Efficient Hierarchical Reinforcement Learning for Mapless Navigation with Predictive Neighbouring Space Scoring

Yan Gao[1], Jing Wu[2], Xintong Yang[1], Ze Ji[1]

***Abstract*—Solving reinforcement learning (RL)-based mapless navigation tasks is challenging due to their sparse reward and long decision horizon nature. Hierarchical reinforcement learning (HRL) has the ability to leverage knowledge at different abstract levels and is thus preferred in complex mapless navigation tasks. However, it is computationally expensive and inefficient to learn navigation end-to-end from raw high-dimensional sensor data, such as Lidar or RGB cameras. The use of subgoals based on a compact intermediate representation is therefore preferred for dimension reduction. This work proposes an efficient HRL-based framework to achieve this with a novel scoring method, named Predictive Neighbouring Space Scoring (PNSS).**

The PNSS model estimates the explorable space for a given position of interest based on the current robot observation. The PNSS values for a few candidate positions around the robot provide a compact and informative state representation for subgoal selection. We study the effects of different candidate position layouts and demonstrate that our layout design facilitates higher performances in longer-range tasks. Moreover, a penalty term is introduced in the reward function for the high-level (HL) policy, so that the subgoal selection process takes the performance of the low-level (LL) policy into consideration. Comprehensive evaluations demonstrate that using the proposed PNSS module consistently improves performances over the use of Lidar only or Lidar and encoded RGB features.

***Note to Practitioners*—This paper seeks to improve robot mapless navigation capabilities where the robot is expected to navigate to a goal location without knowing the map of the environment. This ability is highly demanded in many applications that require autonomous operations in unstructured environments, including both indoor and outdoor scenarios, involving tasks such as service robots for domestic and public environments, logistics in industrial warehouses, urban search and rescue missions, and disaster relief efforts, where detailed and accurate maps are difficult to obtain in advance. In this work, we focus on reinforcement learning-based mapless navigation. It is known that such methods struggle in complex long-range tasks, e.g. stuck in a local region by multiple objects. Therefore, this paper proposes a novel mapless navigation method inspired by human navigation behaviours.**

We enable a robot to split a long-range navigation task into multiple segments, by selecting and navigating to short-term goals. These subgoals are selected each time from a number of candidate positions located around the robot. The process stops when the robot reaches the final target location. When selecting a short-term goal, we use a deep neural network to predict the openness around each candidate subgoal position, named the Predictive Neighbouring Space Scoring (PNSS), from raw images and Lidar scans. In addition, we study the effects of different arrangements of candidate subgoal locations and select the optimal one. Experiments conducted in photo-realistic simulation environments demonstrate the effectiveness of our method, showcasing superior performance over baselines. It is worth noting that our agent is only trained in domestic environments using the iGibson simulator. For applications in other environments, additional training in more representative settings specific to corresponding scenarios will be necessary. In the future, our intention is to validate our methods in complex real-world environments and narrow the simulation-to-reality gap for long-horizon navigation tasks.

***Index Terms*—Mapless navigation, Deep Reinforcement Learning, Collision Avoidance, Motion Planning, Hierarchical reinforcement learning**

I. Introduction

Mapless navigation refers to the task of finding a collision-free path for a mobile robot with only partial observations of the environment. This ability is highly demanded for many applications in unstructured environments, such as urban search, rescue, disaster relief, or domestic service robots, where an accurate global map is not always available. Extensive research [1] has been carried out on path planning for navigation, and has seen advancements to tackle challenges such as planning under perception uncertainty [2]. However, conventional path planning algorithms typically require hand-crafted heuristic functions, which do not generalise well and, hence, would require considerable customisation for different environments [3], [4].

In recent years, deep reinforcement learning (DRL) has shown promising performance in many research fields, including mapless navigation tasks [5]–[9]. The working principle of DRL-based mapless navigation is straightforward, i.e., to reward the robot for getting closer to the target. The reward function is commonly based on the Euclidean distance from the robot to the target location with other terms such as penalties on collisions [5], [7]–[9]. Similarly, the performance is usually evaluated based on the success rates of achieving a goal or collision avoidance [8], [10]. The current work [5], [7] of DRL-based mapless navigation have shown promising performance in simple environments. However, they struggle in complex environments due to long decision horizons and sparse rewards that can only be earned when reaching the target location. As a result, the agent tends to be stuck in local

Manuscript received: Month, Day, Year; Revised Month, Day, Year; Accepted Month, Day, Year.

This paper was recommended for publication by Editor Editor A. Name upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by (organizations/grants which supported the work.)

[1]School of Engineering, Cardiff University, Cardiff, UK {gaoy84, jiz1, yangx66}@cardiff.ac.uk (Corresponding author: Ze Ji)

[2]School of Computer Science and Informatics, Cardiff University, Cardiff, UK wuj11@cardiff.ac.uk

regions, i.e, the local minimum problem, when encountering unknown and complex environments.

Hierarchical reinforcement learning (HRL) has been proven to be an efficient and promising framework to tackle this local minimum problem in navigation tasks [11]–[14]. This is because HRL enables a robot to decompose a long-horizon navigation task into several intermediate destinations (subgoals), which are much easier to reach than the long-term distant goal. Analogous to human navigation, a navigation system can be decomposed into two levels, where a high-level (HL) planning policy selects subgoals as short-term targets, and a low-level (LL) policy moves the robot to these short-term targets at the locomotion level [11]–[14]. Such hierarchical navigation systems tend to be more interpretable than non-hierarchical methods [15].

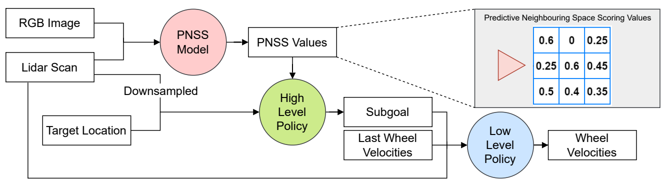


Figure 1. Overall Framework. The HL policy selects a subgoal based on the predicted PNSS values, the Lidar observation and the coordinates of the target coordinates. The PNSS values are predicted by the PNSS model for a set of explorable positions in front of the robot. The LL policy controls the robot to reach the subgoal. The process repeats until the robot reaches the target location.

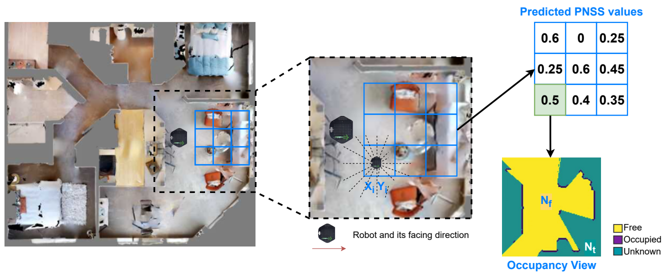


Figure 2. An example of the predicted PNSS values of 3×3 positions, which are located in the forward direction of its ego-centric view.

However, designing an HRL framework for mapless navigation is non-trivial and this paper seeks to improve existing frameworks in three directions as discussed below.

First of all, in mapless navigation, the subgoal layouts of most HRL algorithms are defined based on a local grid map. For example, a 3×3 grid centred around the robot, which allows 8 subgoals in the neighbouring cells [12]. Such a subgoal space limits the robot's exploration within a small area around its current location, while it would be more desirable to explore locations further away from the current location.

Secondly, many HRL methods assume that the low-level tasks are always achievable, i.e., the robot can always reach the subgoal selected by the HL policy [16], [17]. This assumption is unrealistic as it is common for the robot to encounter unreachable locations in complex environments.

Thirdly, mapless navigation can be based on different sensors including Lidar, cameras, or the combination of both [5], [10]. Training RL agents from raw images or Lidar scans has been proven highly inefficient [18], [19]. Although downsampled Lidar scans can provide sufficient information for short-term RL-based navigation [5], [7]–[9], it does not provide sufficient information for long-term navigation. On the other hand, image data contain much richer information, but has more redundancy and is more difficult to process. Existing works propose to represent images using an intermediate state space that covers essential information in a more compact way, but they generalise poorly in unseen environments or new tasks [12], [14].

To address these limitations, we propose a new HRL-based mapless navigation framework with three novel improvements, as shown in Fig. 1. Similar to existing HRL frameworks, our framework also has a high-level (HL) policy that selects subgoals and a low-level (LL) policy that controls the wheels to reach the selected subgoals. However, we first design a novel subgoal layout that encourages the robot to explore far places ahead (the blue grid in Fig. 1 and 2). Secondly, we include a penalty term in the reward function for the HL policy, so that the subgoal selection process would avoid subgoals that are physically unrealistic to achieve. Last yet most importantly, we propose a novel abstract state representation to reduce the input dimension for the HL policy, a method named Predictive Neighbouring Space Scoring (PNSS). The PNSS model extracts the essential information from raw images and Lidar, providing the HL policy with estimated values that measure the explorable space for each of the candidate subgoal locations, as shown by the matrix values in Fig. 2.

The main contributions are summarised below:

- We introduce a new learning-based framework for mapless navigation using RGB images and Lidar scans;
- A new metric, the PNSS, that evaluates the worthiness of navigating to a location, is introduced as part of the HL policy observation, providing a compact and informative input representation;
- We design a novel subgoal layout that improves navigation performance over existing subgoal layouts;
- We propose to penalise the HL policy for selecting subgoals that are not achievable by the LL policy, resulting in more reliable subgoal selection;
- Extensive comparative and ablation experiments are conducted to prove that our method generalises better and outperforms state-of-the-art methods within the iGibson environment [20].

The rest of this paper is organised as follows. Section II discusses the related work. Section III provides important preliminaries, and Section IV describes the proposed method in detail. Section V introduces our experiment setup followed by the results reported in Section VI. Section VII summarises the work.

II. Related Work

Mapless navigation [5], [6] was proposed to address some limitations of conventional navigation solutions, which require a series of operational steps, such as Simultaneous Localization and Mapping (SLAM) [21] or other map construction methods like [22], [23], path planning [24], [25], etc. The control performance of conventional methods highly depends on the simplified mathematical models, which lead to unreliable navigation systems [5], [6].

In contrast, RL enables an agent to learn control policies directly from high-dimensional sensory inputs, skipping several intermediate computation modules as used in classic navigation methods. For example, Tai et al. [5] use Lidar observations as the input for an RL policy, and trained it to generate motions for the robot to navigate and obstacle avoidance without a pre-built map.

Goal-conditioned Reinforcement Learning (GRL)-based methods have shown state-of-the-art performance for mapless navigation problems [5]–[9]. Unlike standard RL, GRL requires the agent to take different goals into consideration while making decisions [26], [27]. A typical goal space is a subspace of a state space, such as an image [28] or a feature vector [29], which describes the desired conditions that the agent is required to satisfy. In addition, language [30], rewards [31] and commands [32] are also used as representations of goals for different tasks.

While standard RL only requires the agent to complete a specific task defined by a reward function, GRL focuses on more general problems with multiple goals. In mapless navigation tasks, a robot is expected to have the generalisability to this class of tasks, for example by navigating to different target locations in different scenarios, where the different target locations can be modeled as goals in different episodes [5].

However, most of the RL-based approaches are tested using simple environments [8] and are mostly based on random exploration strategies or maximizing the entropy of the policy, which are insufficient for complex real-world environments. In addition, most RL-based mapless navigation models have long decision-making horizons, so that the rewards are sparse as they are usually only obtained when the target location is reached or the robot has a collision with an obstacle.

HRL has been shown to be well suited for learning goal-conditioned behaviours in long-horizon and complex tasks [18], [19], [33]. This approach typically designs HL policies that operate on a coarser time scale and controls the execution of LL policies. The main working principle of recent HRL methods, such as Option Critic [34], Feudal Networks (FuN) [35] and HiRO Networks [19], is to combine the ideas of subgoal generation and policy combination with neural networks and RL. FuN and HiRO combine an HL policy that can generate a subgoal at a lower frequency with an LL policy that receives intrinsic rewards for reaching subgoals. In goal-conditioned tasks, the HL policy usually selects a subgoal to provide to the LL model, which is easier for the LL policy to reach than the distant final goal [16], [17]. The HL policy can be learned to iteratively predict sequences of intermediate subgoals, which can then be used as targets for LL policies [19], [36], [37]. Also, as an alternative to iterative planning, some methods generate the sequence of subgoals using a divide-and-conquer approach [38], [39]. HRL method can be divided into several categories according to the HL representations, such as symbolic representation [17], predicate-logic-based representation [16], and learned skills [40]. In addition, some methods do not set the representation form for the HL model in advance, and connect the experience in the buffer to a graph to form the representation through the accessibility estimation network [41].

Several methods are proposed to apply HRL to robot navigation tasks [11]–[14]. A rough map of the environment is usually required, represented in the format of small grids or hex-grids that provide occupancy information of neighbouring regions [12], [42]. Usually, such navigation tasks can be employed in two layers, where the higher layer decomposes the trajectory in discrete primitives corresponding to the decomposed grids, and the lower layer deals with movement control [12], [14].

However, some HRL methods do not take into account the LL policy performance in the HL planning [16], [17]. Also, some existing HRL-based navigation models [12], [14] would require prior knowledge, such as the grid map of the environment, which is not realistic. Additionally, we have observed that many HRL methods struggle to perform well in complex environments, due to the unrepresentative features or high-dimensional redundant data to represent the state as the HL policy input [18], [19]. The simplicity of subgoal space layout [12] could also impede the performance. In this work, we propose a new HRL-based mapless navigation model with a new subgoal space form. Our HL policy can select a distant subgoal based on a more informative and compact state representation. To the best knowledge of the authors, there is no previous work of HRL for mapless navigation in photo-realistic environments containing rich visual features, such as iGibson or Habitat [43], [44]. Most HRL-based mapless navigation methods are tested in simple synthetic environments, such as ROS/Gazebo, and would require a rough map in advance [12]–[14], which is different from our problem definition of mapless navigation.

III. Preliminaries

We formulate both the HL policy and the LL policy as goal-conditioned reinforcement learning (GRL) problems. We use two different RL methods to train our policies, which are DQN [45] and DDPG [46]. In this section, we briefly recall the related preliminaries.

Markov Decision Process: An MDP can be represented by a tuple $\langle S, A, R, p, \gamma, \rho_0 \rangle$. 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 of a state to an action. 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 $Q^{\pi}(s, a)$ represents the same quantity when an action a is taken at state s. The objective of RL is to find an optimal policy that maximises the value function [47].

Goal-conditioned Reinforcement Learning: The GRL problem adds a goal space G to the MDPs of the standard RL paradigm. Standard RL is to achieve a single goal, while a GRL agent tries to maximise a multi-goal reward function $R(s_t, a_t, g)$, resulting in a goal-conditioned value function $Q^{\pi}(s, g, a)$ or policy $\pi(a|s, g)$ [26]. The HL and LL policies in our method are both goal-conditioned.

Deep Q-Network: DQN is an off-policy algorithm applicable to discrete action space. It uses a neural network to approximate the Q function. In practice, researchers often use a target network to stabilise training processes [45]. During learning, the agent uses a common *epsilon-greedy* exploration method. It takes a random action with a probability ϵ and takes an action according to the learnt Q function with a probability $1-\epsilon$. ϵ is decayed linearly during the course of training. In our method, DQN is used to train the HL planning policy.

Deep Deterministic Policy Gradient: DDPG is an off-policy actor-critic RL algorithm for continuous control tasks. It uses two neural networks to approximate Q-function and a deterministic policy, updating the two parameters alternately [46]. The target network is proven to reduce the 'overestimation' of DDPG and improve the learning performance of the agent [46]. During learning, the agent uses an exploration strategy that takes random actions with a probability ϵ and takes learnt actions with Gaussian noises with a probability $1-\epsilon$. In our method, DDPG is used to train the LL control policy.

IV. Methods

We propose a novel HRL framework, in which an HL policy selects subgoals in an abstract space and an LL policy is responsible for the locomotion control of the robot to reach the corresponding subgoals, as illustrated in Fig.1. For the selection of subgoals, we introduce a prediction mechanism to evaluate the worthiness level of a location for further exploration, named Predictive Neighbouring Space Scoring (PNSS). Briefly, the PNSS module is to predict the area of unoccupied space that can be observed at a corresponding location. A deep neural network is trained for the prediction of the PNSS values given the measurement of local surroundings. These PNSS values are then provided to the HL policy for the selection of the next subgoal. We design a reward function for training the HL policy including a penalty that occurs when the LL policy fails to reach a subgoal. Our framework contains three key modules:

- PNSS module for estimating the PNSS values based on the current observation,
- HL policy that decides the next subgoal, and
- LL policy that is responsible for the locomotion control of the robot to reach the target subgoals selected by the HL policy.

A. PNSS model

As mentioned, we introduce a metric to score the explorable worthiness of a given location. As illustrated in Fig. 2, the explorability of a location is estimated from the current egocentric observation of the robot. The worthiness for exploration is related to the free neighbouring space available at each corresponding location. We introduce the PNSS metric, defined as the proportion of the observable free space of a local region (see Fig. 2), formulated as:

$$s(x_i, y_i) = \frac{N_f}{N_t} \tag{1}$$

where x_i and y_i represent the coordinate of the ith cell with respect to the robot's coordinate frame, for which the score is calculated. N_t represents the area of the local region of interest (represented by 128×128 cells), and N_f represents the area of free space (i.e. number of non-occupied cells) measurable with Lidar observations at $[x_i, y_i]$, $s \in [0, 1]$.

The PNSS model is trained in a supervised manner. We use the iGibson simulation environment [20] to obtain the ground-truth scores as training labels. For data collection, the robot is randomly placed at any arbitrary position of concern. As illustrated in Fig. 2, since the robot is equipped with a Lidar with the Field of View (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 PNSS using Eq. (1).The higher the score is, the more free space the robot observes in the local region. A score of 0 indicates that the neighbouring areas are fully occupied.

In addition, for predicting PNSS, we also use a standard forward-looking camera that can only provide RGB images with a horizontal FoV of 58 degrees (Asus Xtion pro). We believe that the visual features from the RGB images with rich information will improve the prediction accuracy. This is validated in Section VI-B.

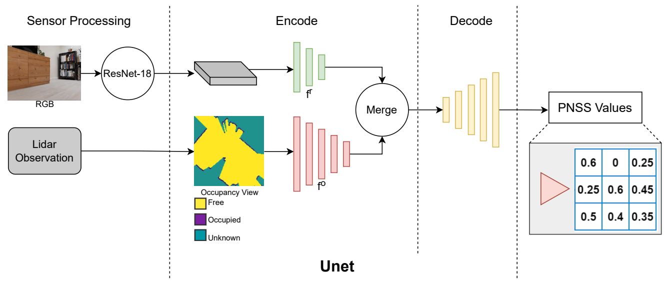


Figure 3. The PNSS model extracts features from the RGB image firstly. The Lidar observation is then projected into a 2D occupancy view. A UNet network is used for predicting the PNSS values for a 3×3 grid map.

We adopt a network architecture inspired by the occupancy anticipation model [48]. The PNSS network predicts the PNSS values for a few candidate locations based on the current observation of RGB images and Lidar data. The main components are shown in Fig. 3 and summarised below:

- The sensor pre-processing module contains two parts: 1) feature extraction from RGB images using a pre-trained ResNet-18 model, and 2) the current Lidar observation that will be transformed into the occupancy view.
- Given the RGB features and the occupancy view, we encode them using UNet encoders [49] individually. The RGB features are encoded using a stack of three convolutional blocks denoted as f^r. The occupancy view is processed by a stack of five convolutional blocks into a feature vector, denoted by f^o.
- We then combine f^r with f^o through the Merge module to construct a combined feature f^g. The Merge module contains layer-specific convolution blocks to merge all layers in $[f_i^r, f_i^o]$ [48]. It can be formulated as $f^g = \text{Merge}(f^o, f^r)$.

- The combined feature, f^g, is decoded using a Unet decoder that outputs the PNSS values for the corresponding positions in the subgoal space, formulated as $S_{PNSS} = \sigma(\text{Decode}(f^g))$. Fig. 3 shows a 3×3 subgoal space, as an example.

B. High-Level policy

The HL policy is used to generate the next short-term navigation subgoal for a robot to navigate. The following subsections describe the main components of the HL policy, namely 1) the observation, 2) the action/subgoal space, and 3) the reward function.

1) Observation: The observation for the HL policy comprises three main components, denoted by $o_t^H = \{o_L || o_{PNSS} || g_{p\_t}^H\}$, where $||$ represents vector concatenation combining two vectors into one higher-dimensional vector. o_L is the Lidar reading at the robot's current location, $o_{PNSS} = \text{Flatten}(S_{PNSS})$ is the vector of PNSS values by flattening the S_{PNSS} matrix, whose size depends on the subgoal space, and $g_{p\_t}^H = (r_t, \theta_t)$ represents the polar coordinates of the target location with respect to the robot frame. Our work is based on the assumption that the global coordinates of the target and robot locations are available for a mapless point navigation task, usually known as the PointNav task [50].

2) Action space (Subgoals): The action space of the HL policy produces the subgoals for navigation that will serve as the goals for the LL policy, i.e., $A^H = G^L$, where, A^H is the high-level action space and G^L is the low-level goals.

Efficient robot learning for the high-level policy requires careful consideration of the action space in terms of the locations of the subgoals and the complexity of the subgoals (number of actions). Some previous works use the surrounding neighbour regions centred at the robot's current pose and most of them rely on 360 Lidar only. In this work, we propose to deploy a 3×3 grid subgoal space, located in front of the robot's current view. We consider the method more intuitive because the forward-looking camera in our work would provide rich information about the environment in front of the robot, hence predicting more accurate PNSS values. On the other hand, these positions cover a larger range of explorable areas compared to those used by existing works [12]. We compare the performance of different choices of such explorable positions in section VI-E. Due to the spatial constraints with the introduced subgoal space, when the above subgoals are invalid, (e.g., falling outside of the environment), we also introduce some additional rotation actions in the subgoal space (14 angles in our work). In such cases, the HL policy would encourage the robot to rotate with an angle. Therefore, there are in total 23 subgoals, i.e., HL actions, available for selection.

3) Reward function: The HL policy will be rewarded or penalised in the different cases, as formulated below:

$$R^H = \begin{cases} r_{arrive}^H & \text{if } d_t \leq \delta^H \\ r_{collision}^H & \text{if collision} \\ r_{overtime}^H & \text{if } t_L \geq T \\ r_{approach}^H & \text{if approaching the subgoal} \\ r_{rotate}^H & \text{if rotate} \end{cases} \tag{2}$$

where

- r_{arrive}^H is a positive value when the robot reaches the target location, i.e., when its distance to the final target location, d_t, is within a radius δ^H;
- $r_{collision}^H$ and $r_{overtime}^H$ are the penalty values that occur when the LL policy fails to reach a selected position due to collision or timeout;
- $r_{approach}^H = d_{t-1} - d_t$ is the change of distance from the robot to the target location between two consecutive time steps. $r_{approach}^H$ is positive when the robot is getting closer to the target, and negative when moving away;
- $r_{rotate}^H = -c_r(|\frac{7\theta}{pi}|)$ is a term that penalises when the HL policy selects a subgoal to rotate. The greater the rotation angle θ, the greater the penalty. c_r is a weighting factor that scales the penalty value. The term r_{rotate}^H is used to encourage smoother motions.

It is worth noting that, as discussed before, the reward function above takes the LL policy's capabilities into consideration for HL decision making. It is not realistic to assume that all LL tasks can be completed. We, therefore, introduce rewards, e.g. $r_{collision}^H$ and $r_{overtime}^H$, for the HL model such that it is penalised when the LL model fails to reach a subgoal. This will encourage the HL model to consider reachability in its subgoal selection.

C. Low-level policy

The LL policy is used to train a robot to learn how to reach any given goal in a short range. It interacts directly with the environment selecting actions for the robot.

This subsection describes the details of the LL policy in three parts: the observation, the actions and the reward function.

1) Observation: The observation of the LL policy comprises three parts, denoted as $o_t^L = \{o_L || a_{t-1}^L || g_{p\_t}^L\}$, where $||$ denotes vector concatenation. o_L denotes the Lidar data of its local surroundings. The action from the last timestep, a_{t-1}, is included in the observation because, due to the inertia, the robot's motion commands will have effects on its successor steps. Last, $g_{p\_t}^L$ is the target location represented in the polar coordinates of the robot frame.

2) Action: The LL policy directly controls the wheel velocities of the robot (a TurtleBot in our experiments), $a_t^L = \{v_{left}, v_{right}\}$. Each velocity action lasts for 0.1 seconds.

3) Reward function: The reward function for training the LL policy is as follows:

$$R^L(o_t^L, a_t^L, g^L) = \begin{cases} r_{arrive}^L & \text{if } d_t \leq \delta^L \\ r_{collision}^L & \text{if collision} \\ r_{approach}^L & \text{otherwise} \end{cases} \tag{3}$$

where

- r^L_{arrive} is a positive value, when the robot reaches the target location, i.e., when its distance to the target d_t is within a radius δ^L.
- $r^L_{collision}$ is the penalty value that occurs when the robot collides with an obstacle; and
- $r^L_{approach} = c_d(d_{t-1} - d_t)$ is the distance reward, where $(d_{t-1} - d_t)$ is the change of the distance from the robot to the target location at two consecutive time steps, and c_d is a weighting factor.

Algorithm 1: HL policy of our HRL model

```
Given:
    • Pretrained LL policy $\pi_{LL}$ ;
    • HL policy $\pi_{HL}$, HL buffer $D_{HL}$ ;
Initialise DQN of HL policy
for $m \leftarrow 0$ to $M$ epoch do
    # Sample training environment Env 1, 2, 3...(once
      m is changed, sample another different
      environment in turn)
    for $j \leftarrow 0$ to $J$ training episode do
        $t_{HL} = 0$,
        $done = 0$
        Sample a target location $g_t$
        while do
            # Obtain a subgoal $g_s$
            $g_s \sim$ epsilon-greedy$(\pi_{HL}(o^H_t))$
            while do
                $t_{LL} = 0$
                $a_{t_{LL}} \sim \pi_{LL}(o^L_t)$
                $t_{LL} = t_{LL} + 1$
                $o^L_t = o^L_{t+1}$
                if $t_{LL} \geq T_{LL}$, or $g_s$ reached, or
                  collision then
                    break
            $t_{HL} = t_{HL} + 1$
            if $t_{LL} \geq T_{LL}$, or $g_t$ reached, or collision,
              or $t_{HL} \geq T_{HL}$ then
                $done = 1$
            $D_{HL} \leftarrow (o^H_t, g_s, R^H, o^H_{t+1}, \text{done})$
            $\lambda_{t_{HL+1}} \leftarrow$ Adam $(\lambda_{t_{HL}}, D_{HL})$
            $o^H_t = o^H_{t+1}$
            if $done = 1$ then
                break
```

V. Experiments

A. Simulation environment

The iGibson simulator [20] is used in our work. It is based on the Gibson dataset [51] that includes a large number of complex and photo-realistic 3D indoor environments, such as houses, offices, restaurants and coffee shops. We use a Turtlebot provided by the simulator in our experiments. It is equipped with a camera that generates $3 \times 480 \times 640$ RGB images and a Lidar with 360 laser beams covering a FoV of 360 degrees.

In our work, 23 environments are selected from the Gibson dataset, where 10 are used for the PNSS model training, 10 are used for the HL policy and the LL policy training and 3 are used for testing (shown in Fig. 4).

One main cause of failures is the local minimum problem, i.e., robot being trapped in a local region. To better demonstrate the improvement of our work in terms of solving the local minimum problem, the tests are performed at three difficulty levels. The difficulty levels are defined based on the distances from the robot to the destinations. Respectively, the three difficulty levels correspond to tasks with distance ranges of $[2, 5]$, $[5, 8]$, and $[8, 10]$. Tasks in each of the above categories are initialised with randomly generated starts and destinations that range between the corresponding bounds above. For example, tasks in the first category have target distances of between $[2, 5]$ meters. Tasks with large ranges would be more challenging and would include scenarios with more complex local maps that tend to lead to local minima. Each test is performed with 500 episodes to compute the average success rate. The same start and goal positions are used for different algorithm configurations to ensure fair comparisons.

B. PNSS model training

In the iGibson environment, we can directly acquire the ground-truth occupancy view of a given location. In this work, the occupancy view is represented as a 128×128 matrix, with each cell labeled as occupied, free, or unknown. Then from the occupancy view, the PNSS value for the location can be calculated. We obtained 5000 sets of data from randomly selected poses in each environment, each consisting of the egocentric RGB image, the Lidar scan and the calculated ground-truth PNSS value. In total 50000 sets of data are collected, where 40000 of them are used for training, 5000 are for validation and 5000 are for testing.

C. LL policy training

The LL policy is trained separately. For each episode, the robot is randomly placed in an environment. Since the LL policy is only concerned about short-range navigation, we limit the distance to the destination for each LL episode. The target is randomly sampled at least 0.5 meters away from the robot, but within a square that is centred at the robot, with each side of 4 meters. The parameters in Eq. 3 are set as below. The arrival reward, $r^L_{arrive} = 20$, is given, when the robot is no more than $\delta^L = 0.36$ meters away from the target position (the chassis radius of the Turtlebot is 0.36m). The collision penalty is set as $r^L_{collision} = -3$. The hyperparameter c_d in the distance reward, $r^L_{approach}$, 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 to the next environment, until a total of 1 million timesteps is reached. At each timestep, the agent explores the environment by taking random actions with a probability $\epsilon = 0.2$ and learnt actions with Gaussian noises with a probability $1 - \epsilon = 0.8$.

We use DDPG to train the LL policy. The actor network for DDPG has three MLP layers with the same size of 512.

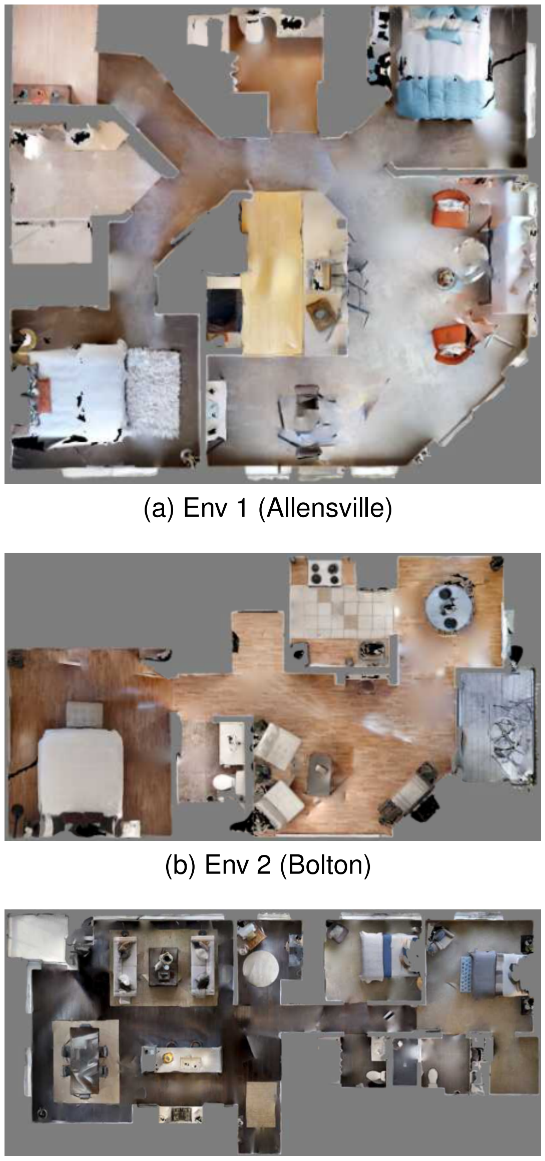


Figure 4. Experiment environments for testing.

The critic network also has three MLP layers, the size of the first and last layers is 512, and the dimension of the second layer is 514, with two extra dimensions for the action. ReLU activation is used for each layer on both the actor and the critic networks except for the output layers. Hyperbolic tangent is used for the actor networks to activate the last layer, while the critic network has no activation on the output.

D. HL policy training

After the PNSS model and the LL policy are trained, we then train the HL policy. For each episode, the robot is placed at a random location in an environment. We randomise the target positions within a sphere centred at the robot's position, with a distance between the corresponding range bounds, as introduced in subsection V-A. An episode ends in three cases: 1) when the LL policy cannot reach the subgoal within 1500 timesteps; 2) when the HL policy cannot reach the target position within 400 selections of subgoals; and 3) the robot collides with an obstacle. An episode is considered successful, when the robot is no more than $\delta^H = 0.86$ meters away from the target position, and an arrival reward, $r^H_{arrive} = +20$, is given. The hyperparameter c_r in the rotation penalty term is empirically set to 0.05. The collision and overtime penalties are set as $r^H_{collision} = -3$ and $r^H_{overtime} = -3$.

We train the HL policy for 150 episodes in one environment and continue to the next environment, until a total number of 60000 episodes is reached. The HL policy uses a *epsilon-greedy* to explore the environment, with *epsilon* decaying linearly from 1 to 0.05 within the first $42,000$ episodes, and being kept 0.05 to the end of training. These parameters are empirically set after trial-and-errors.

We use Deep Q Learning to train our HL model. The network is represented by two MLP layers of sizes 512 and 256. ReLU activation is used only on the output of the first layer. The output of the network is the Q values of selecting the subgoals with a given observation.

E. Subgoal layouts

Since there are infinite combinations of subgoal layouts (HL action space), it is impractical to evaluate all of them. We focus on three subgoal layouts, as shown in Fig. 5. Each cell in the layouts is 0.5 meter in width and length.

- **Layout 1** simply takes the 8 neighbouring cells as its next candidate subgoals. This is the same as used in [12].
- **Layout 2** includes another 3 cells in the forward direction of the robot. This would allow a robot to explore its next subgoal with a larger range.
- **Layout 3** is used in our work, as introduced in section IV-B2. We eliminate the cells behind the robot and introduce a 3×3 grid in the forward direction of the robot. This will encourage the robot to explore further distance in the forward direction. We also include 14 rotation subgoals to avoid the robot being stuck in local minimum.

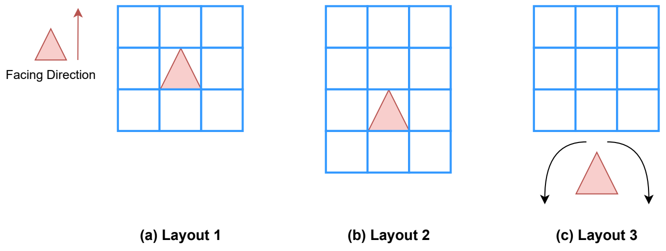


Figure 5. Three different layouts we mainly focus on (a) Layout 1 simply takes the 8 neighbouring grids as its subgoal space. (b) Layout 2 adds three more grids in front of the robot on the basis of Layout 1. (c) Layout 3 includes 9 subgoals in the forward direction and 14 rotation subgoals.

VI. Results and discussions

To study the performance of our proposed method, we carried out comprehensive experiments and analyses from various aspects, as follows:

- Overall performance of our work in comparison with other RL-based mapless navigation approaches
- Performance of the PNSS value prediction
- Choice of RL algorithms for training
- Effectiveness of the PNSS module in comparison with using Lidar data and encoded RGB image features
- Comparison of different subgoal layout configurations
- Effectiveness of the proposed reward function

A. Performance comparison with other RL-based approaches

To evaluate the performance of our work, we compare our method with three other RL-based algorithms, including non-hierarchical and hierarchical methods respectively.

1) Non-hierarchical RL-based methods: There are a number of non-hierarchical RL-based mapless navigation methods. Most of them are inheritance of the work proposed in [5]. For comparative evaluation, we choose two approaches to compare with our proposed method, including one DDPG-based approach for the continuous action space [5] and one Double DQN-based algorithm [52] in the discrete space [7].

The input for both methods includes Lidar observations and the polar coordinates of the target. The work in [5] also includes the velocity at the previous timestep. The output is the velocity commands, except that one is in the continuous action space and the other one is discrete. It is worth mentioning that the same network architecture as used in [5] is deployed for the LL policy in our work, with the same reward function.

As mentioned, one reason for choosing the above methods is that it is known to be the most popular solution for RL-based mapless navigation. It should be noted that there is no widely-deployed non-hierarchical navigation solution that uses both Lidar and raw images, which are also found inefficient based on our preliminary experiments. Also, the PNSS module is not compatible with this non-hierarchical configuration. Therefore, in this work, we could not perform a direct comparison here, and only perform the comparison between our method using PNSS and the above-mentioned methods without PNSS.

The experiments were performed in three environments with different difficulty levels with all methods. The success rates of the testing tasks are shown in Table I. It is obvious from the table that our method outperforms both the non-hierarchical approaches [5], [7] in all three cases, except for tests with ranges of $2-5$m in Environment 3. The improvement is considered significant, especially for tasks with longer ranges. In the most difficult experiments (target range: $8-10$m) in three environments, the success rate of our method is nearly 40% higher than the discrete non-hierarchical method [7], indicating that our method outperforms the non-hierarchical methods when faced with complex scenarios. We speculate that this is attributed to the ability of our method to tackle the local minimum problem which is more often encountered with longer ranges.

Fig. 6 illustrates a few cases of the local minimum problem, where a robot is trapped by a corner, a wall, or furniture. As can be seen in the figure, with the HL subgoals, our method would encourage the robot to explore further and could more effectively tackle such situations. This is attributed to the PNSS values for tackling the local minimum problem, which is more often encountered with longer ranges.

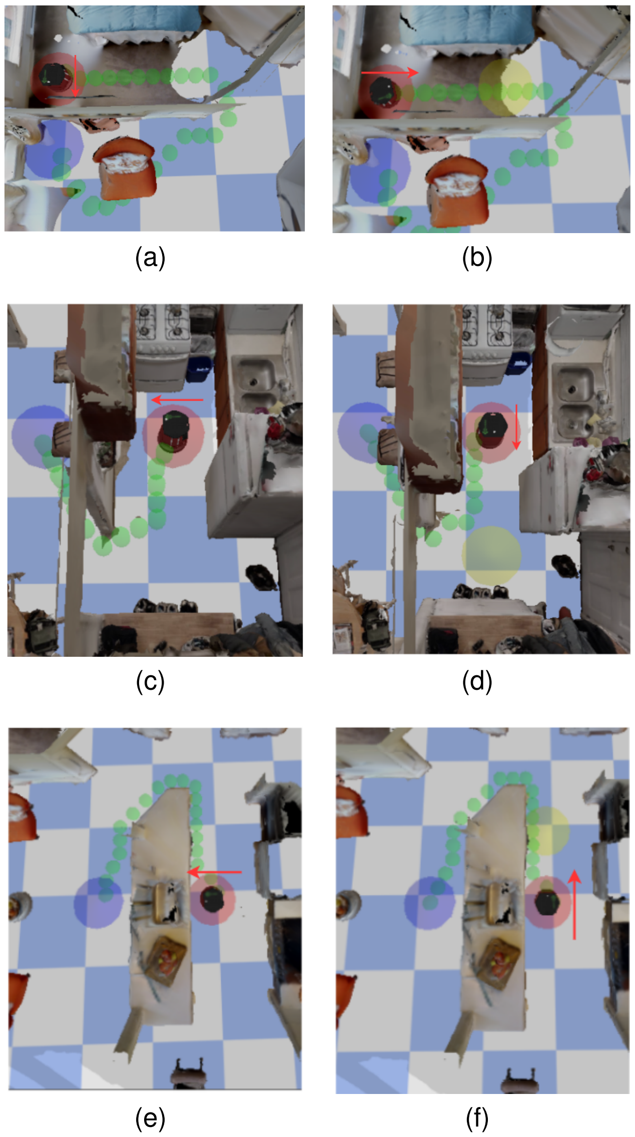


Figure 6. Examples of the robot being trapped by obstacles. The red and blue circles are the start positions and the target positions. The green lines are ground-truth paths. The red arrows are the robot's heading directions. (a), (c), (e) are three cases where the robot keeps heading towards the direction of the target and cannot get around the obstacle, using the non-hierarchical Lidar-based mapless navigation method [5]. (b), (d), (f) are the solutions provided by our model in these situations. The yellow circles are the subgoals given by our HL policy that leads the robot to bypass the obstacles.

2) HRL method: HiRO [19] is a state-of-the-art HRL method. Its HL policy uses conventional MLP neural networks, trained by TD3 [53], operating in the continuous state space as the LL policy. Since the subgoal space is continuous, it is not compatible to use our proposed PNSS model. Therefore, the inputs to the HL policy for this study include Lidar observations and the polar coordinates of the target location. In [19], they train their HL and LL policies jointly. The reward functions for both policies are distance-based. To ensure a fair comparison, we utilise the same reward function as defined in our work. The average reward per 1000

Table I
PERFORMANCE COMPARISON WITH TWO NON-HIERARCHICAL METHODS IN THE CONTINUOUS AND DISCRETE SPACE RESPECTIVELY [5], [7]

Env	Target range	Continuous space [5]	Discrete space [7]	Ours
1	2-5m	55.0%	38.4%	**59.4%**
	5-8m	50.4%	23.4%	**58.8%**
	8-10m	38.0%	14.2%	**52.6%**
2	2-5m	68.0%	45.0%	**71.6%**
	5-8m	57.6%	21.6%	**61.2%**
	8-10m	42.0%	14.0%	**52.0%**
3	2-5m	**74.8%**	65.0%	71.6%
	5-8m	65.0%	29.8%	**65.6%**
	8-10m	61.0%	26.2%	**64.6%**

episodes for training the HL policy of HiRO is shown in Fig. 7.

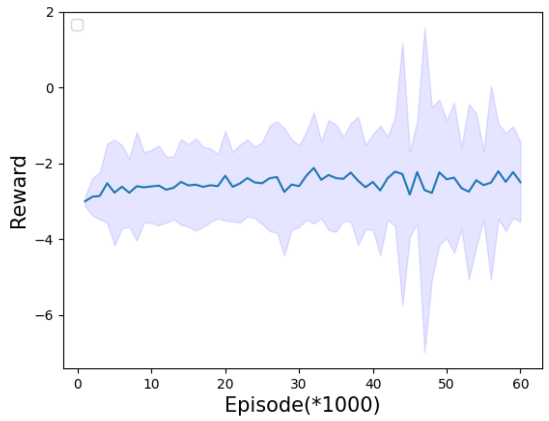


Figure 7. Average HL rewards achieved by the agent (HiRO), the shaded area represents the standard deviation.

As shown in the figure, the average rewards achieved by the agent do not show a sign of improvement by increasing the number of episodes for training. The rewards are all negative values too. Therefore, we consider that HiRO is unsuitable for such navigation tasks. It suggests that a continuous subgoal space may not be suitable for the HL policy, due to the large search space of subgoals for efficient policy training. Furthermore, simultaneous training of the HL and LL policies is more challenging for this problem due to the nonstationary problem. Efficient training of the HL policy requires a stable LL policy, which would need random explorations before it stabilises. The untrained LL policy would yield an unstable HL policy training and hence causes inefficiency or even failure in learning effective policies.

B. PNSS value prediction

In this section, we evaluate the performance of the PNSS prediction module and test our hypothesis that RGB observation will help predict the PNSS value of a location. We show the results with 1) only Lidar observation and 2) both Lidar and RGB observations. The model is trained to predict the PNSS values using three subgoal layouts, as shown in Fig. 5. The performance of the prediction is measured using the L1 distance between the model's predicted PNSS values and the ground truth.

Table II
PERFORMANCE OF PNSS VALUES PREDICTION USING DIFFERENT SENSING MODALITIES AND SUBGOAL LAYOUTS BASED ON THE L1 DISTANCE METRIC.

Subgoal Space	Lidar	Lidar + RGB
Layout 1	0.0996	**0.0921**
Layout 2	0.0889	**0.0791**
Layout 3	0.1016	**0.0813**

As shown in Table II, it is clear that the accuracy of PNSS estimation is higher, when RGB observation is considered in addition to Lidar data, producing smaller errors based on the L1 distance metric. We have evaluated several NN structures and loss functions and found that it is difficult to further reduce the training errors. However, the effect of adding RGB observations is rather notable. This is expected as our hypothesis is that visual observations embed richer semantic information that would help in determining available free space for navigation. For the above reasons, we will only use RGB and Lidar data for PNSS estimation in the rest of the experiments.

To statistically evaluate the PNSS prediction module, we record the average prediction errors (L1 distance) for each of the 5000 groups in the test dataset. Fig. 8 shows the distributions of the prediction errors. The prediction errors for each subgoal in each layout are also plotted on a heat map, as shown in Fig. 9.

As can be seen in Fig. 8, Layout 1 has the largest error distribution that mainly ranges in 0.04-0.13. In contrast, Layout 2 has the smallest error distribution range. Considering that Layout 2 is similar to Layout 1 except the three extra subgoals in front of the robot, the reduction of prediction errors indicates that the model is more accurate in predicting the PNSS values of the subgoals in the front region of the robot, resulting in a lower mean error. Layout 3 contains more subgoals in front of the robot. The overall error distribution is similar to the other two layouts, except that it has the smallest median value, hence higher accuracy. On the other hand, the results further suggest that RGB images can contribute to the prediction of PNSS values, as the camera's FoV is only for the forward direction.

The heat map (Fig. 9) shows the prediction errors of the PNSS values for each subgoal in three layouts. The maximum error is around 0.11, which shows good reliability and accu-

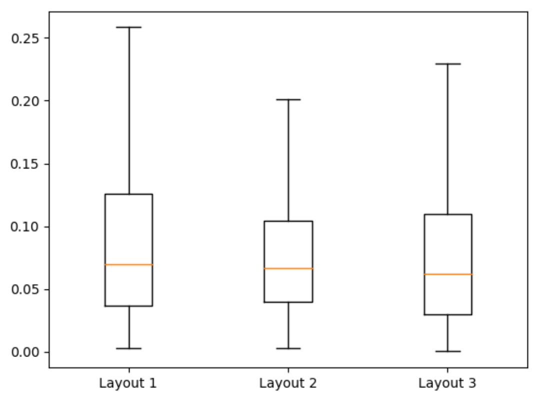


Figure 8. Box plot for average PNSS prediction errors with three layouts.

racy of our model. In addition, the prediction is less accurate in the PNSS of the subgoals on the rear and two sides than the forward direction of the robot. This demonstrates that the RGB images contribute to the improvement of the prediction accuracy, as mentioned above. Considering that Layout 3 has the lowest median PNSS value and overall more evenly distributed errors across all subgoals, we choose Layout 3 for our work.

C. RL algorithms used to train the HL and LL policies

To decide on the algorithms for training the HL and LL policies, we select several widely used RL algorithms respectively. We first use the TD3 [53], SAC [54], and DDPG [46] to train the LL policy separately. The training strategies and reward functions for the three methods are identical, as described in Sections IV-C and V-C. Since the LL policy is responsible for short-range navigation, we set the distance between the target location and the robot's initial location to $1-3$m for each episode. For testing, each test lasts for 100 episodes and the success rates of the three methods are as follows. DDPG produces the highest success rate of 77%, and the success rates trained by the TD3 and SAC are close, 72% and 71% respectively. This suggests that DDPG is more suitable among the three tested algorithms for short-term navigation tasks. Therefore, we select DDPG as the RL algorithm to train our LL policy.

For the HL policy, since the action space is discrete, we choose two widely used algorithms designed for discrete action space, namely DQN [45] and Double DQN [52]. Both methods have identical training strategies and reward functions, and utilise the same LL policy trained above.

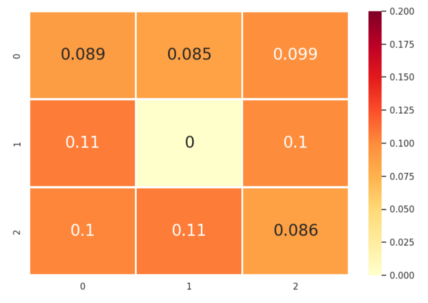
In the process of training, we test both methods every 3000 episodes, with 50 episodes per test. The success rates are shown in Fig. 10.

The success rates of both methods start to rise after about 30000 training episodes, with the method using DQN maintaining a higher success rate than DDQN. The DQN-based policy stabilises at approximately 50% success rate and can reach up to 56%. In contrast, the DDQN-based method maintains a stable success rate of about 40% and remains below 50% overall.

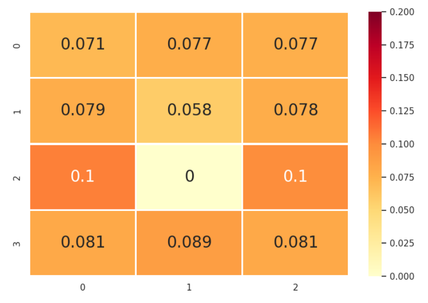
Both methods are tested in the three environments (Fig. 4). The success rates are shown in Table III. The agent trained by DQN achieves the best performance in 7 out of the 9 tasks. DDQN is only better in two configurations in Env 3, where, however, the gap between the two is considered insignificant at about 4%. However, in other configurations, the improvement by using DQN is considerably more obvious. Considering the overall higher success rates and simpler implementation, the DQN is deployed as a more suitable choice for our HL policy.

D. Ablation study: observation modality

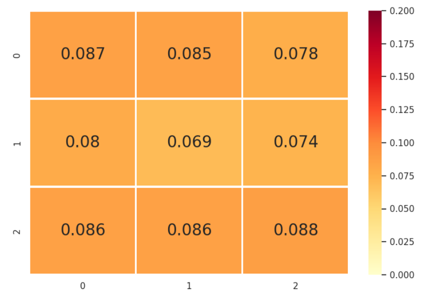
In this section, we evaluate the importance of the PNSS prediction in improving the HL policy and conduct ablation studies with different observation modalities, as follows:



(a) Layout 1



(b) Layout 2



(c) Layout 3

Figure 9. Heat map for the prediction errors for each subgoal in each layout. In (a) and (b), 0 represents the location of the robot.

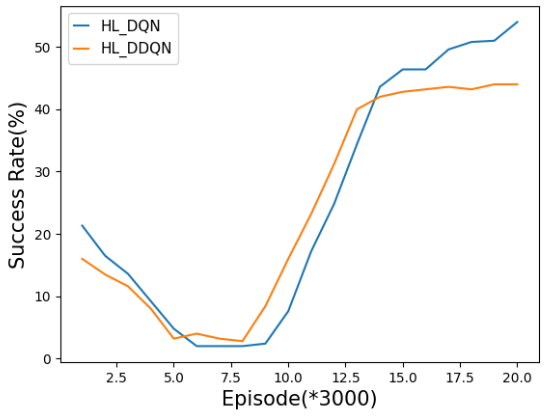


Figure 10. Success rates of the DQN and Double DQN algorithms for training the HL policy

Table III
TEST SUCCESS RATES WITH DIFFERENT RL ALGORITHMS USED TO TRAIN THE HL POLICY

Env	Target range	DDQN	DQN
1	2-5m	59.2%	**59.4%**
	5-8m	54.0%	**58.8%**
	8-10m	43.6%	**52.6%**
2	2-5m	68.6%	**71.6%**
	5-8m	55.8%	**61.2%**
	8-10m	46.8%	**52.0%**
3	2-5m	**74.6%**	71.6%
	5-8m	65.0%	**65.6%**
	8-10m	**68.8%**	64.6%

- Only Lidar observations
- Lidar observations concatenated with RGB image features, which are extracted based on [10] using ResNet18.
- Our method, which is similar to the above, but uses predicted PNSS values instead of extracted ResNet18 features.

Fig. 11 shows the average reward per 3000 episodes and the test success rates of Layout 1 (Fig. 5a), which is one commonly used subgoal space [12]. As mentioned above, RGB images are encoded using ResNet18 to obtain a compact representation, as used in [10]. Fig. 11a shows that the agent with PNSS achieves the highest reward (blue line). The one with only Lidar and target information achieves the second (orange line) and the one with RGB image observation achieves the least (green line). Moreover, the shaded region of each curve is corresponding to the standard deviation of the rewards. It can be seen that both the upper and lower bounds of the rewards obtained by the agent with the PNSS values are higher than the other two methods, indicating that our method has a better overall performance. Although the increase in reward does not demonstrate a statistical significance visually, this is considered primarily attributed to the inclusion of a large number of random episodes for training, encompassing varying levels of difficulty. During training, we also conduct testing for each method by running 100 episodes per test and calculating the success rates correspondingly. These tests are performed at regular intervals of every 3000 episodes of training. The success rates are shown in Fig. 11b for layout 1. It is clear that after 30000 training episodes, the success rates of all methods start to rise. Our proposed method (PNSS values and Lidar) clearly reaches a higher success rate than the other two methods, and stabilises at around 38%. The methods with only Lidar data and both RGB features and Lidar reach about 33% and 20% respectively.

The success rates of tests in unseen environments (Environment 1, 2, and 3) are shown in Table IV for Layout 1. The results demonstrate that our proposed method using both the PNSS values and Lidar outperforms the other two sensing modalities in all cases. Combining RGB ResNet18 features and Lidar produces the least performance overall, while the Lidar-based method performs in between the other two. We observe that directly using the ResNet18 features or RGB data would not contribute to the performance improvement. This is due to the redundancy in the RGB features, where the HL policy struggles to extract information helpful for the task. For the above reason, we do not consider ResNet18-encoded features for the other two layouts here.

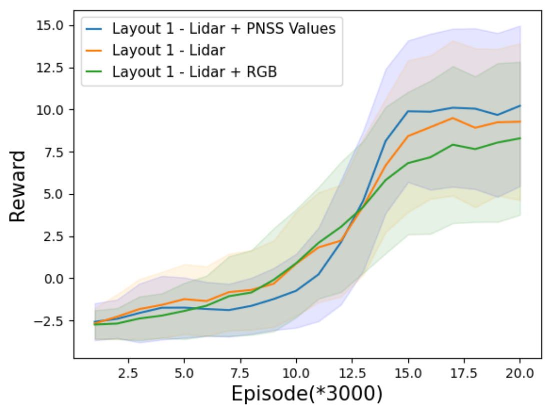
The same experiments are conducted with Layout 2 and Layout 3 too. The average reward per 3000 episodes and the test success rates during training are shown in Fig. 12 and Fig. 13 respectively. The corresponding success rates of tests in Environments $1, 2$, and 3 are illustrated in Table V and Table VI.

These results suggest that extracting task-relevant information by pre-processing raw observations encoded as the PNSS values has played an important role in improving its performance. On the other hand, directly using the Lidar observations makes it more challenging to train the policy, because Lidar observation can only provide information about the robot's observed surroundings. It should be also noted that the tasks with larger ranges show more obvious performance improvement while using the PNSS data as observations.

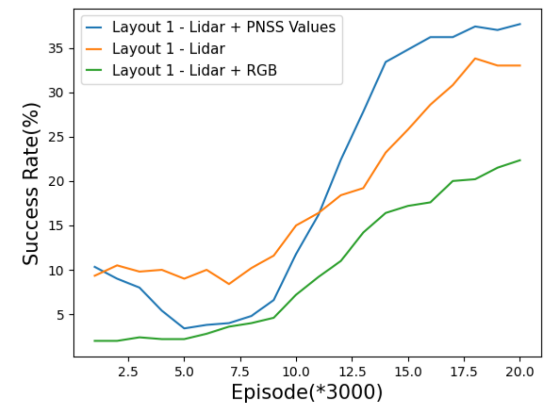
Table IV
TEST SUCCESS RATES WITH DIFFERENT OBSERVATION MODALITIES-LAYOUT 1

Env	Target range	Lidar	Lidar + RGB	Lidar + PNSS
1	2-5m	50.8%	48.2%	**53.4%**
	5-8m	44.6%	33.4%	**54.0%**
	8-10m	31.8%	25.8%	**39.8%**
2	2-5m	60.2%	55.0%	**64.8%**
	5-8m	47.0%	34.2%	**49.6%**
	8-10m	30.6%	22.6%	**38.8%**
3	2-5m	68.0%	66.6%	**73.2%**
	5-8m	57.8%	55.0%	**63.2%**
	8-10m	55.8%	46.8%	**62.2%**

Despite the promising results with our proposed approach, the overall success rates still present a gap from reliable deployment for real-world problems. One of the main reasons is the limited sensing capability and the complexity of the indoor environments in our work. The iGibson environment includes complex layouts and furniture of various shapes. Due to the sensing limitation, some furniture parts cannot be detected by Lidar, such as the legs of chairs which can

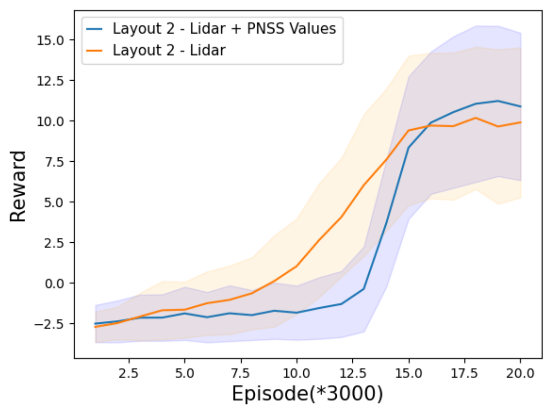


(a)

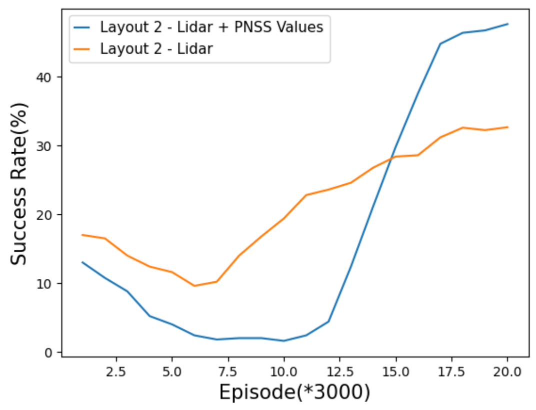


(b)

Figure 11. Average rewards (a) and test success rates (b) achieved by the agent with different observation modality-layout 1



(a)



(b)

Figure 12. Average rewards (a) and test success rates (b) achieved by the agent with different observation modality-layout 2

Table V
TEST SUCCESS RATES WITH DIFFERENT OBSERVATION MODALITIES-LAYOUT 2

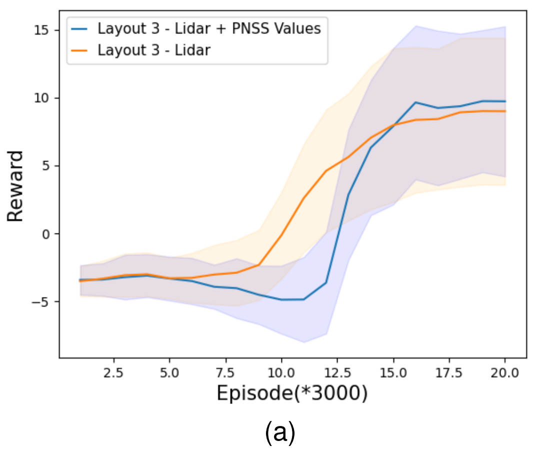
Env	Target range	Lidar	Lidar + PNSS
1	2-5m	52.2%	**57.0%**
	5-8m	47.0%	**55.6%**
	8-10m	31.2%	**42.2%**
2	2-5m	60.0%	**67.8%**
	5-8m	48.2%	**56.4%**
	8-10m	32.2%	**41.8%**
3	2-5m	72.6%	**72.8%**
	5-8m	63.2%	**66.6%**
	8-10m	56.4%	**62.4%**

Table VI
TEST SUCCESS RATES WITH DIFFERENT OBSERVATION MODALITIES-LAYOUT 3

Env	Target range	Lidar	Lidar + PNSS
1	2-5m	55.8%	**59.4%**
	5-8m	49.8%	**58.8%**
	8-10m	33.4%	**52.6%**
2	2-5m	59.6%	**71.6%**
	5-8m	39.6%	**61.2%**
	8-10m	25.4%	**52.0%**
3	2-5m	70.4%	**71.6%**
	5-8m	61.6%	**65.6%**
	8-10m	57.2%	**64.6%**

easily lead to collisions. Also, some obstacles, like coffee tables, are above the horizontal scanning plane of the Lidar but can collide with the robot. The collisions will trigger signals for terminating corresponding episodes, hence limiting the average success rate. Overall, our method improves the success rate in most of the difficult tasks compared to other methods.

On the other hand, we also consider the practicality of deploying the policy for real-world control. The average computational times of the models with different input modalities are measured too. Our proposed method takes about 0.358s for calculating the PNSS values and the subsequent action. The method that uses Lidar and RGB image features takes about 0.186s for computation. The method with only Lidar needs the least computational time of 0.172s. This is reasonable as our work introduces another tailored step of predicting the PNSS values for the navigation task. Despite the longest computational time required, we consider our work highly



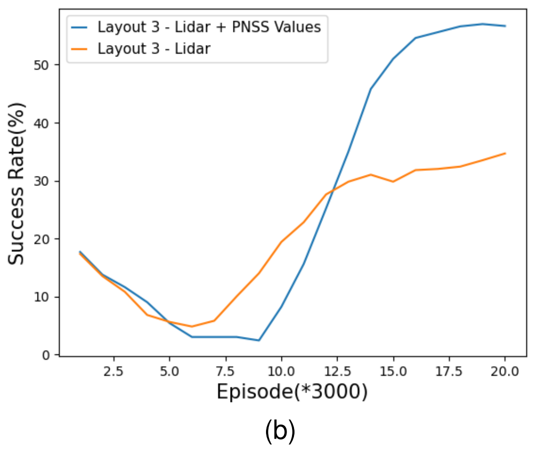


Figure 13. Average rewards (a) and test success rates (b) achieved by the agent with different observation modality-layout 3

feasible for real-world indoor robots that are usually operating at a low speed, as used in our work.

In addition, we also compare the training time required for the three methods. All methods are computationally expensive and would take a long time to run. Our method using both Lidar and PNSS values would take the longest time of about 8 days. The methods with 'Lidar' and 'Lidar + RGB features' as inputs require similar time durations of about 6 days. We use a workstation with an Intel i9-10900X CPU (3.7GHz x 20) and an NVidia RTX-2080 TI GPU. This is partially attributed to the fixed number of episodes (60,000) for all three configurations. With the neighbour scoring mechanism of the PNSS method, a more valid sub-goal could usually be chosen by the HL policy for the LL policy to execute. The lower chance of collision with the PNSS-based method would lead to a longer time for each episode of the LL policy. The other methods, however, tend to have higher chances of earlier terminations due to collisions, hence shorter overall training time.

E. Ablation study: subgoal layouts

In order to investigate whether our choice of subgoal layout (see Fig. 5c) helps improve navigation performance, we conduct experiments with all three different subgoal layouts (Fig. 5). The HL policy reward function for the case of 'Layout 3' subgoal space is described in subsection IV-B, while the reward functions for the other two cases are the same except that they have no rotation penalties.

Table VII
TEST SUCCESS RATES WITH DIFFERENT SUBGOAL SPACES

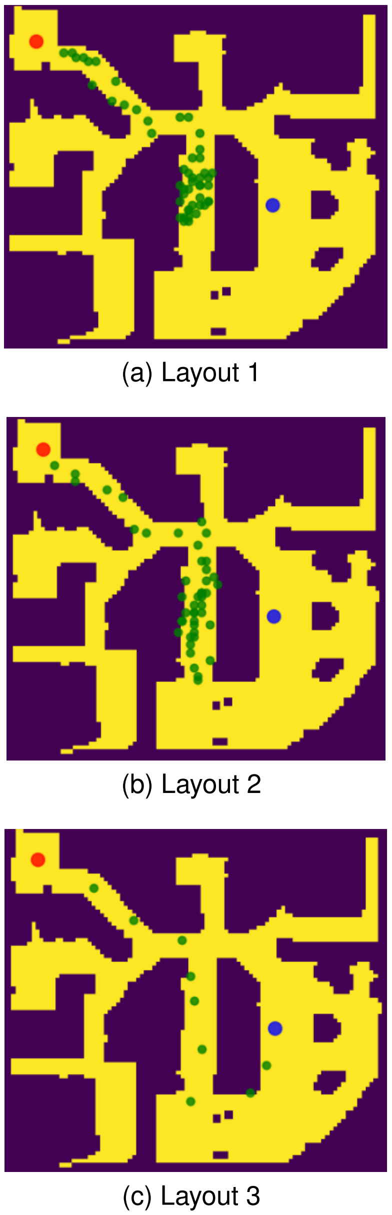
Env	Target range	Layout 1	Layout 2	Layout 3
1	2-5m	53.4%	57.0%	**59.4%**
	5-8m	54.0%	55.6%	**58.8%**
	8-10m	39.8%	42.2%	**52.6%**
2	2-5m	64.8%	67.8%	**71.6%**
	5-8m	49.6%	56.4%	**61.2%**
	8-10m	38.8%	41.8%	**52.0%**
3	2-5m	**73.2%**	72.8%	71.6%
	5-8m	63.2%	**66.6%**	65.6%
	8-10m	62.2%	62.4%	**64.6%**

We first report the test success rates of each layout in Table VII. One can see that our subgoal space design helps the agent achieve the best performance in 7 out of 9 tasks. Our proposed method has the highest success rate in all experiments executed in Env 1 and 2. In Env 3, which is slightly less complex than the other two (higher average success rates and more regular geometric features), the performances for the three layouts are considered similar, except that Layout 3 outperforms in the long range case. Our chosen layout shows clear overall superior results and is considered more suitable for the majority of tasks. Especially in the difficult tasks ($8-10$m), the HRL model using 'Layout 3' form has an average success rate 56.4%, while the average rates of the models using 'Layout 2' and 'Layout 1' are 48.8% and 46.9% respectively. The result supports our hypothesis in that the proposed subgoal space helps the robot navigate better as it allows the robot to explore subgoals with a larger range.

Fig. 14 illustrates an example of a long-range navigation task with different subgoal layouts. As can be seen, the policies using subgoal Layout 1 and Layout 2 do not perform well. The robot is trapped in local regions in both cases. However, our method would encourage the robot to explore further and effectively tackle the local minimum problem.

By further investigating the results between the 'Layout 1' and 'Layout 2' cases, one can see that the robot performs better when it includes more subgoal options in front of it, i.e. Layout 2 in this case. This then suggests that predicting whether there is more free space further ahead is useful for navigation tasks.

To understand how the subgoals are selected in the actual navigation tasks, we tested 100 episodes in Env 1 (Fig. 4a), and counted the occurrences that each subgoal was selected in the successful episodes. Fig. 15 shows the results grouped into the three difficulty settings. We observe that all the subgoals in Layout 3 were selected. When the task is relatively simple, that is, when the target position is close to the initial position, the subgoals selected by our HL policy are concentrated in the 9 grids directly in front of the robot, indicating that the robot is more confident in performing forward translation motions. However, with the increase in the difficulty of the tasks, the



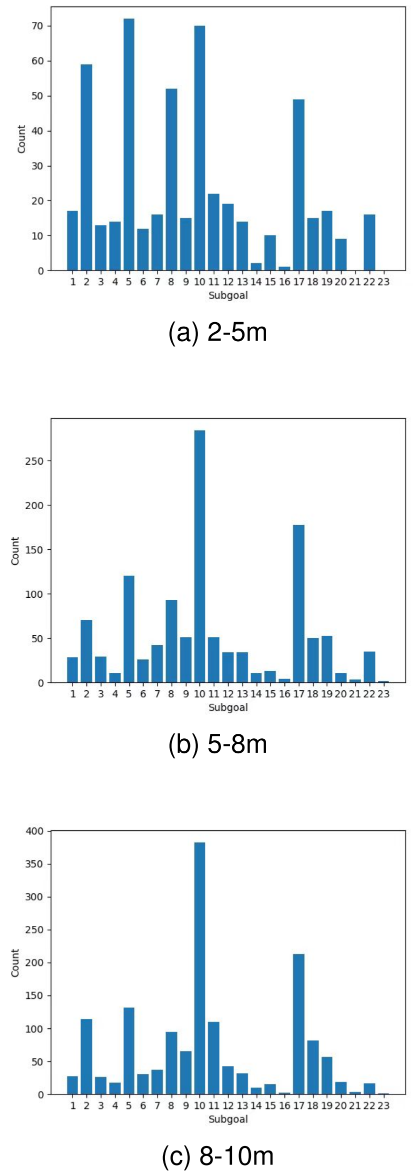


Figure 14. An example of the local minimum problem in a long-range navigation task. Red and blue circles represent the start position and the target position respectively. Green circles are the subgoals selected by the policies with different layouts.

Figure 15. We record the total number of occurrences different subgoals were selected in all successful episodes when our model is tested in Env 1 (Fig. 4a) on tasks of different difficulty settings. 1-9 refers to the 9 grids in front of the robot from near to far and from left to right. 10-16 indicates that the robot rotates to the right. The higher the number, the greater the rotation angle. Similarly, 17-23 represents HL policy selects the left-rotating subgoals.

robot has to face a more complex scenario, so the proportion of rotation subgoals increases considerably, especially with small angle rotations to the left (subgoal 10) or right (subgoal 17), because, in our reward setting, the larger the rotation angle the HL policy selects, the greater the penalty will be. Only when necessary will the robot choose to rotate at a large angle in place.

Although the subgoal space layout is set arbitrarily in our work, this provides another insight into HRL mapless navigation problems that subgoal layouts could be optimised or even learned in future work.

F. Reward function

To validate the design of our reward function, we perform the ablation experiment by removing the timeout penalty element from the HL reward function, which is proposed with considerations of the LL policy's capability. Specifically, we remove the item $r^{H}_{overtime}$ that penalises the agent if it could not reach the selected subgoal within a certain period of time, and keep the rest the same. After training, we test the two methods, with and without the timeout penalty. The success rates are shown in Table VIII.

The result shows that the inclusion of the timeout penalty can overall improve the success rates in most tasks. As hypothesised, unreachable or difficult-to-reach subgoals are

Table VIII
TEST SUCCESS RATES WITH DIFFERENT REWARD FUNCTIONS FOR TRAINING THE HL POLICY

Env	Target range	Without timeout penalty	With timeout penalty
1	2-5m	**60.6%**	59.4%
	5-8m	56.4%	**58.8%**
	8-10m	46.8%	**52.6%**
2	2-5m	71.4%	**71.6%**
	5-8m	59.2%	**61.2%**
	8-10m	48.2%	**52.0%**
3	2-5m	**74.4%**	71.6%
	5-8m	65.4%	**65.6%**
	8-10m	**68.2%**	64.6%

not desired and the long time costed for the LL policy to attempt navigating to these subgoals should be integrated into the reward function as penalty. Without the timeout penalty, the agent will continue choosing unreachable subgoals and subsequently impair the overall performance.

VII. Conclusion

In this paper, we proposed a novel HRL-based mapless navigation method, where the high-level policy generates a subgoal for the low-level policy, while the low-level policy is responsible for maneuvering the robot to the given subgoal at the locomotion control level. For the HL policy, we introduce a novel scoring method, namely the Predictive Neighbouring Space Scoring (PNSS), to obtain a compact state representation. The PNSS values indicate the explorable space around a given position, and a PNSS model is trained to predict the PNSS values for candidate positions around the robot based on the robot's current view. The PNSS values can be then deployed by the HL policy as observations in addition to Lidar.

Extensive experiments have been carried out to demonstrate the effectiveness of the proposed methods. This PNSS-based observation has shown significant improvements in success rate, due to its compactness and task-related nature. We demonstrated that our work outperformed Lidar-based policy or policy with both Lidar and ResNet18-based RGB image features.

Different layouts for the subgoals are also studied, demonstrating the effectiveness of our proposed method. One notable improvement of the navigation performance is for longer range navigation, where Lidar-based methods or non-hierarchical methods would more likely be stuck in local minimum.

For future research, we will scale up our model to handle more difficult tasks, focusing on the local minimum problem. The framework could be further extended with other predictive metrics in future, rather than only the PNSS metric. We will also investigate the mechanism for efficient parallel training, as achieving stable parallel training remains a challenge in mapless navigation tasks that lack a task decomposition scheme [33]. In addition, model-based learning will be considered in the future to allow efficient sampling of data for training the agent [55]. Moreover, real-life environments are mostly dynamic. We will focus on addressing the problem of RL-based mapless navigation in dynamic environments [56]. Ultimately, we will aim to deploy this work on real robots and evaluate the performance in real-world environments, which present many challenges in ensuring safety and filling the gap between the real world and simulation.

ACKNOWLEDGEMENT

We are grateful to Dr Rob Deaves from Dyson with whom we have discussed some of the ideas in this work. Yan Gao thanks the China 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 Dr Ze Ji (Grant No. IF2223-199).

References

[1] B. Patle, A. Pandey, D. Parhi, A Jagadeesh, *et al.*, "A review: On path planning strategies for navigation of mobile robot", *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[2] H. Kurniawati, "Partially observable markov decision processes and robotics", *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 253–277, 2022.

[3] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey", *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[4] A. H. Qureshi and M. C. Yip, "Deeply informed neural sampling for robot motion planning", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 6582–6588.

[5] 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.

[6] L. Xie, "Reinforcement learning based mapless robot navigation", Ph.D. dissertation, University of Oxford, 2019.

[7] 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.

[8] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning", *arXiv preprint arXiv:1804.00456*, 2018.

[9] M. Dobrevski and D. Skočaj, "Deep reinforcement learning for map-less goal-driven robot navigation", *International Journal of Advanced Robotic Systems*, vol. 18, no. 1, p. 1 729 881 421 992 621, 2021.

[10] 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.

[11] W. Ding, S. Li, H. Qian, and Y. Chen, "Hierarchical reinforcement learning framework towards multi-agent navigation", in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2018, pp. 237–242.

[12] 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.

[13] X. Zhou, T. Bai, Y. Gao, and Y. Han, "Vision-based robot navigation through combining unsupervised learning and hierarchical reinforcement learning", *Sensors*, vol. 19, no. 7, p. 1576, 2019.

[14] 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.

[15] 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.

[16] 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.

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

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

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

[20] 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.

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

[22] C. Wang, W. Chi, Y. Sun, and M. Q.-H. Meng, "Autonomous robotic exploration by incremental road map construction", *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1720–1731, 2019.

[23] W. Zhao, R. Lin, S. Dong, and Y. Cheng, "A study of the global topological map construction algorithm based on grid map representation for multirobot", *IEEE Transactions on Automation Science and Engineering*, 2022.

[24] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning", in *2011 second international conference on mechanic automation and control engineering*, IEEE, 2011, pp. 1067–1069.

[25] X. Zhou, X. Yu, Y. Zhang, Y. Luo, and X. Peng, "Trajectory planning and tracking strategy applied to an unmanned ground vehicle in the presence of obstacles", *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1575–1589, 2020.

[26] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators", in *International conference on machine learning*, PMLR, 2015, pp. 1312–1320.

[27] M. Andrychowicz, F. Wolski, A. Ray, *et al.*, "Hindsight experience replay", *Advances in neural information processing systems*, vol. 30, 2017.

[28] S. Nair, S. Savarese, and C. Finn, "Goal-aware prediction: Learning to model what matters", in *International Conference on Machine Learning*, PMLR, 2020, pp. 7207–7219.

[29] M. Zhu, M. Liu, J. Shen, *et al.*, "Mapgo: Model-assisted policy optimization for goal-oriented tasks", *arXiv preprint arXiv:2105.06350*, 2021.

[30] H. Chan, Y. Wu, J. Kiros, S. Fidler, and J. Ba, "Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning", *arXiv preprint arXiv:1902.04546*, 2019.

[31] A. Kumar, X. B. Peng, and S. Levine, "Reward-conditioned policies", *arXiv preprint arXiv:1912.13465*, 2019.

[32] R. K. Srivastava, P. Shyam, F. Mutz, W. Jaśkowski, and J. Schmidhuber, "Training agents using upside-down reinforcement learning", *arXiv preprint arXiv:1912.02877*, 2019.

[33] X. Yang, Z. Ji, J. Wu, *et al.*, "Hierarchical reinforcement learning with universal policies for multistep robotic manipulation", *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

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

[35] A. S. Vezhnevets, S. Osindero, T. Schaul, *et al.*, "Feudal networks for hierarchical reinforcement learning", in *International Conference on Machine Learning*, PMLR, 2017, pp. 3540–3549.

[36] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning", in *Conference on Robot Learning*, PMLR, 2020, pp. 1025–1037.

[37] S. Nair and C. Finn, "Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation", *arXiv preprint arXiv:1909.05829*, 2019.

[38] T. Jurgenson, O. Avner, E. Groshev, and A. Tamar, "Sub-goal trees a framework for goal-based reinforcement learning", in *International Conference on Machine Learning*, PMLR, 2020, pp. 5020–5030.

[39] G. Parascandolo, L. Buesing, J. Merel, *et al.*, "Divide-and-conquer monte carlo tree search for goal-directed planning", *arXiv preprint arXiv:2004.11410*, 2020.

[40] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, “Dynamics-aware unsupervised discovery of skills”, *arXiv preprint arXiv:1907.01657*, 2019.

[41] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, “Search on the replay buffer: Bridging planning and reinforcement learning”, *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[42] Y. Zhu, Z. Wang, C. Chen, and D. Dong, “Rule-based reinforcement learning for efficient robot navigation with space reduction”, *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 2, pp. 846–857, 2021.

[43] E. Wijmans, A. Kadian, A. Morcos, *et al.*, “Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames”, *arXiv preprint arXiv:1911.00357*, 2019.

[44] M. Rosano, A. Furnari, L. Gulino, and G. M. Farinella, “On embodied visual navigation in real environments through habitat”, in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021, pp. 9740–9747.

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

[46] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, “Continuous control with deep reinforcement learning”, in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1509.02971.

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

[48] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman, “Occupancy anticipation for efficient exploration and navigation”, in *European Conference on Computer Vision*, Springer, 2020, pp. 400–418.

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

[50] Abhishek Kadian*, Joanne Truong*, A. Gokaslan, *et al.*, “Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?”, 4, vol. 5, 2020, pp. 6670–6677.

[51] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, “Gibson env: Real-world perception for embodied agents”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079.

[52] 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.

[53] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods”, in *International conference on machine learning*, PMLR, 2018, pp. 1587–1596.

[54] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”, in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[55] L. Kaiser, M. Babaeizadeh, P. Milos, *et al.*, “Model-based reinforcement learning for atari”, *arXiv preprint arXiv:1903.00374*, 2019.

[56] Z. Wang, C. Chen, H.-X. Li, D. Dong, and T.-J. Tarn, “Incremental reinforcement learning with prioritized sweeping for dynamic environments”, *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 2, pp. 621–632, 2019.