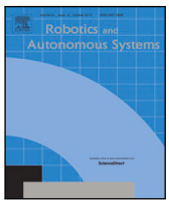
Robotics and Autonomous Systems 182 (2024) 104815

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Mapless navigation via Hierarchical Reinforcement Learning with memory-decaying novelty

Yan Gao [a], Feiqiang Lin [a], Boliang Cai [a], Jing Wu [b], Changyun Wei [c], Raphael Grech [d], Ze Ji [a,*]

[a] School of Engineering, Cardiff University, Cardiff, CF24 3AA, UK
[b] School of Computer Science and Informatics, Cardiff University, Cardiff, CF24 4AG, UK
[c] College of Mechanical and Electrical Engineering, Hohai University, Changzhou, China
[d] Spirent Communications, Paignton, TQ4 7QR, UK

ARTICLE INFO

Keywords:
Mapless navigation
Deep reinforcement learning
Collision avoidance
Hierarchical Reinforcement Learning
Path planning

ABSTRACT

Hierarchical Reinforcement Learning (HRL) has shown superior performance for mapless navigation tasks. However, it remains limited in unstructured environments that might contain terrains like long corridors and dead corners, which can lead to local minima. This is because most HRL-based mapless navigation methods employ a simplified reward setting and exploration strategy. In this work, we propose a novel reward function for training the high-level (HL) policy, which contains two components: extrinsic reward and intrinsic reward. The extrinsic reward encourages the robot to move towards the target location, while the intrinsic reward is computed based on novelty, episode memory and memory decaying, making the agent capable of accomplishing spontaneous exploration. We also design a novel neural network structure that incorporates an LSTM network to augment the agent with memory and reasoning capabilities. We test our method in unknown environments and specific scenarios prone to the local minimum problem to evaluate the navigation performance and local minimum resolution ability. The results show that our method significantly increases the success rate when compared to advanced RL-based methods, achieving a maximum improvement of nearly 28%. Our method demonstrates effective improvement in addressing the local minimum issue, especially in cases where the baselines fail completely. Additionally, numerous ablation studies consistently confirm the effectiveness of our proposed reward function and neural network structure.

1. Introduction

Mapless navigation refers to the task of finding a collision-free path to a specified goal relative to the agent, in situations where the mobile robot receives only local environmental information, without pre-constructed descriptions of the environment or online constructed maps. The mapless navigation ability is crucial for various applications in unstructured environments. These include indoor and outdoor scenarios, such as service robots for domestic and public environments, logistics in industrial warehouses, and urban search and rescue missions, where obtaining detailed and accurate maps in advance is challenging.

In recent years, deep reinforcement learning (DRL) has proven to be suitable for various complex tasks [1], including robot mapless navigation [2–5]. DRL-based mapless navigation works on a simple principle that an agent receives a positive reward for reaching the target location and a penalty for colliding with an obstacle [2,4,5]. Also, performance is usually evaluated based on the success rate of reaching the target location without collisions [4,6]. These works have good performance in simple environments [2,5], but it is difficult to obtain satisfactory results in complex ones, because of the long decision horizon and the sparse rewards. As a result, the agent tends to be stuck in a local region when it encounters complex and unseen environments, i.e., the local minimum problem.

Hierarchical Reinforcement Learning (HRL) is considered a promising learning framework for tackling the local minimum problem [7–9]. This is attributed to the nature of HRL that enables an agent to decompose a long-horizon task into a series of subtasks, which, in the case of navigation tasks, are intermediate destinations (subgoals). Such subgoals are easier to reach than a distant goal. Navigation planning can be divided into two levels, with a high-level (HL) policy selecting a subgoal for short-term navigation and a low-level (LL) policy controlling the robot's locomotion, which allows the robot to reach the subgoal [7–10].

Despite the effectiveness of HRL for tackling the local minimum problem [7–9], it remains limited for complex cluttered environments.

* Corresponding author.
E-mail address: jiz1@cardiff.ac.uk (Z. Ji).

https://doi.org/10.1016/j.robot.2024.104815
Received 20 February 2024; Received in revised form 11 July 2024; Accepted 12 September 2024
Available online 20 September 2024

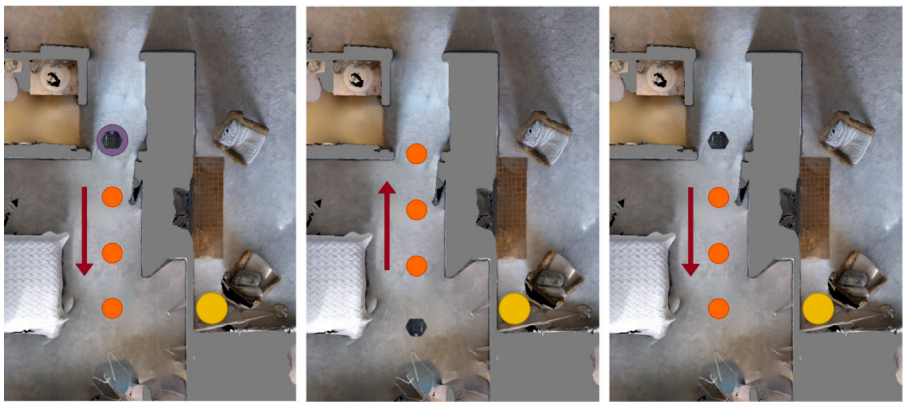


Fig. 1. This is an example of a local minimum problem, where the purple and yellow circles denote the starting and target locations, respectively. The target is situated behind a long wall. (a) Initially, the HL policy selects subsequent subgoals (denoted by the orange circles) in a downward direction, leading the robot towards the target based on the simple Euclidean distance. (b) However, since there is no direct path to the goal due to the obstruction posed by the wall, the robot would need to find alternative routes to bypass the wall. (c) The robot will continue to be attracted by the goal while exploring areas further away from the target, potentially getting trapped in the local area. To address this, we consider a memory mechanism or effective exploration motivation would enable the agent to avoid re-entering previously visited states, thereby mitigating the issue of local minima. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

As one example, Fig. 1 illustrates one case, where the target is located behind a long wall. In this case, the HRL agent can easily be trapped in a local area.

This is due to the following reasons. First, most (H)RL-based mapless navigation methods rely on a simplified reward setting, e.g. rewarded when getting closer to the goal and penalised for any collisions [8–10]. The distance-based reward is usually simply calculated based on the Euclidean distance, which is unrealistic in cluttered environments. Second, exploration is usually based on a simple inefficient random strategy for exploring unknown areas, especially for complex cluttered environments [8,9]. Last, in the case of being trapped in local areas, it would be desirable for the agent to choose alternative paths. We hypothesise that incorporating a memory mechanism could effectively address and resolve such issues.

To address these issues, we propose a new reward function for the HL policy, containing two main components, namely extrinsic reward and intrinsic reward. The extrinsic reward motivates the agent to move closer towards the target location, expressed as the change in the distance from the robot to the target at two consecutive HL steps.

The intrinsic reward in our work is inspired by the novelty theory defined in [11], which posits that animals reward themselves for identifying something novel. Analogously, an intrinsic reward can be designed to encourage robots to explore unknown environments by quantifying novelty as an intrinsic mechanism that drives curiosity about the world [12–14]. Episode memory is also an important attribute for designing an intrinsic reward [15]. Therefore, in this work, our intrinsic reward function has the following features. First, when the agent reaches a subgoal, previously visited areas or states will be retrieved for calculating the reward. Specifically, we use a count-based method [13]. Second, memory decaying is introduced in our work [16] as the basis for assigning rewards, where the size of the reward depends on the steps required to move from the current state to the corresponding state in the memory.

Our main contributions are as follows:

- We propose a novel intrinsic reward function, which incorporates components of novelty, episode memory, and memory decaying, that encourage the agent to explore the environment effectively.
- A novel neural network (NN) structure combined with an LSTM network [17] is introduced, enabling the agent with the capability to memorise past states for reasoning about subgoal selection.
- Numerous comparisons and ablation studies are conducted to demonstrate the superior performance of our method, focused specifically on the local minimum issue.

The rest of this paper is organised as follows. Section 2 discusses related works. Section 3 briefly reviews important preliminaries. Section 4 introduces our proposed method in detail. Section 5 introduces our experiment setup, followed by experimental results and discussions in Section 6. Section 7 concludes the paper.

2. Related work

Conventional map-based navigation methods have some obvious limitations. They require significant computational resources to build and update maps of the environment, using techniques such as Simultaneous Localisation and Mapping (SLAM) [18]. The control performance is entirely dependent on a mathematical model of the robot that is often simplified or linearised. This can often lead to weakened robustness for the system [2,3]. Mapless navigation is widely regarded as an approach that relieves the navigation system from the prerequisites of a map [2,3,5]. In particular, the approach based on DRL can model a direct mapping between sensory inputs and robot actions and has gained increasing interest in recent years [2,3,19]. For example, Tai et al. [2] successfully apply DRL to mapless navigation, using sparse Lidar readings, robot velocity, and the relative position of the target location as inputs. The policy network can output control commands for nonholonomic robots. Lin et al. [20] enables the robot to learn a mapless navigation policy that prevents localisation failure by proposing a reward metric to penalise behaviours that lead to localisation failures and a reconfigured state representation, including the current observations and historical trajectory information to transfer the problem to a Markov Decision Processes (MDP) model.

However, most of the works have been trained and tested in simple environments [2,5,20]. Considering that most methods have long decision horizons and sparse rewards [5], they struggle to perform well in complex environments. Wijmans et al. [21] propose the DD-PPO algorithm to achieve great performance in mapless navigation tasks. DD-PPO represents a decentralised distributed framework designed for GPU clustering. Specifically, they utilise 64 Nvidia V100 GPUs to train their model over 2.5 billion steps. However, meeting these training requirements is challenging.

HRL is widely regarded as a promising way to learn goal-conditioned behaviours in long-horizon and complex tasks [22–25]. Such methods typically design HL policies that operate on a coarser time scale and control the execution of the LL policy. Typical HRL approaches, such as Option Critic [26], Feudal Networks (FuN) [27], and HiRO Networks [24] make use of these ideas of generating subgoals and

hierarchically integrating policies to combine them with NNs and RL algorithms. FuN and HiRO combine the HL policy, which generates subgoals at a lower frequency, and the LL policy, which reaches these subgoals and receives an intrinsic reward.

Numerous works have applied HRL to robotic navigation tasks with promising performance [7–10]. Wohlke et al. [7] use grids to represent a rough map of the environment, providing the agent with obstacle and reachability information. The HL policy will select a subgoal in a 3×3 subgoal space centred at the agent and pass relevant information to the LL policy. Bischof et al. [9] divide the entire path from the robot's current location to the target location into several segments, i.e., a list of discrete waypoints, which can be seen as subgoals of the HL policy.

However, most HRL methods do not consider the performance of the LL policy in the HL planning and always assume that the robot under the control of the LL policy can reach any subgoal selected by the HL policy [28,29]. In addition, some methods require prior knowledge, such as a coarse grid map of the environment [7,9]. Furthermore, despite the promising performance with HRL for mapless navigation tasks, the local minimum problem remains challenging in complex environments. This might be due to the simplified HL reward setting, e.g. the agent is rewarded based on a simple Euclidean distance-based metric, ignoring the complexity of cluttered environments [8–10]. Most HRL-based approaches use simple exploration strategies, such as ϵ-greedy [8,9], such that the agents lack effective exploration motivation. Also, most agents lack the memory mechanism, so they may revisit those already visited areas, including the paths that have proven infeasible for the navigation task and will be trapped in a local region.

With regard to exploration in unknown environments, many studies show that biological entities use novelty as the driving force for exploration [30]. Inspired by this phenomenon, many researchers have attempted to quantify novelty in artificial intelligent (AI) algorithms, formulating novelty as an intrinsic mechanism of AI agents that drives the curiosity for understanding the world [12–14]. For example, Tang et al. [14] combine hash tables and classical count-based exploration to compute novelty rewards. This combination enables the method to achieve state-of-the-art performance in various continuous DRL benchmark tests. Zhelo et al. [4] delve into the examination of exploration strategies employed by DRL to learn navigation policies. Specifically, they enhance the conventional extrinsic reward utilised for training DRL algorithms by incorporating intrinsic reward signals obtained through modelling curiosity. Currently, in RL, when the agent is anticipated to be subject to rewards emanating from multiple sources concurrently (such as intrinsic and extrinsic rewards), it is commonly observed that all rewards are aggregated through summation to construct a comprehensive reward function. This approach has been demonstrated to be effective [4,31].

Moreover, as discussed, it is hypothesised that a memory mechanism could enhance navigation performance. Hausknecht et al. [32] propose to replace the first post-convolutional fully-connected layer with a recurrent LSTM to enable the agent to memorise past states. A number of related works, including this paper, have adopted a similar architecture to address various problems. For example, Mnih et al. [1] train an RL-based model combined with an LSTM network to enable agents with memory units for navigating in 3D mazes. Singla et al. [33] propose a memory-based DRL method for avoiding collisions in indoor environments, with an LSTM network processing partial information acquired previously. Hu et al. [34] present a sim-to-real pipeline for DRL-based autonomous robot navigation in cluttered rough terrain, where an A3C-based policy is adopted [1]. In the work, LSTM is employed to capture information from previous states. Mirowski et al. [35] construct a stacked LSTM framework demonstrating enhanced data efficiency and task performance through the incorporation of supplementary navigation-related signals. Tang et al. [36] combine the cognitive mapping capability of the entorhinal cortex with the episodic memory function of the hippocampus. By recollecting past travel experiences stored in episodic memory, robots construct a map of the environment, facilitating more sophisticated cognitive navigation tasks. In our work, new NNs are proposed to equip the agent with memory and reasoning capabilities. To the best of the authors' knowledge, no research has been conducted considering these issues of the HRL-based navigation scheme.

In addition, most HRL-based mapless navigation approaches have primarily been evaluated in straightforward synthetic environments, such as ROS/Gazebo. The authors are not aware of any work on HRL-based mapless navigation in photo-realistic environments with intricate characteristics, like in iGibson [37] and Habitat [38].

3. Preliminaries

In this section, we will briefly review the important preliminaries on which our HL model and LL model are built, including Markov Decision Processes (MDPs), Reinforcement Learning-based Mapless Navigation, Goal-conditioned Reinforcement Learning (GRL), and two main Reinforcement Learning algorithms, namely Deep Q Network (DQN) and Deep Deterministic Policy Gradient (DDPG).

Markov Decision Process: An MDP is defined by a tuple $\langle S, A, R, p, \gamma, \rho_0 \rangle$, where S is the state space, A the action space, $R(s,a)$ the reward function, $p(s'|s,a)$ the system transition model, γ the discount factor, and ρ_0 the initial state distribution. A policy $\pi(a|s)$ is a mapping from states to actions. A state value function $V^\pi(s)$ is defined as the sum of the expected, discounted returns obtained by an agent following a policy π starting 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, $Q^\pi(s,a)$, is defined as the same quantity starting from taking an action a at state s and following a policy π thereafter. The goal of RL algorithms is to obtain an optimal policy that maximises the value functions.

Reinforcement Learning-based Mapless Navigation: Mapless navigation can be formulated as an MDP problem. At each timestep t, an action $a_t \in A$ is selected, given the current state $s_t \in S$, by policy $\pi(a_t|s_t)$, formulated as:

$$a_t \sim \pi(a_t|s_t) \tag{1}$$

Specifically in this work, the MDP can be described as

$$(v_{left}, v_{right}) \sim \pi(a_t|o_t, d, \theta) \tag{2}$$

where s_t is substituted by the Lidar observation o_t and goal location $g_t = (d, \theta)$ in the polar coordinates with respect to the robot frame. Action a_t, in this case, is the angular velocities of both wheels (v_{left}, v_{right}). It is worth noting that the policy π does not rely on the map for decision making, hence mapless navigation. Also, it can be seen that π is a function of not only the observation, but also the goal. This makes the problem a goal-conditioned reinforcement learning problem, as introduced next.

Goal-conditioned Reinforcement Learning: A GRL problem acts on an MDP with a goal space G. The standard RL problem pursues a single specific goal, while the agent in GRL seeks to maximise a universal value function conditioned by an arbitrary goal, defined as $V^\pi(s,g)$. Typically, a goal is defined as a transition to a state, i.e., $g = m(s)$. GRL would usually update its policy based on an achieved state of an episode, as the conditional goal g, for training a GRL policy that would consider multiple goals.

Deep Q Network (DQN): DQN is an off-policy deterministic RL algorithm that uses NNs that are parameterised with θ^Q to approximate the Q function of a discrete action task. Given a specific learning sample (s_t, a_t, r_t, s_{t+1}), the loss function for optimising the network parameters is formulated as $L(\theta^Q) = [y - Q(s_t, a_t)|\theta^Q]^2$, where the target $y = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{a+1}|\theta^Q)$. In practice, it is common to use target networks to stabilise the learning process. **Deep Deterministic Policy Gradient (DDPG)**: DDPG is an RL algorithm for continuous action spaces. It has an actor–critic style, using separate NNs to approximate the Q function and the deterministic policy. DDPG updates both networks at regular intervals. In practice, it is common to use a target network and a second critic network to mitigate the overestimation problem in DDPG and to improve learning efficiency.

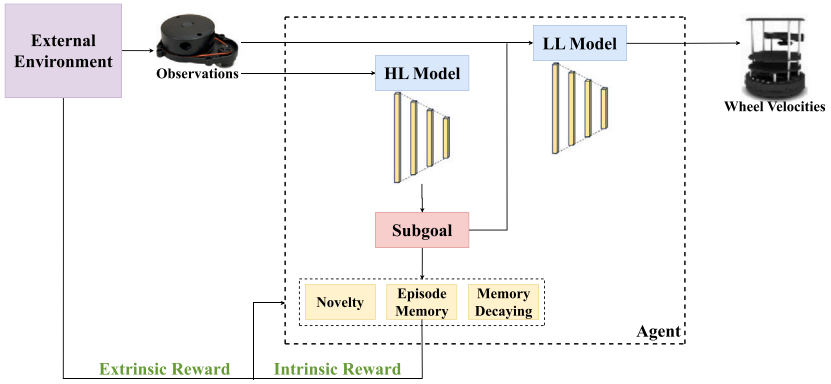


Fig. 2. The overall framework. The HL policy selects a subgoal based on the HL input. The LL policy controls the robot to reach the subgoal, given the subgoal and other LL input information. The process repeats until the robot reaches the target location. Regarding HL model training, we propose a new reward function that has two components: extrinsic and intrinsic rewards. The environment provides the extrinsic reward, and the intrinsic reward is calculated based on novelty, episode memory, and memory decaying.

4. Methods

We propose an HRL-based mapless navigation method, where the HL policy is responsible for selecting a subgoal in the subgoal space and the LL policy controls the locomotion of the robot to reach the corresponding subgoal. Fig. 2 illustrates the proposed framework. To train the HL policy, we introduce a new reward function that includes two components: extrinsic and intrinsic rewards. Briefly, the extrinsic reward originates from the environment and motivates the agent to move towards the target location. The intrinsic reward depends on three key variables, namely novelty, episode memory and memory decay. In this section, we will describe the working principles of the HL and LL policies and their training processes, respectively.

4.1. High-level policy

The following subsections describe the main components of the HL policy, namely the observation, the action/subgoal space, the reward function, and the network structure.

4.1.1. Observation

The observation of the HL policy comprised of 6 parts, denoted as $o_t^H = \{o_H \| g_{p\_t}^H \| a_{t-1}^H \| r_{t-1}^H \| x_t, y_t, \phi_t \| N(x_t, y_t)\}$, where $\|$ denotes vector concatenation. o_H is the current sensor reading, and $g_{p\_t}^H$ denotes the polar coordinates of the target location with respect to the robot frame. For the agent to learn policies through past experience, the action a_{t-1}^H executed at the last HL step $t-1$, i.e. the last selected subgoal, and the HL reward r_{t-1}^H at the previous step $t-1$, are also included. x_t, y_t, ϕ_t represent the coordinates of the current location and heading of the robot. The magnitude of the reward is contingent upon the frequency of the robot's visits to the current location, which will be described in Section 4.2.3. To expedite the agent's acquisition of the correlation between the reward and visit frequency, we add $N(x_t, y_t)$, which is the number of visits to the current location by the robot, to the observation.

4.1.2. Action space (subgoals)

The HL action space, denoted by A^H, is constructed by a list of subgoals that are needed by the LL policy for short-term navigation. Effectively, the HL action space can be seen as the LL goals, i.e., $A^H = G^L$.

Some works use 8 adjacent regions centred on the robot as the action space [7]. To provide the robot with more options in a complex and obstacle-laden environment, we add 2 additional regions in the front and rear regions of the robot, respectively, as shown in Fig. 3. In addition, since the HL input contains the past states, to initialise the agent when no historical experience is available, we use stand still as historical experience for the initial state. Therefore, our subgoal space consists of 13 subgoals, with 12 surrounding areas and one for keeping the robot standing still.

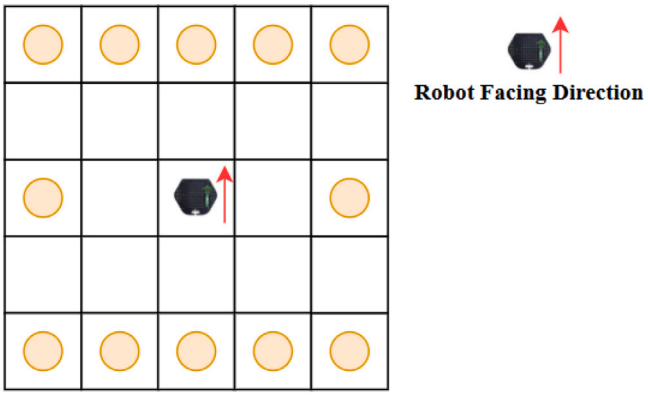


Fig. 3. Subgoal space. We set the surrounding area, centred on the robot's current pose, as the subgoal space (yellow circle). Each grid is 0.35 m in width and length. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.1.3. Reward function

As mentioned, our reward function contains two components: extrinsic and intrinsic rewards. The total HL reward for each step is the sum of the extrinsic and intrinsic rewards except when the robot reaches the target, or the agent remains standing still. The reward R_t^H is defined as

$$R_t^H = \begin{cases} r_{arrive}^H & \text{if } d_t \le \delta^H \\ R_{ex} + R_{in} & \text{other} \\ R_s & \text{stand still} \end{cases} \tag{3}$$

Three conditions are considered for defining R_t^H. R_s is a negative reward for penalising standing still. A positive value r_{arrive}^H is assigned to the reward, when the robot reaches the target location. This is based on a distance threshold condition when the distance to the target d_t is within a radius δ^H. R_{ex} and R_{in} are the extrinsic and intrinsic rewards, respectively, and their summation of them forms the reward in other situations. This approach has been demonstrated to be effective [4,31].

The extrinsic reward is defined in a similar manner as employed in other non-hierarchical RL-based mapless navigation methods [2,4]. Its mathematical expression is

$$R_{ex} = L_{ex_{t-1}} - L_{ex_t} \tag{4}$$

The extrinsic reward is defined as the change of distance from the robot to the target location between two consecutive HL steps. L_{ex_t} represents the distance from the target at HL step t. The extrinsic reward magnitude increases as the robot approaches the target with

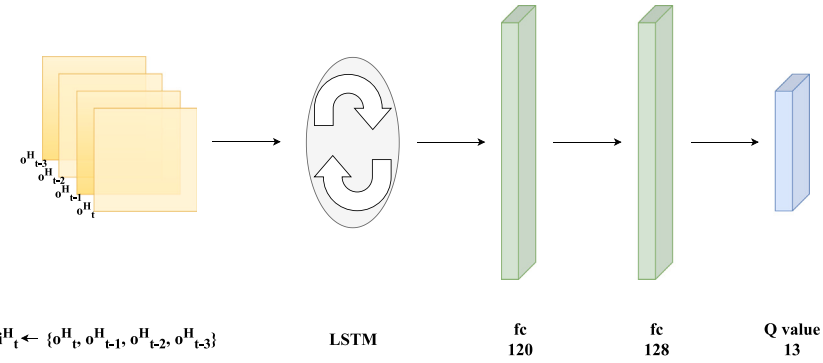


Fig. 4. Network structure of HL policy. The input i_t^H includes the agent's current observation and also the observations from the last three HL steps. The output is the Q value of each subgoal.

greater proximity. This is directly useful for the agent to be motivated by the extrinsic reward to learn to move towards the target location.

The intrinsic reward encourages the agent to explore unknown environments. Analogous to the behaviours of animals that rely on spatial memories to determine the novelty of a location [39], for navigation tasks, we consider spatial memory an important mechanism for implementing robot intrinsic motivation for exploration. Specifically, in the work, we store the coordinates of all subgoals the HL policy has previously selected, formulated as below:

$$M = [x_0, y_0; x_1, y_1; ...; x_{t-1}, y_{t-1}] \tag{5}$$

The novelty of the current state is derived from the number of occurrences in which the agent has visited the corresponding location. We name this the Novelty Count (NC), denoted by $N(x, y)$.

A threshold δ^B is used as the condition to determine if the location has been visited. The number of visits at the current place $N(x_t, y_t)$ will be increased by 1 if the current location is within δ^B to the corresponding location.

A high NC indicates low novelty for the current state of the agent. The intrinsic reward function can be formulated as

$$R_{in} = \alpha N(x_t, y_t) \tag{6}$$

where $N(x_t, y_t)$ is the NC and α is a weighting factor that is negative.

The intrinsic reward encourages the robot to explore more areas efficiently by avoiding reaching locations that have been previously visited. However, this is not ideal in all situations. One example is when the robot arrives at a dead end, where alternative paths are limited or do not exist. In this case, the above intrinsic reward would prohibit the robot from leaving the local region through the only path that has been traversed. Although previously visited places are less favourable, we consider them still worth further exploration. In our work, we introduce a memory decaying factor M_d to the intrinsic reward. Mathematically, we define M_d as a function of the time that the agent has spent travelling to the current location (x_t, y_t) from the corresponding location in the episode memory. M_d is formulated as follows:

$$M_d = e^{-(L_M - 1)/10} \tag{7}$$

where L_M represent the number of HL steps since the last time the robot visited the location. M_d is zero if the location has not been previously visited. A high L_M indicates that the agent visited the corresponding place long ago, leading to a small value of M_d. Conversely, if the agent recently visited a location, the value of L_M will be low. L_M is 1 if the agent visited the place in the previous HL step. Since α is negative, the agent will incur a significant penalty for revisiting recently visited locations. Integrating the memory decaying with the intrinsic reward function gives Eq. (6), as follows:

$$R_{in} = \alpha N(x_t, y_t) M_d \tag{8}$$

In order to avoid an infinitely small reward, a minimum value R_0 is set, and $R_{ex} + R_{in}$ thus becomes:

$$R_{ex+in} = \max(R_0, R_{ex} + R_{in}) \tag{9}$$

In addition, to enable the HL policy to take the LL policy's ability into account when selecting subgoals, we add some penalty elements to the reward function. The complete reward function R_t^H for the HL policy becomes:

$$R_t^H = \begin{cases} r^H_{arrive} & \text{if } d_t \le \delta^H \\ r^H_{collision} & \text{if collision} \\ r^H_{overtime} & \text{if } t_L \ge T \\ R_{ex+in} & \text{other} \\ R_s & \text{stand still} \end{cases} \tag{10}$$

where $r^H_{collision}$ and $r^H_{overtime}$ are the penalty values that occur when the LL policy fails to reach a selected subgoal due to collision or timeout.

4.1.4. Network structure

We use DQN to train our HL policy. As shown in Fig. 4, we employ the LSTM network to equip the agent with memory and reasoning capabilities. By leveraging memory units and gating mechanisms, the LSTM module enables the capture and utilisation of past information, thereby assisting the agent in making more informed policies. Specifically, we input a sequence of the four recent states $i_t^H = \{o^H_{t-3}, o^H_{t-2}, o^H_{t-1}, o^H_t\}$ into a single layer of 30 LSTM cells, facilitating the extraction and integration of relevant temporal patterns. These are followed by two fully connected layers with 120 and 128 units. The network output includes the Q values for selecting the corresponding subgoals with a given input. Alg. 1 details the algorithmic implementation of our HL policy.

4.2. Low-level policy

The LL policy is responsible for outputting the robot's control commands in order to navigate to any given short-term goal. This subsection will introduce the LL policy in terms of observation, action, reward, and network structure.

4.2.1. Observation

The observation of the LL policy comprises five parts, formulated as $o_t^L = \{o_L \| v^L \| g^L_{p\_t} \| a^L_{t-1} \| r^L_{t-1}\}$, where $\|$ denotes vector concatenation, o_L represents the Lidar readings at the current pose, $g^L_{p\_t}$ represents the target location in the polar coordinates of the robot frame, a^L_{t-1} is the action produced by the LL policy at the last timestep and r^L_{t-1} is the reward by executing action a^L_{t-1}.

4.2.2. Action

We use a TurtleBot, which has a differential drive configuration. Therefore, the LL policy outputs the velocities of the two wheels, i.e., $a_t^L = \{v_{left}, v_{right}\}$.

4.2.3. Reward function

The reward function for the LL policy is shown below:

$$R^L(o_t^L, a_t^L, g^L) = \begin{cases} r^L_{arrive} & \text{if } d_t \le \delta^L \\ r^L_{collision} & \text{if collision} \\ r^L_{approach} & \text{otherwise} \end{cases} \tag{11}$$

where

- r^L_{arrive} is a large positive value that will be given when the robot reaches the target location, i.e., when its distance to the target d_t is within a radius δ^L;
- $r^L_{collision}$ penalises the policy when the robot collides with an obstacle; and
- $r^L_{approach} = c_d(d_{t-1} - d_t)$ is the approaching reward, where $(d_{t-1} - d_t)$ is the distance difference to the target between the current and the last timesteps, and c_d is a weighting factor.

4.2.4. Network structure

DDPG is used for the LL policy. The actor network for the DDPG-based agent has a single layer of 64 LSTM cells first, followed by three MLP layers, with the same size of 512. The critic network also contains a single layer comprising 64 LSTM cells and three MLP layers, where the sizes of the first and last MLP layers are both 512, and the second layer has two extra dimensions for the actions, making the size 514. Both the actor and critic networks use ReLU activation on each MLP layer, except for the output. The actor network uses hyperbolic tangent to activate the final layer, and the critic network has no activation on the output.

Algorithm 1: HL policy of our HRL model

Given:
- Pretrained LL policy π_{LL};
- HL policy π_{HL}, HL buffer D_{HL};

Initialise DQN of HL policy;
for $m = 0$ **to** M *epoch* **do**
  # Sample training environment Env 1, 2, 3...(once m is changed, sample another different environment in turn)
  for $j = 0$ **to** J *training steps* **do**
    $t_{HL} = 0$,
    $done = 0$
    Sample a target location g_t
    while do
      # Obtain a subgoal g_s
      $g_s \sim$ epsilon-greedy$(\pi_{HL}(i_t^H))$
      while do
        $t_{LL} = 0$
        $a_{t_{LL}} \sim \pi_{LL}(o_t^L)$
        $t_{LL} = t_{LL} + 1$
        $o_t^L = o_{t+1}^L$
        if $t_{LL} \ge T_{LL}$, *or* g_s *reached, or collision* **then**
          break
      $t_{HL} = t_{HL} + 1$
      if $t_{LL} \ge T_{LL}$, *or* g_t *reached, or collision, or* $t_{HL} \ge T_{HL}$ **then**
        $done = 1$
      $D_{HL} \leftarrow (i_t^H, g_s, R^H, i_{t+1}^H, \text{done})$
      $\lambda_{t_{HL+1}} \leftarrow$ Adam $(\lambda_{t_{HL}}, D_{HL})$
      $i_t^H = i_{t+1}^H$
      if $done = 1$ **then**
        break

5. Experiments

In our work, we use the iGibson simulator [37], which is developed based on the Gibson dataset [40] that contains 572 realistic 3D indoor environments. The robot we use for training is a TurtleBot equipped with a 360 laser beam Lidar covering a field of view (FoV) of 360 degrees.

We use a total of 10 different environments to train the HL policy and the LL policy separately. As mentioned, one main cause for failure in mapless navigation tasks is the local minimum problem. To better illustrate the effectiveness of our approach in addressing this issue, three complex indoor environments which have never been seen before are chosen for testing, as shown in Fig. 5. The dimensions of the environments are as follows: Env 1 is 9.5 m × 8.5 m, Env 2 is 12.0 m × 8.0 m, and Env 3 is 10.0 m × 9.5 m.

The tests in each environment are divided into three difficulty levels, which are defined according to the distance between the robot's initial location and the target location, ranging from 2–5 m, 5–8 m, and 8–10 m. The greater the distance, the more difficult the task. Difficult tasks may contain complex scenarios and more likely result in local minima. We perform 500 episodes per test and record the average success rate, with the initial and target locations randomly generated by the environment. Every randomised initial state needs to satisfy the corresponding task difficulty. To ensure fair evaluation, the initial and target locations are the same across different methods when testing. In addition, to verify our method's superior performance in tackling the local minimum problem, we also choose 3 specific scenarios that are prone to this problem for demonstration, as shown in Fig. 6. In the three chosen scenarios, the length of the wall between the initial location and the target location is 5 m, 6 m and 3.5 m, respectively. Each scenario is tested for 100 episodes, and the success rates are recorded.

5.1. LL policy training

The LL and HL policies are trained separately. Considering that the reward function of the HL policy takes into account the performance of the LL policy, we train the LL policy first. In each episode, the initial location of the robot is randomised. Since the LL policy is responsible for short-range navigation, we randomly generate the target location within a square with a side of 2 m centred on the robot.

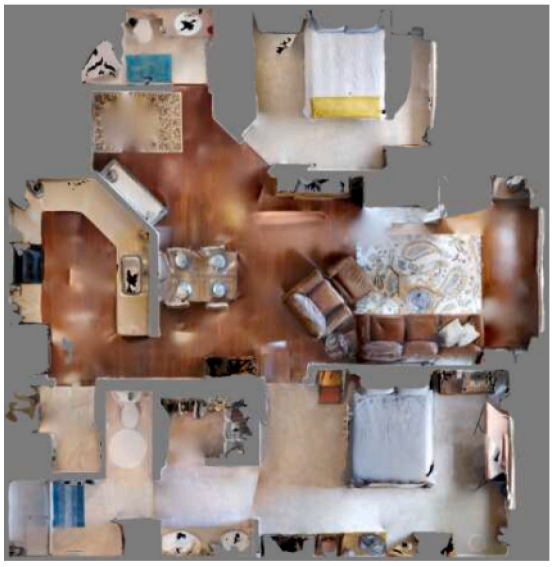
The parameters in Eq. (11) are set as below. The arrival reward $r^L_{arrive} = 20$ is given when the robot is no more than $\delta^L = 0.36$ m away from the target location (the chassis radius of Turtlebot is 0.36 m). The collision penalty is $r^L_{collision} = -3$. The hyperparameter c_d in $r^L_{approach}$ is set to 10 empirically. The full length of an episode is 1500 timesteps. We train the LL policy for 20000 timesteps in one environment and then continue with other environments until a total of 8 million timesteps is reached. Our LL policy requires the computational time of 0.002 s to compute the velocity action.

5.2. HL policy training

The HL policy training commences after the LL policy training is completed. In each episode, the robot is randomly placed at a location, and the target location is also randomly generated by the environment. However, the distance of each episode is set at 2–10 m. The conditions for terminating each episode are as follows. The robot reaches the target location successfully, i.e. the distance between the robot and the target location is less than 0.86 m. In this case, the arrival reward $r^H_{arrive} = 20$ is given. When the robot collides with an obstacle, the collision penalty is set as $r^H_{collision} = -3$. If the target location cannot be reached after 200 subgoals selected by the HL policy, this is considered too long and not worth further exploring. Also, if the LL policy cannot reach the subgoal selected by the HL policy within 800 timesteps, then the overtime penalty is $r^H_{overtime} = -3$. If the agent chooses to stand still, $R_s = -2.5$ will be given. In the intrinsic reward function, the weighting

(a) Env 1 (Allensville)

(b) Env 2 (Artois)

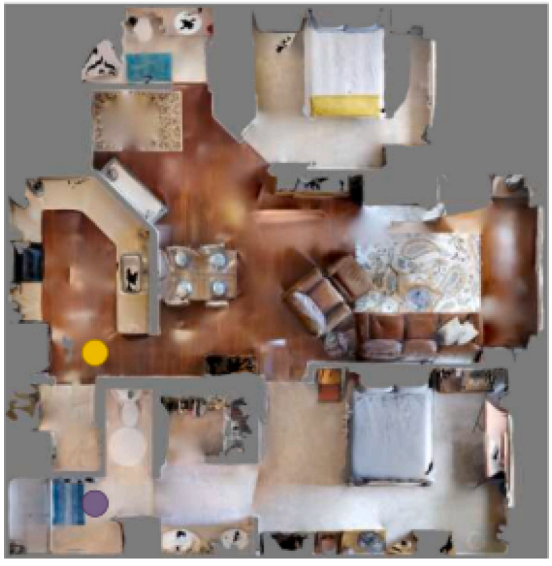


(c) Env 3 (Aulander)

Fig. 5. Experiment environments for testing.

(a) Scenario 1

(b) Scenario 2



(c) Scenario 3

Fig. 6. Specific scenarios for testing. Purple and yellow circles represent the start location and the target location, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

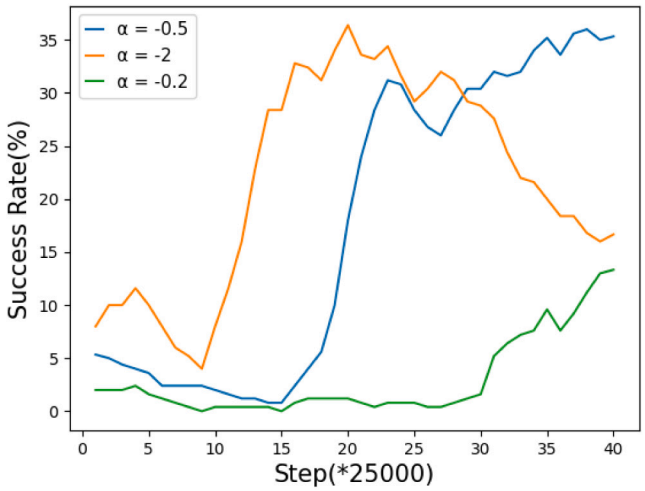


Fig. 7. Test success rates with different values of α in the intrinsic reward during the training process.

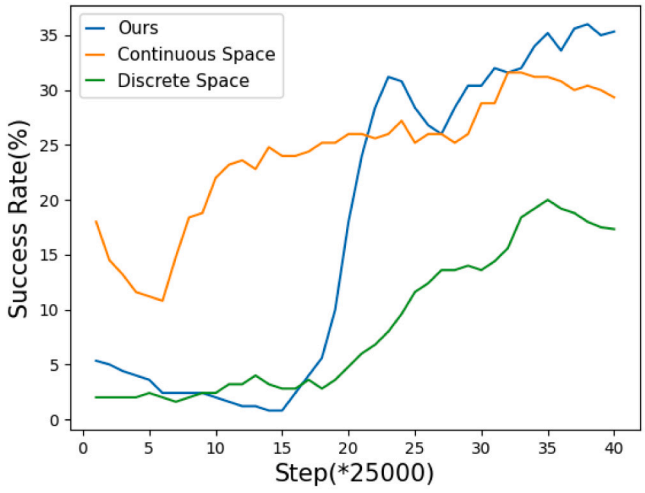


Fig. 8. Test success rates of the continuous space-based method [2], discrete space-based method [41] and our method.

factor α is set as -0.5 and $R_0 = -2$. The boundary threshold δ^B is 0.3. These parameters are empirically set.

We train the HL policy for 5000 steps in one environment and then switch to another environment until the total number of training steps reaches 1 million. Our proposed method takes about 0.0025 s for selecting a subgoal.

6. Results

To investigate the performance of our method, we perform different experiments to address the following aspects.

- Choice of hyperparameter in intrinsic reward
- Overall performance of our work in comparison with other RL-based mapless navigation approaches
- Effectiveness of our proposed reward function in comparison with other reward functions used by most HRL methods
- Effectiveness of the intrinsic reward
- Effectiveness of our proposed neural network structure

6.1. Choice of hyperparameter in intrinsic reward

In this section, we investigate the effect of the hyperparameter α in the intrinsic reward (Eq. (8)) on training with our model. To ensure the reward (R_{ex+in}) is not infinitely small, a minimum value $R_0 = -2$ (Eq. (9)) is set. Given the dynamic nature of the extrinsic reward R_{ex} and the memory decaying factor M_d, α primarily determines the value of N, the number of visits to the current location by the robot, against the reward size. That is, α predominantly governs the point at which the reward attains a lower bound. To investigate the effect of different α on our model, we set three values of α: -0.2, -0.5 and -2. A large N will be required for $\alpha = -0.2$ to attain the lower reward bound, but a small value for N is needed when α is -2.

In training, we test three models with different values of α every 25,000 HL steps, with 50 episodes per test. The test success rates during training are shown in Fig. 7.

It shows that when α is -0.5 and -0.2, the success rates tend to increase. However, the success rates are noticeably higher and remain stable at around 35%, when α is -0.5. On the other hand, when α is -2, the success rates steadily increase in the early stage, and start to decline after 500,000 training steps, eventually dropping to about 15%. Considering the overall performance, $\alpha = -0.5$ is deployed as a suitable choice for our reward function.

Table 1
Performance comparison with two non-hierarchical methods in the continuous and discrete space respectively [2,41].

Env	Target range	Continuous space [2]	Discrete space [41]	Ours
1	2–5 m	55.0%	38.4%	**59.4**%
	5–8 m	**50.4**%	23.4%	46.6%
	8–10 m	28.6%	10.2%	**38.2**%
2	2–5 m	56.6%	45.0%	**59.0**%
	5–8 m	37.8%	21.6%	**40.8**%
	8–10 m	21.6%	7.0%	**27.6**%
3	2–5 m	55.4%	45.0%	**57.8**%
	5–8 m	**43.4**%	12.8%	40.4%
	8–10 m	20.4%	5.2%	**26.6**%

6.2. Performance comparison with other RL-based approaches

6.2.1. Comparison with non-hierarchical methods

We first compare our approach with two non-hierarchical RL-based methods proposed by Tai et al. [2] and Marchesini et al. [41] respectively. Tai et al.'s work is based on DDPG, applicable to continuous action space, while Marchesini et al.'s approach is trained with the Double DQN algorithm [42], tailored for discrete space. Both methods utilise Lidar observations and the polar coordinates of the target as input. In addition, Tai et al.'s work incorporates the velocity from the previous timestep. The output of both approaches consists of the velocity commands. We train both models with the same reward functions proposed in [2,41], respectively. In the process of training, we test each method every 25000 steps, with 50 episodes per test. The success rates are shown in Fig. 8. It is clear that the success rates of all methods have a rising trend. However, the success rates of our method are noticeably higher and remain stable at around 35%. The methods with continuous space and discrete space reach about 30% and 18%, respectively.

After training, tests are first performed in three unseen environments with different difficulty levels. The success rates of the testing tasks are shown in Table 1.

It is clear that our method outperforms both non-hierarchical RL-based methods in 7 out of 9 tasks. The gap becomes more notable as the task difficulty increases. In the tests with the target range of 8 – 10 m, the continuous space-based method [2] and the discrete space-based method [41] achieve average success rates of 23.5% and 7.5%, respectively, while our method demonstrates a much higher success rate of 30.8%. As mentioned, tasks at a higher difficulty level would be more prone to the local minimum problem. As expected,

(a) Non-hierarchical method

(b) Our method

Fig. 9. An example in the test of scenario 1. Purple and blue circles represent the start position and the target position, respectively. (a) The orange line represents the robot's trajectory generated by the non-hierarchical method [2]. (b) The orange circles are the subgoals selected by our HL policy. The numerical sequence represents the selection order. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

our method demonstrates its superior performance compared with the non-hierarchical method.

Furthermore, as mentioned, we also test in three specific scenarios (Fig. 6). Both approaches fail to complete the tests in the chosen scenarios, while our method completes the tasks with success rates of 20.0%, 15.0%, and 30.0%, respectively.

Fig. 9 shows an example of scenario 1. The robot with the non-hierarchical method [2] is trapped in a local region. In contrast, our method first moves downwards, due to the large extrinsic rewards obtained from the trajectory that shortens the distance to the target location. However, when the robot finds that the current path would not lead to the target location, the intrinsic reward encourages the robot to explore more areas instead of wandering in a local place.

The above results show that the non-hierarchical method struggles when faced with complex local minimum problems, whereas our method substantially improves the success rate and demonstrates effectiveness.

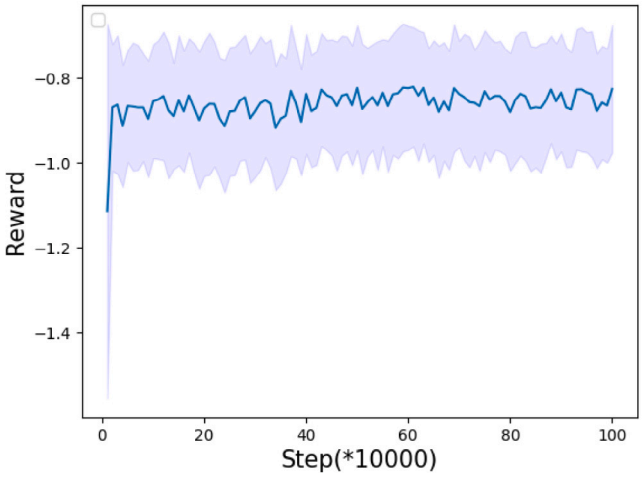


Fig. 10. Average rewards achieved by the agent when training the HL policy of HiRO.

6.2.2. Comparison with HRL-based methods

HiRO [24] is one state-of-the-art HRL approach that leverages a conventional MLP HL policy, trained via an off-policy RL algorithm, TD3 [43]. It operates in the continuous state space. The LL policy is trained to track target vectors that are generated by the HL policy. Notably, they train both policies jointly. In [24], they apply distance-based reward shaping to both policies to ensure comparability with our approach. We utilise the same HL (without intrinsic reward) and LL reward functions as proposed in our work. The average rewards per 10000 HL steps for training HiRO are shown in Fig. 10. The shaded region corresponds to the standard deviation of the rewards.

The average rewards achieved by the agent do not show any indication of improvement or stabilisation with increased training steps. Consequently, the results show that HiRO may struggle with complex navigation tasks. It also suggests that a continuous subgoal space may not be suited for the HL policy due to the considerable search space, thereby hindering the efficiency for policy training. Additionally, parallel training of both policies presents challenges in navigation tasks. Efficient training of the HL policy requires a stable LL policy, which may present considerable randomness in exploration during training, hence highly unstable. In the absence of a properly trained LL policy, the HL policy training may become unstable, resulting in failing to learn effective policies.

6.3. Effectiveness of the proposed reward function

To validate the effectiveness of our proposed reward function, we train another HRL-based model. As the LL policy only controls the locomotion of the robot, we still use our LL policy for this model. For the HL policy training, we utilise the reward function commonly used by most HRL-based methods [8–10], i.e., the agent receives a positive reward for reaching the target location and is penalised for collisions. Regarding NN structure, instead of incorporating an LSTM network as with our HL agent, we replace it with two MLP layers to output the Q value for each subgoal. For evaluation purposes, we exclude the elements of novelty from the reward function and historical information for the NN. The HL inputs to the model include the current Lidar readings and the polar coordinates of the target. In the following experiments, we refer to this as the "basic HRL" method.

The results of the tests in three unseen environments and specific scenarios are shown in Table 2 and Table 3, respectively.

The results show that our method performs better in all the tasks, and this dominance is particularly evident in the specific scenarios, which can be prone to the local minimum problem. The basic HRL-based method achieves a success rate of nearly zero in the three specific

Table 2
Comparison between our method and the basic HRL-based model in the three unseen environments.

Env	Target range	Basic HRL method	Ours
1	2–5 m	46.8%	**58.4%**
	5–8 m	32.6%	**46.6%**
	8–10 m	20.4%	**38.2%**
2	2–5 m	39.0%	**59.0%**
	5–8 m	15.0%	**40.8%**
	8–10 m	10.2%	**27.6%**
3	2–5 m	51.2%	**57.8%**
	5–8 m	29.8%	**40.4%**
	8–10 m	13.0%	**26.6%**

Table 3
Comparison between our method and the basic HRL method in the three specific scenarios.

Scenario	Basic HRL method	Ours
1	0.0%	**20.0%**
2	0.0%	**15.0%**
3	2.0%	**30.0%**

Fig. 11. An example of the basic HRL-based method in the test of scenario 1. The orange points represent the subgoals selected by the HL policy. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

tasks. Fig. 11 shows an example of when the basic HRL method is tested in scenario 1. The agent loiters in a local region; however, our method enables the agent to escape from the dilemma (Fig. 9).

It supports our hypothesis that our proposed reward function and the NN structure can encourage robots to explore unknown environments and enable them to have certain memory and reasoning abilities.

Furthermore, we conduct a comparative analysis of the training time. Both approaches entail substantial computational resources and necessitate considerable time for training. Specifically, our method demands more training time, approximately 216 h. Conversely, the basic HRL method requires a shorter time of about 168 h. We employ a workstation equipped with an Intel i9-10900X CPU (3.7 GHz × 20) and an Nvidia RTX-2080 TI GPU to conduct our experiments. As described, 10 environments have been selected for training in our work. In the training, we switch to a new environment after each training session. This requires frequent reloading and initialisation of new environments in the iGibson simulator. This process requires about 34.08 s, thus time-consuming. In addition, the HL policy needs to wait for the completion

Table 4
Comparison between our method and the method without the intrinsic reward in the three unseen environments.

Env	Target range	Without intrinsic reward	Ours
1	2–5 m	50.0%	**58.4%**
	5–8 m	39.2%	**46.6%**
	8–10 m	22.4%	**38.2%**
2	2–5 m	40.0%	**59.0%**
	5–8 m	20.4%	**40.8%**
	8–10 m	13.6%	**27.6%**
3	2–5 m	55.4%	**57.8%**
	5–8 m	35.4%	**40.4%**
	8–10 m	18.4%	**26.6%**

Table 5
Comparison between our method and the method without intrinsic reward in the three specific scenarios.

Scenario	Without intrinsic reward	Ours
1	0.0%	**20.0%**
2	0.0%	**15.0%**
3	3.0%	**30.0%**

of the task of moving the robot to the subgoal under the control of the LL policy for each HL training step. As such, the total training time is considerably long. The longer training time required by our method is partially due to the fixed number of HL training steps (1 million) for both methods. Since our method encourages the agent to explore, it will select more subgoals in various locations. Therefore, the probability of the agent colliding or choosing an inappropriate subgoal increases. We terminate the current episode and start the next episode when the robot fails to reach the selected subgoal. This process involves re-initialisation, which introduces additional time overhead and prolongs the overall duration of training. In contrast, the basic HRL method repetitively selects 'safe' subgoals that may lead to local minima, as shown in Fig. 11. As a result, the episode does not end prematurely, requiring fewer episodes in total and, consequently, reducing the training time.

6.4. Ablation study: Intrinsic reward metric

In this section, we study the effect of our proposed intrinsic reward in addition to the extrinsic reward, which is commonly deployed in previous work [2,4]. Specifically, we train our HL policy without the intrinsic reward and the LL policy remains the same. Due to the lack of intrinsic rewards in the reward function, the HL reward will not be changed dynamically based on past states and the subgoal selection is only related to current observation. Therefore, the HL input includes its current Lidar readings and the polar coordinates of the target with respect to the robot. Also, the LSTM network is removed to increase its learning efficiency. The success rates of the tests in the three unseen environments and the three specific scenarios are shown in Table 4 and Table 5, respectively.

The results show that our method has significantly higher success rates in all cases, suggesting that our proposed intrinsic reward plays an important role. Fig. 12 shows an example without the intrinsic reward tested in scenario 1. The robot moves downwards in order to obtain higher extrinsic rewards, and, due to the lack of exploration motivation and memory mechanisms, the robot will repeat the same path.

We can observe that the extrinsic reward is effective in most navigation tasks. However, the performance deteriorates in the presence of local minima, where the role of our intrinsic reward becomes more effective.

Fig. 12. An example of the method without intrinsic reward in scenario 1. The orange points represent the subgoals selected by the HL policy. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

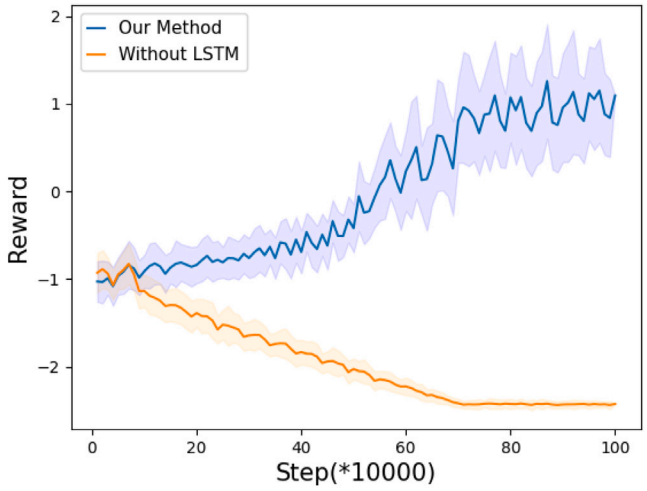


Fig. 13. Average reward for training steps using our method and the NN structure without LSTM.

6.5. Ablation study of the proposed HL network

To validate the design of the proposed HL network, we conduct two ablation experiments: (1) removing the LSTM network from our HL network, and (2) replacing the LSTM network with another network, named Frame Stacking [44,45], which enables the agent's memory capability.

6.5.1. Comparison between our method and the NN structure without LSTM network

We remove the LSTM network from our HL NN. The Q value of each subgoal is obtained through two MLP layers based on the HL input i_t^H, which only includes the current observation o_t^H. The rest, including the HL reward functions, training strategies, etc., are identical to our HL policy. In addition, the LL policy is also the same as ours. The average rewards per 10000 HL steps for our methods and the one without LSTM are shown in Fig. 13. The shaded region of each curve corresponds to the standard deviation of the rewards.

The figure shows that, without the LSTM network, the agent cannot learn an effective navigation policy. This is mainly due to the dynamic nature of our reward function. Consequently, the agent might select the same subgoal for identical observations but may receive varying rewards. In the absence of pertinent memory and prior experiences, this could lead to inconsistency of state–action mapping, failing to optimise the agent's policy efficiently.

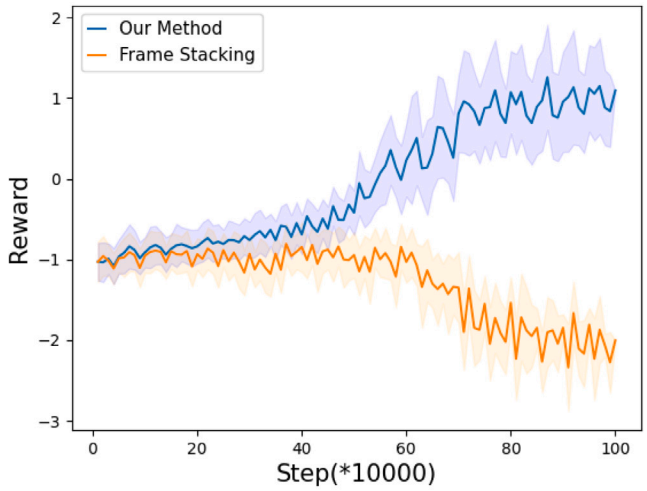


Fig. 14. Average reward for training steps using our method and the NN structure utilising frame stacking.

6.5.2. Comparison between our method and the NN structure using frame stacking

We replace the LSTM network with another widely used method, Frame Stacking, which is an effective way to enable NNs to learn temporal context knowledge [44–46]. Frame stacking is a kind of frame re-segmentation, which stacks temporal neighbouring states to form a state [45]. The inputs to the HL model include not only the current observation but also the last three HL steps' observations, denoted as $i_t^H = \{o_t^H \| o_{t-1}^H \| o_{t-2}^H \| o_{t-3}^H\}$. The rest is identical in terms of the reward function, training strategy, LL policy, etc. The average rewards per 10,000 steps for our method and the method with frame stacking are shown in Fig. 14.

From the figure, we can observe that the average reward of the method with frame stacking would not continue growing within its training time (1 million steps), indicating that it could not learn an effective policy and may require more training steps and resources. In contrast, the average reward of our model is steadily increasing, proving that the use of the LSTM network greatly improves learning efficiency.

6.6. Real world experiments

Experiments were conducted to demonstrate the effectiveness of the proposed method in real-world environments. The policy, trained within simulation environments, was deployed directly onto a mobile robot (Turtlebot 3) without any modifications. As illustrated in Fig. 15, the robot is equipped with two independent controllable wheels and a Lidar sensor (Laser Distance Sensor LDS-01), capable of 360-degree sensing, mirroring the simulation setup. The Robot Operating System (ROS) platform is used as the robot control and communication framework in our real-world experiments. Each wheel's velocity is computed and commanded by a laptop. It takes about 0.007 s for ROS to export the velocity command to Turtlebot. In addition, as the robot is operated under direct speed-mode control, the robot maintains its current velocity until a new command is received. The LL policy outputs zero motion only upon reaching a subgoal, ensuring smooth transitions

between consecutive subgoals. The total computational and communication time is approximately 0.01 s, during which the robot maintains its velocity. Due to the short interval and low-speed operation, the computational time will not have an impact on the robot's motion. The experimental setup featured a target location positioned behind a long wall, approximately 6 m in length, designed to test the proposed approach's ability to overcome local minima. In this scenario, the robot autonomously explores to first locate the door — the sole access point to the adjacent room — before proceeding to enter.

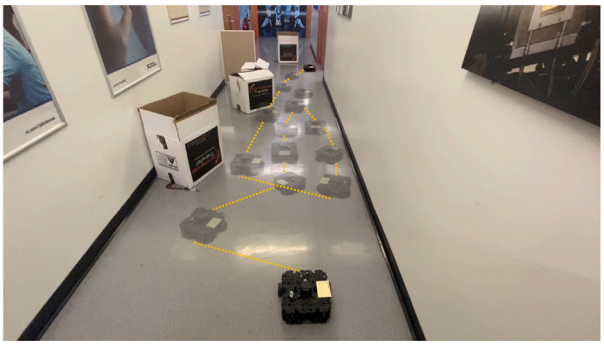
For comparative analysis, the non-hierarchical method was evaluated. Fig. 15(c) shows that, using this method, the robot initially moves straight towards the goal but soon finds itself trapped in a local area close to the goal yet obstructed by a wall. In contrast, our method prompted the robot to explore more extensively, enabling successful navigation into another room. Figs. 15(a) and 15(b) depict the robot exploring various subgoals along the corridor in both forward and backward directions, ultimately finding the door leading to the goal. This observation underscores our proposed framework's ability to encourage further exploration and effectively prevent revisiting already-explored areas.

7. Conclusion

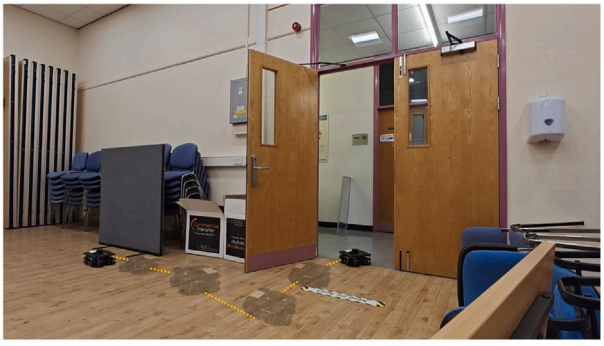
In this work, we proposed an HRL-based approach for mapless navigation. A novel reward function and a neural network are introduced for training the HL policy. Our HL reward function consists of two components, namely extrinsic reward and intrinsic reward. The extrinsic reward is provided by the environment and encourages the robot to move towards the target location. The intrinsic reward is inspired by the novelty theory [11], which suggests that animals will reward themselves for identifying novelty. We determine the magnitude of the intrinsic reward by the novelty of the current state. To decide the novelty of a state, we introduce a count-based method according to episode memory. In addition, to enhance the robot's ability to navigate in complex environments, we incorporate the memory decaying mechanism into the intrinsic reward. Also, some penalty elements are included in the overall HL reward function to ensure the LL policy's performance is considered when selecting subgoals. In addition, we propose to add an LSTM network to equip the agent with memory and reasoning capabilities.

Experimental results and analyses highlight the effectiveness of our proposed reward function and the NN structure in learning the navigation policy. Especially in complex tasks, our approach performs better and significantly improves the success rate.

Although our approach successfully learns mapless navigation policies through an end-to-end HRL framework, the size of the episode memory for the agent grows linearly with the states explored by the agent. This can become problematic when the agent needs to accomplish a task in large environments. In the future, we will focus on solving this problem, and one possible solution is to perform a secondary sampling of the states stored in the memory, storing only the states that are more valuable for navigation tasks. Ultimately, we will migrate our approach to the real world and deploy the algorithm on a real robot to decrease the gap between reality and simulation. Moreover, our method primarily relies on Lidar, where the lack of semantic information is a key limitation. Extending our approach to include visual sensors will provide richer environmental information, enhancing the overall performance and applicability of the system.



(a) Our method - 1



(b) Our method - 2

(c) Non-hierarchical method

Fig. 15. Experiments in a real environment with (a)–(b) our method and (c) the non-hierarchical method [2].

CRediT authorship contribution statement

Yan Gao: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Feiqiang Lin:** Visualization, Software, Methodology. **Boliang Cai:** Visualization, Software, Methodology. **Jing Wu:** Writing – review & editing, Supervision, Conceptualization. **Changyun Wei:** Writing – review & editing, Methodology, Conceptualization. **Raphael Grech:** Writing – review & editing, Methodology, Conceptualization. **Ze Ji:** Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

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, United Kingdom under the Industrial Fellowships programme for Dr Ze Ji (Grant No. IF2223-199), hosted by Spirent Communications.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2024.104815.

References

[1] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1928–1937.

[2] L. Tai, G. Paolo, 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] L. Xie, Reinforcement learning based mapless robot navigation, (Ph.D. thesis), University of Oxford, 2019.

[4] O. Zhelo, J. Zhang, L. Tai, M. Liu, W. Burgard, Curiosity-driven exploration for mapless navigation with deep reinforcement learning, in: ICRA 2018 Workshop on Machine Learning in Planning and Control of Robot Motion, 2018.

[5] M. Dobrevski, D. Skočaj, Deep reinforcement learning for map-less goal-driven robot navigation, Int. J. Adv. Robot. Syst. 18 (2021) 1729881421992621.

[6] Y. Zhu, R. Mottaghi, E. Kolve, J.J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, 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.

[7] J. Wöhlke, F. Schmitt, 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. 10682–10688.

[8] X. Zhou, T. Bai, Y. Gao, Y. Han, Vision-based robot navigation through combining unsupervised learning and hierarchical reinforcement learning, Sensors 19 (2019) 1576.

[9] 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.

[10] A. Staroverov, D.A. Yudin, I. Belkin, V. Adeshkin, Y.K. Solomentsev, A.I. Panov, Real-time object navigation with deep neural networks and hierarchical reinforcement learning, IEEE Access 8 (2020) 195608–195621.

[11] X. Ruan, P. Li, X. Zhu, P. Liu, A target-driven visual navigation method based on intrinsic motivation exploration and space topological cognition, Sci. Rep. 12 (2022) 3462, http://dx.doi.org/10.1038/s41598-022-07264-7.

[12] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, R. Munos, Unifying count-based exploration and intrinsic motivation, in: Advances in Neural Information Processing Systems, vol. 29, 2016.

[13] G. Ostrovski, M.G. Bellemare, A. Oord, R. Munos, Count-based exploration with neural density models, in: International Conference on Machine Learning, PMLR, 2017, pp. 2721–2730.

[14] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, P. Abbeel, # Exploration: A study of count-based exploration for deep reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 30, 2017.

[15] A. Pritzel, B. Uria, S. Srinivasan, A.P. Badia, O. Vinyals, D. Hassabis, D. Wierstra, C. Blundell, Neural episodic control, in: International Conference on Machine Learning, PMLR, 2017, pp. 2827–2836.

[16] W. Dodd, R. Gutierrez, The role of episodic memory and emotion in a cognitive robot, in: ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005, IEEE, 2005, pp. 692–697.

[17] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: Lstm cells and network architectures, Neural Comput. 31 (2019) 1235–1270.

[18] H. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: part i, IEEE Robot. Autom. Mag. 13 (2006) 99–110.

[19] Y. Chang, Y. Cheng, U. Manzoor, J. Murray, A review of UAV autonomous navigation in GPS-denied environments, Robot. Auton. Syst. (2023) 104533.

[20] F. Lin, Z. Ji, C. Wei, R. Grech, Localisation-safe reinforcement learning for mapless navigation, in: 2022 IEEE International Conference on Robotics and Biomimetics, ROBIO, IEEE, 2022, pp. 1327–1334.

[21] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, D. Batra, Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames, in: International Conference on Learning Representations, 2019.

[22] X. Yang, Z. Ji, J. Wu, Y.-K. Lai, C. Wei, G. Liu, R. Setchi, Hierarchical reinforcement learning with universal policies for multistep robotic manipulation, IEEE Trans. Neural Netw. Learn. Syst. (2021).

[23] A. Levy, G. Konidaris, R. Platt, K. Saenko, Learning multi-level hierarchies with hindsight, in: Proceedings of International Conference on Learning Representations, 2019.

[24] O. Nachum, S.S. Gu, H. Lee, S. Levine, Data-efficient hierarchical reinforcement learning, in: Advances in Neural Information Processing Systems, Vol. 31, 2018.

[25] T.G. Dietterich, Hierarchical reinforcement learning with the maxq value function decomposition, J. Artif. Intell. Res. 13 (2000) 227–303.

[26] P.-L. Bacon, J. Harb, D. Precup, The option-critic architecture, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, 2017.

[27] A.S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, K. Kavukcuoglu, Feudal networks for hierarchical reinforcement learning, in: International Conference on Machine Learning, 2017, pp. 3540–3549.

[28] M. Eppe, P.D. Nguyen, S. Wermter, From semantics to execution: Integrating action planning with reinforcement learning for robotic causal problem-solving, Front. Robot. AI 6 (2019) 123.

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

[30] B. Franks, et al., What do animals want, Anim. Welf 28 (2019) 1–10.

[31] N. Dilokthanakul, C. Kaplanis, N. Pawlowski, M. Shanahan, Feature control as intrinsic motivation for hierarchical reinforcement learning, IEEE Trans. Neural Netw. Learn. Syst. 30 (2019) 3409–3418.

[32] M. Hausknecht, P. Stone, Deep recurrent q-learning for partially observable mdps, in: 2015 Aaai Fall Symposium Series, 2015.

[33] A. Singla, S. Padakandla, S. Bhatnagar, Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge, IEEE Trans. Intell. Transp. Syst. 22 (2019) 107–118.

[34] H. Hu, K. Zhang, A.H. Tan, M. Ruan, C. Agia, G. Nejat, A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain, IEEE Robot. Autom. Lett. 6 (2021) 6569–6576.

[35] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al., Learning to navigate in complex environments, in: International Conference on Learning Representations, 2016.

[36] H. Tang, R. Yan, K.C. Tan, Cognitive navigation by neuro-inspired localization, mapping, and episodic memory, IEEE Trans. Cogn. Dev. Syst. 10 (2017) 751–761.

[37] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. Tchapmi, 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.

[38] 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.

[39] A. Kemp, D. Manahan-Vaughan, Hippocampal long-term depression and long-term potentiation encode different aspects of novelty acquisition, Proc. Natl. Acad. Sci. 101 (2004) 8192–8197.

[40] F. Xia, A.R. Zamir, Z. He, A. Sax, J. Malik, 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.

[41] E. Marchesini, A. Farinelli, Discrete deep reinforcement learning for mapless navigation, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2020, pp. 10688–10694.

[42] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, 2016.

[43] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.

[44] 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 518 (2015) 529–533.

[45] X. Tian, J. Zhang, Z. Ma, Y. He, J. Wei, Frame stacking and retaining for recurrent neural network acoustic model, 2017, arXiv preprint arXiv:1705.05992.

[46] V. Vanhoucke, M. Devin, G. Heigold, Multiframe deep neural networks for acoustic modeling, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 7582–7585.

Yan Gao received the bachelor's degree in machine design and manufacturing and automation from Harbin Engineering University, Harbin, China, in 2018, the master's degree in robotics from the University of Bristol, Bristol, U.K., in 2019. He is pursuing the Ph.D. degree with the School of Engineering, Cardiff University, Cardiff, U.K. His research is mainly focused on robot navigation via deep reinforcement learning.

Feiqiang Lin received the Bachelor degree and the Master degree from Beihang University, China in 2019. Currently, he is a Ph.D. student at School of Engineering, Cardiff University, U.K. His research interests include autonomous mobile robot navigation, reinforcement learning, drone control and simultaneous localisation and mapping (SLAM).

Boliang Cai received the bachelor's degree and master's degree in mechanical engineering from the college of mechanical and electrical engineering, Hohai University, Changzhou, China, in 2018 and 2021 respectively under the supervision of Dr. Wei. He is pursuing a Ph.D. degree in the Robotics and Autonomous Systems laboratory under the supervision of Dr. Ze, Cardiff University, Cardiff, U.K. His research is mainly focused on robotic mapless navigation with deep reinforcement learning algorithms under safety-critical and uncertain scenarios.

Jing Wu is a lecturer in computer science and informatics at Cardiff University, UK. She received B.Sc. and M.Sc. from Nanjing University, and Ph.D. from the University of York UK. Her research interests are in computer vision and graphics including image-based 3D reconstruction, face recognition, machine learning and visual analytics. She is on the editorial board of Displays and serves as a PC member in CGVC and BMVC.

Dr. Changyun Wei received the Ph.D. degree in Artificial 2015. Currently, he is an Associate Professor with the College of Mechanical and Electrical Engineering, Hohai University, China. He has published more than 30 papers in the fields of computer vision, artificial intelligence and robotics. His research interests include robotics, multi-agent systems, intelligent systems and computer vision.

Dr Raphael Grech possesses an Engineering Degree in Electrical and Electronics Engineering, an MPhil in Mobile Robot Control, and a Ph.D. in Multi-Robot Vision and Perception. He is a distinguished Technical Strategist specialising in Position, Navigation, and Timing (PNT) for Connected Autonomous Mobility at Spirent Communications. Raphael's main areas of expertise include Connected and Autonomous Vehicles, Digital Engineering, Advanced Robotics, Computer Vision, Machine Learning and Artificial Intelligence. Raphael is a UK Chartered Engineer, an IET Member, a Senior IEEE Member, and a Royal Academy of Engineering Visiting Professor at Cardiff University.

Ze Ji received the Ph.D. degree from Cardiff University, U.K., in 2007. He is a Senior Lecturer (Associate Professor) with the School of Engineering, Cardiff University, U.K. Prior to his current position, he was working in industry (Dyson, Lenovo, etc.) on autonomous robotics. His research interests include autonomous robot navigation, robot manipulation, robot learning, computer vision, simultaneous localisation and mapping (SLAM), acoustic localisation and tactile sensing.