navigation is considered as a remedy to relieve the navigation system from the prerequisite of a map. These situations include unstructured and dynamic environments where the layouts change constantly, such as domestic environments, disaster zones, and construction sites.

Recently, Reinforcement Learning (RL)-based approaches are receiving increasing attention in mapless navigation research [2], [3], where control policies are learned directly from raw sensory inputs without any prior information. RL-based mapless navigation offers an adaptable and flexible solution, enabling autonomous agents to navigate through a trial-and-error process using a reward-based system.

However, such methods struggle to perform well in long-distance navigation tasks in complex environments due to the long-term decision horizon and sparse rewards [2], [4]. A promising technique to overcome this limitation is Hierarchical Reinforcement Learning (HRL) [5], [6]. HRL contains two policies, high-level (HL) policy and low-level (LL) policy. The HL policy typically identifies a suitable sequence of subgoals that the LL policy can easily follow. This paradigm can effectively decompose a complex task into easier subtasks.

Using a hierarchical navigation approach results in more interpretable outcomes compared to non-hierarchical methods. During long-distance navigation tasks, humans often rely on intermediate landmarks, referred to as subgoals, to guide their movements [7]. These subgoals serve as intermediate steps that help individuals reach their final navigation objective. When navigating, humans take into account both intrinsic and extrinsic factors to determine their subgoals, where intrinsic factors could be related to the objectives, preferences, and expectations, and extrinsic factors pertain to the environment's layout, landmarks, available resources and so on [8], [9].

In simple terms, humans have the ability to distinguish the worthiness levels of different subgoals for further exploration, when selecting subgoals, and making appropriate decisions based on the observations or states [8]. For instance, humans can be aware that the space behind a couch is unoccupied, offering more path options. Similarly, when humans encounter a wall, they recognise that there is no viable path in that direction, and therefore, they will not select a subgoal in that direction. Conversely, when humans perceive a door, they understand that it leads to additional rooms, expanding the number of available path options. However, most HRL-based navigation methods lack such a similar capability. Numerous methods rely solely on current sensory input when selecting subgoals [6], [10]–[12]. Widely used sensors, such as Lidar, optical cameras, and RGBD sensors, provide high-dimensional raw measurement data that are inefficient for training RL agents for complex navigation tasks [2], [4]. Also, most methods cannot judge the worthiness for exploring each subgoal as with humans. In addition, many HRL approaches do not consider 'intrinsic factors', presupposing that the LL policy is feasible, indicating that the agent can consistently reach the

subgoal chosen by the HL policy [13], [14].

In this work, we propose a novel HRL-based mapless navigation framework with two layers, namely HL policy and LL policy. Fig. 1 depicts the framework of our work, where the HL policy determines the next subgoal and the LL policy decides the locomotion motion. We incorporate a predictive model called Predictive Exploration Worthiness (PEW) for the HL policy. The PEW is introduced to allow the robot to predict the worthiness level for exploration that is related to attributes of the area, such as the free space area, distribution of obstacles in the area, or the shape/orientation of the area. Based on RGB image and Lidar observations, the predicted PEW scores are used to evaluate the worthiness level of each subgoal candidate for further navigation and the values will be included as part of the HL input, such that the agent will not solely rely on sensory inputs for decision making. The LL policy will generate velocity commands based on the current Lidar observations to control the robot to reach the subgoal selected by the HL policy. After reaching the subgoal, the HL policy will repeat the process to select the next subgoal, until the robot reaches the final target location.

As mentioned, it is unrealistic to assume the optimality of the LL policy. Therefore, we introduce a penalty element to the HL reward function. This allows the HL policy to take into account the LL policy's capabilities when selecting subgoals and avoid the subgoals that the agent cannot reach under the LL policy's control.

II. Reinforcement Learning

A. Markov Decision Process

The objective of RL is to identify a policy for optimal decision-making by maximising the expected value of the cumulative sum of the received reward signal through continuous interactions with a given environment. The process we are interacting with is usually formalised as a Markov Decision Process (MDP). An MDP can be represented by a tuple $\langle S, A, R, p, \gamma, \rho_0 \rangle$. S is a set of states, A is the action space, $R(s, a)$ is a reward given the state s and the action a at each timestep, $p(s'|s, a)$ denotes the system transition function to state s', γ is the discount factor for future rewards and ρ_0 represents the initial state distribution. A policy $\pi(a|s)$ is a mapping of state s to action a. A state value function $V^\pi(s)$ is the expected value of the sum of rewards following policy π from state s, i.e., $V^\pi(s) = \mathrm{E}_{a\sim\pi, s\sim p}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$. A state-action value function $Q^\pi(s, a)$ defines the same quantity of taking action a in state s. The aim is to discover the optimal policy that can maximise the value function.

B. Deep Q-Network

In our method, we utilise DQN [15] to train the HL policy. Given the policy $\pi(a|s)$, the Q-value function can be defined. Then the function can be computed using the Bellman equation, formulated as:

$$Q^\pi(s_t, a_t) = \mathrm{E}[r_t + \gamma \mathrm{E}[Q^\pi(s_{t+1}, a_{t+1})|s_{t+1}, a_{t+1}, \pi]|s_t, a_t, \pi] \quad (1)$$

Using a deterministic greedy policy that selects the action with the highest Q-value at each state, the optimal Q-value function can be defined as:

$$Q^\pi(s_t, a_t) = \mathrm{E}[r_t + \gamma \max Q(s_{t+1}, a_{t+1})|s_t, a_t] \quad (2)$$

which implies that the optimal Q-value at time t is the sum of the current reward r_t and the discounted optimal Q-value available at time $t + 1$. This approach avoids the need to compute the Q-value function across a large state space. Subsequently, a deep neural network with parameter θ^Q is used to estimate the Q-values, expressed as $Q(s_t, a_t|\theta^Q)$.

C. Deep Deterministic Policy Gradients

The deep deterministic policy gradient (DDPG) algorithm [16] is used to train our LL policy. DDPG is an off-policy RL algorithm for continuous action spaces. It is an actor-critic method which contains two networks, an actor network $\pi(x_t|\theta^\pi)$ and a critic network $Q(s_t, a_t|\theta^Q)$. The actor network predicts the action a_t given the state s_t, with the goal to maximise the expected future rewards. The critic network estimates the Q-values. Instead of searching for the optimal Q-value over possible actions, the critic network's responsibility is to evaluate the current policy of the actor network. DDPG updates both networks at regular intervals.

III. HRL-based mapless navigation

In this work, we propose a novel HRL-based mapless navigation framework, as shown in Fig. 1. It contains three key modules, including the PEW model, HL policy and LL policy. Firstly, the PEW model predicts the PEW values for the positions of the subgoals. Then, the HL policy selects a subgoal based on current Lidar observations and the predicted PEW values. The LL policy is responsible for producing locomotion control of the agent to reach the subgoals selected by the HL policy.

A. Predictive Exploration Worthiness

To estimate the exploration worthiness of each subgoal, we propose a new metric, Predictive Exploration Worthiness (PEW). We consider the worthiness is related to the neighbouring space at each corresponding position. Specifically, the PEW model is designed to predict the free space area, as well as other attributes, such as the distribution, orientations and shape of the space. Fig. 2 illustrates one example of an occupancy view. It is obvious that the area can ensure navigation safety, and a larger area may provide more path choices. However, in more complex situations, we need to consider other attributes. For illustration purpose, Fig. 3 shows the occupancy views of three complex cases. In Fig. 3a and Fig. 3b, the areas of the free spaces around the two subgoals are similar, but the shape and distribution are different, and obviously, subgoal 2 is consider more preferable than subgoal 1 due to the complexity of the obstacles. On the other hand, we consider orientation would also be an important feature for navigation decision making. Fig. 3b and Fig. 3c have similar

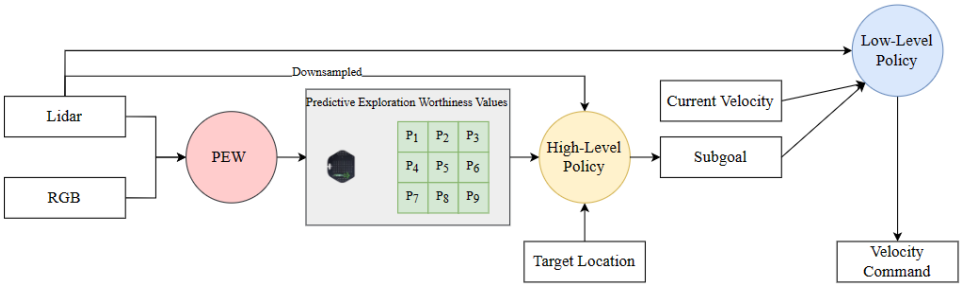


Fig. 1. The overall framework. The HL policy selects a subgoal based on the PEW values of each subgoal ($P_1, P_2, ..., P_9$), the Lidar observation and the relative goal position. The LL policy controls the robot to reach the subgoal. The process repeats until the robot reaches the target location.

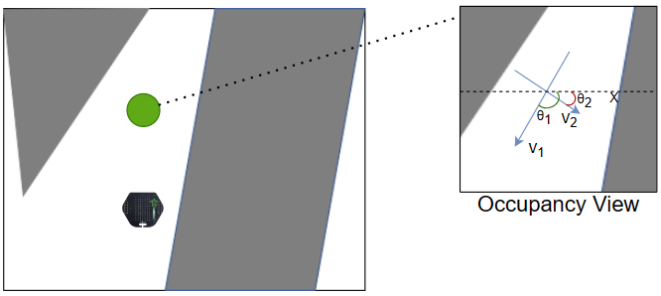


Fig. 2. An example of what the PEW model predicts. In the left figure, the grey areas represent occupied regions, and the white areas represent free space. The green circle denotes one of the subgoals the robot can select. The right figure is the occupancy view measurable with Lidar at the position of the subgoal. The PEW model is used to predict the area of the occupancy view and key features of the area, in terms of the distribution of obstacles and shape/orientation of the free space. To describe the features of the free space, we use the eigenvectors and eigenvalues of the pixels in the free space, where V1 and V2 represent the two eigenvectors with V1 having a larger eigenvalue.

Fig. 3. Occupancy views of some examples of complex subgoals. The white area is free space and the grey regions represent occupied or unknown space. All three figures contain free space of the same area, except their geometric distribution and orientations.

shapes and areas. However, if the target location is located on the right-hand side, subgoal 2 would be more preferred as this is more likely to lead to the goal location. Therefore, we believe the area, distribution, orientation and shape are the key features for the PEW metric, which is formulated as,

$$P(x, y) = [S, E] \tag{3}$$

where x and y are the coordinates of the subgoal with respect to the robot coordinate frame, S represents the area of the free space and E denotes the distribution and shape.

$$S = N_f / N_t \tag{4}$$

N_t represents the total area of the local region of interest (represented by 128×128 cells), and N_f represents the area of the free space (i.e. number of non-occupied cells) measurable by Lidar at $[x, y]$, $S \in [0, 1]$.

Rather than predicting the occupancy map directly, we introduce a compact representation of the free space, based on the Principal Component Analysis (PCA) [17]. PCA is a widely used method for dimension reduction, where the principal components refer to the eigenvectors of the covariance matrix. The specific method is as follows:

- **Matrix of points.** The local region is represented by 128×128 cells. Therefore, the region can be seen as a 128×128 matrix. The coordinates of the free cells are extracted to form a $2 \times n$ matrix, $\begin{bmatrix} x_1, x_2, ..., x_n \\ y_1, y_2, ..., y_n \end{bmatrix}$, where n is the number of free cells. The first row represents the x-coordinates, and the second is the y-coordinates.
- **Subtract the mean for each point.** The mean of the x coordinates is computed, and then, for each x-point, the mean value is subtracted from the x coordinates. This procedure is repeated for the y-coordinates.
- **Covariance matrix calculation.** Calculate the 2×2 covariance matrix.

$$C = \begin{bmatrix} variance(x, x) & variance(x, y) \\ variance(x, y) & variance(y, y) \end{bmatrix} \tag{5}$$

- **Eigenvectors, eigenvalues of covariance matrix.** Calculate the two eigenvalues and two eigenvectors of the covariance matrix.
- **Rearrange the eigen-pairs.** Sort by decreasing eigenvalues d_1, d_2. The dominant direction can be determined by the eigenvector having the largest eigenvalue, as shown in Fig. 2 where v_1 shows a larger eigenvalue.
- **Calculate the orientations of eigenvectors.** Calculate the angles between the two eigenvectors and the x-axis, denoted by θ_1 and θ_2 respectively, as illustrated in Fig. 2.

Therefore, E in Eq. 3 can be defined as $[d_1, d_2, \theta_1, \theta_2]$. Therefore, $P(x, y)$ in Eq. 3 includes 5 values in total.

The PEW model is trained in a supervised manner. We use the iGibson simulation environment [18] to obtain the ground-

truth scores as the training labels. For data collection, the robot is randomly placed at any arbitrary position of concern. Since the robot is equipped with a Lidar with an FoV of 360 degrees, the occupancy view can be obtained directly from the Lidar data. We then count the number of free cells N_f in the occupancy view and calculate the eigenvalues and orientations. If the neighbouring areas are fully occupied, N_f in Eq. 4, d_1, d_2, θ_1 and θ_2 are all set as 0.

We propose the network structure of the PEW model inspired by the occupancy anticipation model [19]. The network structure is shown in Fig. 4. Firstly, the RGB image is processed by ResNet18 to extract features, while Lidar observations are converted into the occupancy view provided by iGibson [18]. Then, both are encoded separately using Unet [20]. The RGB features are processed through a stack of three convolutional blocks, while the occupancy view is processed through a stack of five convolutional blocks. To create a combined feature, we merge these features using the Merge module, which comprises layer-specific convolution blocks to merge each layer of both encoded features. Finally, the combined feature is decoded using the Unet decoder that outputs the PEW values.

B. High-Level Policy

1) Subgoal Space: As mentioned, the HL policy is responsible for selecting a subgoal from the subgoal space. Humans can intuitively decide the next subgoal based on the forward view of observations. We mimic the behaviours by placing the robot's subgoal space in front of the robot, arranged in a 3×3 pattern, as shown in Fig. 5. The distance between each two neighbour subgoals is 0.5m. The PEW model then predicts the PEW value for each of the nine subgoals. Therefore, the final output P_{PEW} in Fig. 4 is a 3×15 matrix. In addition, the subgoal space also includes some rotational movements (14 angles in our work) for the agent to select when none of the nine subgoals in front of it is suitable. Therefore, there are in total 23 subgoals available for HL policy's selection.

2) State Representation: A state S_t^H at time t is defined as $S_t^H = [O_L||G_H||S_{PEW}]$, where $||$ denotes vector concatenation, O_L is the current Lidar observation, and G_H is the relative goal position. $S_{PEW} = Flatten(P_{PEW})$ is a 1×45 vector by flattening the P_{PEW} matrix.

3) Reward Function: The reward function of the HL policy is defined as

$$R^H = \begin{cases} r^H_{arrive} & \text{if } d_t \leq \delta^H \\ r^H_{collision} & \text{if collision} \\ r^H_{overtime} & \text{if } t_L \geq T \\ r^H_{approach} & \text{if approaching the subgoal} \\ r^H_{rotate} & \text{if rotate} \end{cases} \tag{6}$$

where

- r^H_{arrive} rewards the agent arriving at the target location, i.e., when the distance to the target location, d_t, is within a radius δ^H;
- $r^H_{collision}$ is a negative reward for penalising collisions;
- $r^H_{overtime}$ is a penalty for the case when the agent under LL policy's control is not able to reach the subgoal within a certain period of time T;
- $r^H_{approach} = d^H_{t-1} - d^H_t$ is the change of distance from the robot to the target location between two consecutive HL steps. $r^H_{approach}$ is the incentive for the robot to approach the target position; and
- $r^H_{rotate} = -c_r(|\frac{7\theta}{pi}|)$ is the penalty for the HL policy selecting a rotation subgoal. It increases with the rotation angle θ, and is scaled by the weighting factor c_r. The term r^H_{rotate} is incorporated to promote smoother robot motions.

4) Network Architecture: The Q-value of each subgoal is generated by a network comprising two MLP layers with sizes of 512 and 256, respectively. Only the output of the first layer is subjected to ReLU activation.

C. Low-Level Policy

The LL policy is responsible for controlling the robot to reach the subgoal, by directly interacting with the environment and generating the wheel rotational speeds for the agent.

1) State Representation: The LL state space is defined as $S_t^L = [O_L||G_{sub}||V_t]$, where $||$ represents vector concatenation. O_L is the current Lidar observation, and G_{sub} is the subgoal location represented in the polar coordinates of the robot frame. In addition, the LL policy should consider the current robot's velocity V_t when generating the command.

2) Reward Function: The LL policy reward function is

$$R^L = \begin{cases} r^L_{arrive} & \text{if } d_t \leq \delta^L \\ r^L_{collision} & \text{if collision} \\ r^L_{approach} & \text{otherwise} \end{cases} \tag{7}$$

where

- r^L_{arrive} is a positive value assigned to the agent when it successfully reaches the target location, indicated by its distance to the target, d_t, being within a certain radius δ^L;
- $r^L_{collision}$ penalises collisions;
- $r^L_{approach} = c_d(d^L_{t-1} - d^L_t)$ is the difference of the distances to the subgoal at timesteps of t and $t-1$, multiplied by a weighting factor c_d.

3) Network Architecture: The DDPG actor network consists of three MLP layers, all with a size of 512. The critic network also has three MLP layers, where the first and last layers have a dimension of 512 and the size of the second layer is 514, with two additional dimensions allocated for the action produced by the actor network. Both the actor and critic networks utilize ReLU activation for all layers, except for the output layers. The actor network employs hyperbolic tangent activation for the last layer, whereas the critic network has no activation for the output layer.

IV. Experiments

A. Implementation Details

We train the three key modules of our framework, namely PEW model, LL policy and HL policy, sequentially in the

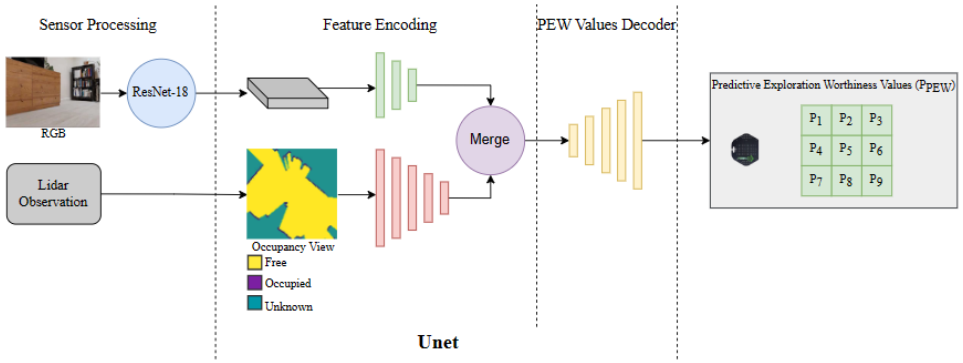


Fig. 4. Network structure of the PEW model.

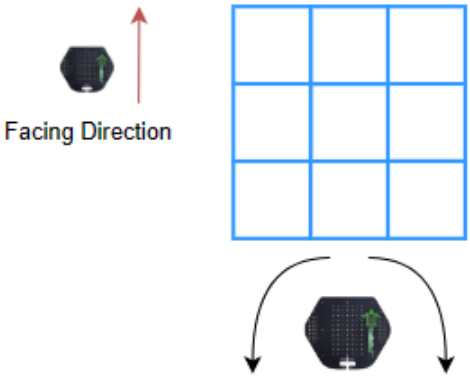


Fig. 5. Subgoal space for HL policy

iGibson environment [18], which contains numerous indoor virtual environments. A total of $40,000$ sets of training data are collected in 8 environments to train our PEW model. 10 additional environments are then selected to train the LL policy. We train the LL policy for $20,000$ steps in one environment and then change to other environments in sequence until the total steps reach 1 million. In each episode, the target location is chosen randomly, with a minimum distance of 0.5 meters from the robot, and within a 4-meter square centred around the robot. In Eq. 7, we set $r^L_{arrive} = 20$, $r^L_{collision} = -3$, $c_d = 10$. After finishing training the LL policy, we train our HL policy using the same 10 environments. The policy is trained with 150 episodes in each of the environments until the total number of episodes reaches $60,000$. In each episode, the initial location and the target location are randomly generated, with the travel distance ranges from 2 to 10m. We set $r^H_{arrive} = 20$, $r^H_{collision} = r^H_{overtime} = -3$, $c_r = 0.05$ in Eq. 6. These parameters are empirically set.

B. Experiment Settings

We select three previously unseen iGibson environments for testing, as shown in Fig. 6. To verify that our approach is capable of complex navigation tasks, we divide the tests in each environment into three levels. The difficulty level is determined by the distance from the robot's initial location to the target location. Specifically, we choose $[2-5]$m, $[5-8]$m and $[8-10]$m in our work for testing. To ensure fair comparisons, the same start and target locations are utilised for different methods with each test, consisting of 500 episodes for computing the average success rate.

C. Baselines

We select two RL-based mapless navigation methods as the baselines for benchmarking:

- Continuous space-based method [2]: The input includes the current Lidar observations, the polar coordinates of the target, and the velocity commands at the previous timestep. The output is the rotational speed of the wheels. The same network architecture is used for this baseline and also deployed by our LL policy. We use DDPG to train their navigation model, with the same reward function proposed in [2].
- Discrete space-based method [3]: The state representation includes the Lidar observations and the relative target position. The action space in their work includes 5 rotation velocities with a constant linear velocity. The actions are not directly provided by iGibson. Therefore, we modified the discrete action space to contain 5 similar actions, including moving forward and rotating by certain angles in place. The model is trained by Double DQN [21] utilising the reward function proposed in [3].

D. Results and Discussions

All methods were tested in the same three environments (Fig. 6) of varying difficulty levels. The success rates are presented in Table I.

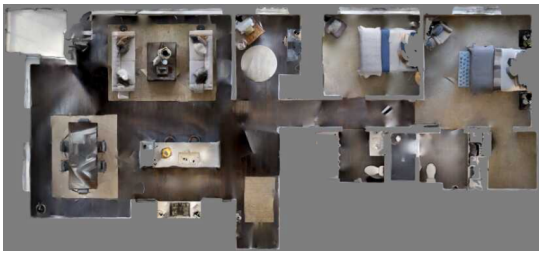
It is obvious that our method performs better than both approaches in all three environments, except for the test conducted in Environment 3 with the range of $2-5$m, which shows a small difference of only 1.6%. Our method exhibits significantly higher success rates than the other two methods in other tests, particularly as the tasks become more challenging. In the tests with the target range of $8-10$m, the continuous space-based baseline [2] and the discrete space-based baseline [3] achieve average success rates of 43.9% and 16.8%, respectively, while our method demonstrates a much higher success rate of 52.5%. Especially, the success

TABLE I
PERFORMANCE COMPARISON WITH TWO RL-BASED METHODS IN THE CONTINUOUS AND DISCRETE SPACE RESPECTIVELY [2], [3]

Env	Target range	Continuous space [2]	Discrete space [3]	Ours
1	2-5m	55.0%	38.4%	**57.4%**
	5-8m	50.4%	23.4%	**54.6%**
	8-10m	28.6%	10.2%	**40.6%**
2	2-5m	68.0%	45.0%	**70.2%**
	5-8m	57.6%	21.6%	**59.4%**
	8-10m	42.0%	14.0%	**50.2%**
3	2-5m	**74.8%**	65.0%	73.6%
	5-8m	65.0%	29.8%	**67.8%**
	8-10m	61.0%	26.2%	**66.6%**

(a) Env 1 (Allensville)

(b) Env 2 (Bolton)



(c) Env 3 (Chireno)

Fig. 6. Experiment environments for testing.

rate of our method is almost 40% higher than that of the discrete space-based method [3], suggesting that our approach outperforms in handling complex scenarios.

Fig. 7 illustrates the performance of the three methods in a long-range navigation task. The discrete action space-based method fails to move towards the target and collides with the obstacle. While the continuous action space-based method is able to approach the target location without any collision, it is trapped by a long table, which is referred as the local minimum problem. In contrast, our approach effectively addresses this issue. We speculate that the PEW assists the robot in selecting a more appropriate subgoal, enabling the robot to explore additional locations and successfully escape the situation, thereby highlighting our method's superior performance.

E. Ablation Studies

We conduct ablation studies to show the effect of our proposed PEW model. Specifically, two HL state configurations are compared:

- Lidar + Target: We remove the PEW module from our framework. The HL policy selects a subgoal solely based on the Lidar observations and the polar coordinates of the target location with respect to the robot frame.
- Lidar + Target + RGB features: With consideration of the RGB data within our framework, we incorporate the image information into the HL state representation to ensure a fair comparison. Specifically, the RGB features are extracted via ResNet18 and are not subject to additional processing to derive PEW values. Rather, the RGB features are directly integrated into the HL input representation.

These two methods differ from our method only in the HL state representation. The training method, the reward function, and the LL policy are identical.

We record the average reward per 1000 episodes during training for the two methods and ours, as shown in Fig. 8. Our proposed HL state representation achieves notably higher rewards during the latter stages of training in comparison to the other two input modalities. This suggests that our method is able to learn a better policy, given an equivalent number of training episodes.

Table II shows the test success rates. The results evidence that our proposed method, which incorporates the PEW value for each subgoal, outperforms the other two HL input modalities in all cases. The fusion of RGB features and Lidar

(a) Continuous space-based method

(b) Discrete space-based method

(c) Ours

Fig. 7. Examples of long-range navigation tasks. Orange and blue circles represent the start position and the target position respectively. (a), (b): Green lines represent the robot's trajectory. (c): Green circles are the subgoals selected by our HL policy.

data yields the poorest performance overall, whereas 'Lidar + Target' performs moderately in comparison. It suggests

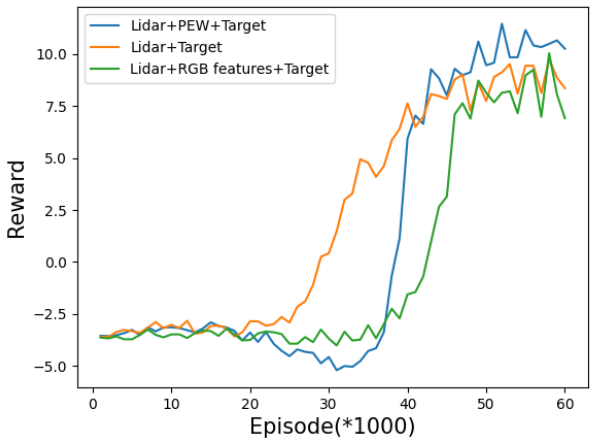


Fig. 8. Average rewards achieved by the agent with different state representations

that our state representation incorporates more environmental information than only utilising Lidar observations. While our method also avoids the inclusion of redundant data that may arise from the direct incorporation of RGB features. The PEW model results in a more compact HL input representation, ultimately increasing task success rates.

TABLE II
TEST SUCCESS RATES WITH DIFFERENT STATE REPRESENTATIONS

Env	Target range	Lidar + Target	Lidar + Target + RGB	Ours
1	2-5m	55.8%	50.4%	**57.4%**
	5-8m	49.8%	48.2%	**54.0%**
	8-10m	29.6%	35.8%	**40.6%**
2	2-5m	59.6%	61.2%	**70.2%**
	5-8m	39.6%	50.8%	**59.4%**
	8-10m	25.4%	37.6%	**50.2%**
3	2-5m	70.4%	72.0%	**73.6%**
	5-8m	61.6%	63.0%	**67.8%**
	8-10m	57.2%	62.0%	**66.6%**

V. CONCLUSIONS

In this paper, we present a novel mapless navigation approach using hierarchical reinforcement learning. Our proposed method addresses the challenge of effective subgoal selection for the HL policy. We introduce a predictive model, namely Predictive Exploration Worthiness (PEW), to enable the HL policy to rely on more effective sensory representations when selecting subgoals. PEW enriches the robot's perception of the environment by providing additional information which the raw sensors cannot provide directly. Also, it maintains the non-redundant nature of state representation. PEW achieves a balance between input information richness and redundancy. In addition, we have introduced a reward metric, incorporating 'intrinsic factors' in the subgoal selection process. Our experimental results demonstrate significant improvements in success rates across various complex environments. Especially,

the benefits of our proposed approach are particularly apparent in long-range tasks. Furthermore, ablation studies show that using Lidar and PEW values improves the performance consistently compared with using Lidar or combined with encoded RGB features.

Our limitation is that we train the two policies separately, resulting in a long training time. For future research, we will explore the mechanism that facilitates efficient parallel training of both policies to address the issues of instability and long training time. Subgoal space layouts hold significant importance in the HRL problem, and will continue to be a subject of our future exploration. Ultimately, our objective is to implement our approach on actual robots, which entails various challenges such as ensuring safety and bridging the gap between simulation and real-world experiments.

ACKNOWLEDGMENT

Yan Gao thanks the Chinese Scholarship Council (CSC) for providing the living stipend for his Ph.D. programme (No. 202008230171). This work was partially supported by the Royal Academy of Engineering under the Industrial Fellowships programme for Ze Ji (No. IF2223-199), hosted by Spirent Communications.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[2] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 31–36.

[3] E. Marchesini and A. Farinelli, "Discrete deep reinforcement learning for mapless navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 10 688–10 694.

[4] Y. Zhu, R. Mottaghi, E. Kolve, *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 3357–3364.

[5] A. C. Li, C. Florensa, I. Clavera, and P. Abbeel, "Sub-policy adaptation for hierarchical reinforcement learning," *arXiv preprint arXiv:1906.05862*, 2019.

[6] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.

[7] R. A. Epstein, E. Z. Patai, J. B. Julian, and H. J. Spiers, "The cognitive map in humans: Spatial navigation and beyond," *Nature neuroscience*, vol. 20, no. 11, pp. 1504–1513, 2017.

[8] T. Wolbers and J. M. Wiener, "Challenges for identifying the neural mechanisms that support spatial navigation: The impact of spatial scale," *Frontiers in human neuroscience*, vol. 8, p. 571, 2014.

[9] A. D. Ekstrom, A. E. Arnold, and G. Iaria, "A critical review of the allocentric spatial representation and its neural underpinnings: Toward a network-based perspective," *Frontiers in human neuroscience*, vol. 8, p. 803, 2014.

[10] J. Wöhlke, F. Schmitt, and H. van Hoof, "Hierarchies of planning and reinforcement learning for robot navigation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 10 682–10 688.

[11] B. Bischoff, D. Nguyen-Tuong, I Lee, F. Streichert, A. Knoll, *et al.*, "Hierarchical reinforcement learning for robot navigation," in *Proceedings of The European Symposium on Artificial Neural Networks, Computational Intelligence And Machine Learning (ESANN 2013)*, 2013.

[12] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," *arXiv preprint arXiv:1712.00948*, 2017.

[13] M. Eppe, P. D. Nguyen, and S. Wermter, "From semantics to execution: Integrating action planning with reinforcement learning for robotic causal problem-solving," *Frontiers in Robotics and AI*, vol. 6, p. 123, 2019.

[14] K. Yamamoto, T. Onishi, and Y. Tsuruoka, "Hierarchical reinforcement learning with abductive planning," *arXiv preprint arXiv:1806.10792*, 2018.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[17] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani, and A. Hooman, "An overview of principal component analysis," *Journal of Signal and Information Processing*, vol. 4, no. 3B, p. 173, 2013.

[18] B. Shen, F. Xia, C. Li, *et al.*, "Igibson 1.0: A simulation environment for interactive tasks in large realistic scenes," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 7520–7527.

[19] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman, "Occupancy anticipation for efficient exploration and navigation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, Springer, 2020, pp. 400–418.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, Springer, 2015, pp. 234–241.

[21] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.