IET Cyber-Systems and Robotics

WILEY

ZHEJIANG UNIVERSITY PRESS | IET The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

Deep Reinforcement Learning for Localisability-Aware Mapless Navigation

Yan Gao[1] | Jing Wu[2] | Changyun Wei[3] | Raphael Grech[4] | Ze Ji[1]

[1]School of Engineering, Cardiff University, Cardiff, UK | [2]School of Computer Science and Informatics, Cardiff University, Cardiff, UK | [3]College of Mechanical and Electrical Engineering, Hohai University, Changzhou, China | [4]Spirent Communications, Paignton, UK

Correspondence: Ze Ji (jiz1@cardiff.ac.uk)

Received: 12 September 2024 | **Revised:** 11 March 2025 | **Accepted:** 2 April 2025

Funding: The authors received no specific funding for this work.

Keywords: motion planning | navigation | reinforcement learning | simultaneous localisation and mapping (SLAM)

ABSTRACT

Mapless navigation refers to the task of searching for a collision free path without relying on a pre-defined map. Most current works of mapless navigation assume accurate ground-truth localisation is available. However, this is not true, especially for indoor environments, where simultaneous localisation and mapping (SLAM) is needed for location estimation, which highly relies on the richness of environment features. In this work, we propose a novel deep reinforcement learning (DRL) based mapless navigation method without relying on the assumption of the availability of localisation. Our method utilises RGB-D based Oriented FAST and Rotated BRIEF (ORB) SLAM2 for robot localisation. Our policy effectively guides the robot's movement towards the target while enhancing robot pose estimation by considering the quality of the observed features along the selected paths. To facilitate policy training, we propose a compact state representation based on the spatial distributions of map points, which enhances the robot's awareness of areas with reliable map points. Furthermore, we suggest incorporating the relative pose error into the reward function. In this way, the policy will be more responsive to each single action. In addition, rather than utilising a pre-set threshold, we adopt a dynamic threshold to improve the policy's adaptability to variations in SLAM performance across different environments. The experiments in localisation challenging environments have demonstrated the remarkable performance of our proposed method. It outperforms the related DRL based methods in terms of success rate.

1 | Introduction

Mapless navigation refers to the task of searching for a collision-free path without relying on a pre-defined map. Consequently, mapless navigation holds significant importance in diverse applications within unstructured environments, where acquiring detailed and precise maps in advance poses a considerable challenge.

Conventional path-planning algorithms have been extensively studied and applied in numerous scenarios [1]. Nevertheless, these approaches typically demand the design of heuristic functions crafted manually, displaying limited generalisation capability. Hence, considerable customisation is required when adapting these methods to different environments [2, 3].

In recent years, deep reinforcement learning (DRL) has demonstrated promising performance across various research domains [4], including mapless navigation [5–8]. Reinforcement learning (RL) is inherently robust, as the agent learns from its own experiences during the training process. RL-based mapless navigation methods usually construct a direct mapping between robot states and actions. In most works, the RL agent state comprises two primary components: the sensory observations and the position information of the target location, which is usually represented in terms of polar coordinates relative to the robot's frame [6, 7]. To obtain the accurate polar coordinates of the target, reliable robot localisation is necessary.

Such studies usually assume that robots have access to their ground-truth poses [6, 7] and focus on path planning only. However, the performance of localisation is highly dependent on the environment. Techniques of visual simultaneous localisation and mapping (SLAM) highly rely on the tracked visual features for global navigation satellite system (GNSS)-denied environments; while, for outdoor, GNSS-based localisation is prone to multipath reflections of radio signals caused by the environment. The ignorance of localisation performance would lead to robots failing to localise themselves and possible catastrophic issues caused by decision-making based on unreliable state estimation. We believe that the perception module for localisation should be closely coupled with decision making in the process of path planning, which should consider not only commonly used criteria such as path length, but also the richness of features in the surroundings that could greatly affect the localisation performance.

However, even state-of-the-art ORB-SLAM2 [9] often suffers from localisation failures. Its performance may be degraded in situations where environmental features are poorly observed, such as when navigating in areas lacking distinct features. In addition, localisation failures could occur even in feature-rich scenarios, that is, when encountering rapid rotations or movement [10]. Therefore, avoiding SLAM failures in navigation is necessary [10–13].

However, ensuring the effectiveness of SLAM in navigation is nontrivial and needs several considerations. Firstly, some studies train RL models using observed or manually extracted image features to evaluate discrete actions and predict SLAM failures [10, 11]. Although image features contain rich information, they often generalise poorly to unseen environments. Secondly, some works incorporate SLAM failure prevention into the RL reward function by penalising absolute trajectory error (ATE), which measures the disparity between SLAM-estimated and ground-truth poses [14]. However, since pose estimation error accumulates over past actions, penalties may be delayed. Thirdly, while discrete action spaces are commonly used [10–14], continuous action spaces offer greater flexibility for navigating complex environments.

With these considerations, we propose a novel DRL-based mapless navigation approach with a continuous action space (Figure 1) shows the overall framework of our work.. Rather than assuming the availability of localisation, our method uses ORB-SLAM2 [9] with an RGB-D camera for localisation in indoor environments. Our method aims to generate an effective policy for the control of the robot's movement towards the designated target, with the capabilities of obstacle avoidance and enhanced robot pose estimation by considering the quality of the observed features along the selected paths, hence consequently leading to optimal SLAM performance. Instead of directly using raw sensory observations as the state input, we adopt a more impact representation. Many works have proved that enough overlap with already mapped points can prevent SLAM failure [10, 11]. On the other hand, it is more desirable to have more ORB features that can be detected by the robot for better localisation. In addition, a more uniform distribution of features in space can enhance the alignment of keyframes, thus improving the localisation accuracy [15]. We categorise map points into two classes: the currently tracked and those previously tracked and stored in the map. Then, we extract information about the distribution of these map points, making it become part of the state input. It may enable the agent to discern areas with reliable map points. Furthermore, we introduce a novel reward function that incorporates relative pose error (RPE) as a penalising factor. RPE can offer a more intuitive representation of the impact of SLAM resulting from each action. In addition, SLAM performance varies across different environments [16]. Therefore, instead of employing a pre-set threshold and penalising the robot when the RPE surpasses the threshold, we adopt a dynamic threshold, wherein the threshold is influenced by the prevailing environmental conditions.

Our main contributions are summarised below:

- We develop a novel DRL-based mapless navigation framework that uses RGB-D-based ORB-SLAM2 for localisation, eliminating the assumption of available localisation as used previously.
- We propose a compact state representation constructed based on the spatial distributions of map points, which allows the robot to be aware of distributions of tracked features, hence encouraging the robot to navigate along localisation-reliable paths.
- RPE is incorporated in our proposed reward function, rather than ATE. This allows the policy to be more responsive to each single action, hence avoiding delayed penalties.
- Our approach is extensively evaluated in photorealistic simulations in various localisation-challenging environments with poor visual features.

2 | Related Work

Conventional map-based navigation approaches face two primary challenges: (1) the labour-intensive process of constructing and updating maps; and (2) heavy reliance on accurate sensors for both mapping activities and local cost-map predictions [7, 17]. Mapless navigation is generally regarded as a solution to alleviate the navigation system from the requirement of a pre-built map. It is often accomplished by directly modelling the correspondence between sensory observations and robot actions [17]. Recently, RL has been proven to be suited for mapless navigation because of its powerful representation learning capabilities [5–7, 17]. RL brings about the possibility of learning control policies directly from raw sensory inputs [18, 19]. For example, Tai et al. [7] conduct training on an end-to-end mapless navigation model, utilising sparse laser range readings, along with the robot's velocity and the relative position of the target, as the inputs for RL. The reward function is based on the Euclidean distance from the robot to the target with

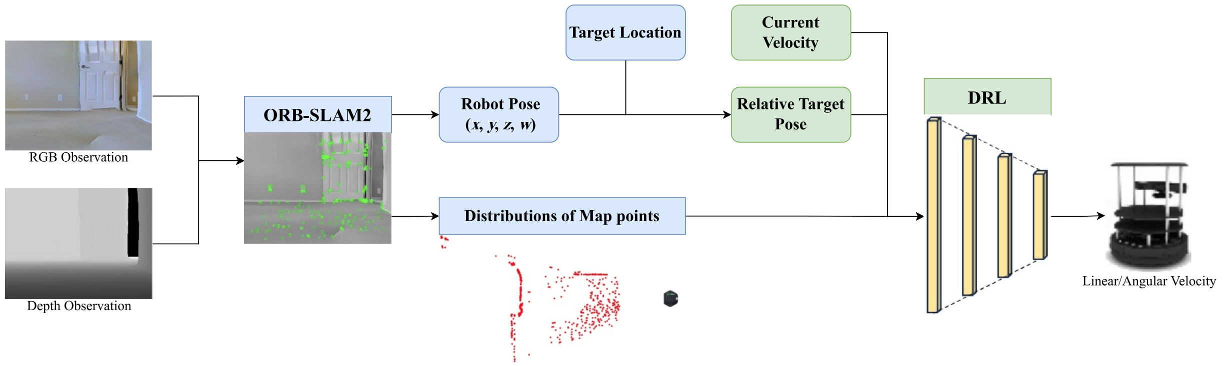

FIGURE 1 | The overall framework. ORB-SLAM2 determines the robot pose through the current RGB-D observations and subsequently acquires the polar coordinates of the target location. Additionally, our proposed method captures information on the distributions of map points derived from SLAM. The input for the DRL policy comprises the target location, the distribution of map points, and the current velocity of the robot. The output consists of linear and angular velocity commands.

other terms such as penalties on collisions. These approaches assume that the agent's pose can be obtained and accurate through various external localisation solutions. However, this assumption is unrealistic, particularly in indoor environments where GNSS-based localisation is unavailable. In such instances, SLAM-based localisation is required. However, its performance is influenced by various factors, leading to failures in specific scenarios with poor feature observations, such as textureless scenes [20], the presence of shadows [21], and situations that involve rapid rotations around corners [11] or saturated images caused by improper illumination/exposures [20].

Several works attempt to prevent SLAM failures through RL-based solutions. For example, Naveed et al. [10] propose a novel DRL network to assess the safety of upcoming navigation steps using only a single image as input. Similarly, Prasad et al. [11] introduce a Q-learning-based approach, aiming to mitigate tracking failures in visual SLAM. They use manually crafted features, specifically the count of tracked features, as inputs to their Q network. In our preliminary experiments, we have observed that these approaches struggle to perform well across challenging environments. This could be due to the presence of high-dimensional redundant data or unrepresentative features, which hinder the effectiveness of the state. Moreover, these methods solely focus on determining if the action may lead to SLAM localisation failures, without incorporating considering navigation-related tasks. Mostly, the pre-built path planning is deployed in those works.

Several RL-based mapless navigation approaches explore the utilisation of SLAM for localisation. For instance, Lin et al. [13] enable the robot to acquire a mapless navigation policy, aiming to avoid localisation failures. They introduce a reward metric to penalise behaviours that may result in localisation failures, along with a reconfigured state representation that incorporates current observations and historical trajectory information. Chen et al. [14] propose a novel approach for enhancing localisation in RL-based navigation within dynamic human environments. They mainly introduce a reward metric capable of providing feedback on behaviours influencing localisation accuracy. Specifically, they impose a penalty on the agent based on the disparity between the robot pose returned by SLAM and the ground-truth. However, this approach may not be reasonable, as the pose estimation error is not only the result of the last executed action. Instead, the error is the accumulation of errors from the past actions executed.

Furthermore, it is notable that previous works [10, 11, 13, 14] utilise discrete action space. In complex navigation tasks, opting for a continuous action space would be more desirable.

In this work, we introduce a novel mapless navigation method with a continuous action space. Our method deviates from the reliance on external localisation methods. The robot pose is provided by RGB-D-based ORB-SLAM2 [9]. The policy is trained to optimise the effectiveness of SLAM-based localisation by encouraging the robot to traverse informative paths. A compact state representation, which incorporates spatial distributions of map points, is introduced for effective policy training.

3 | Methods

This section introduces our proposed method, starting with the preliminaries of ORB-SLAM2 and RL, followed by the detailed RL problem formulation, focusing on the design of state and reward representations.

3.1 | RGB-D-Based ORB-SLAM2

ORB-SLAM2 contains three main components, including (1) the tracking module that focuses on localising the camera with each frame by identifying matched features to the local map and minimising the reprojection error using motion-only bundle adjustment (BA); (2) the local mapping manager that stores the local map, performing optimisation through local BA; and (3) the loop closing component that is responsible for detecting loops and correcting accumulated drifts through pose-graph optimisation.

As discussed earlier, this work is more concerned with how to encourage the robot to traverse areas of reliable feature tracking, allowing robust localisation through reprojection optimisation.

The reprojection error of map points to corresponding matched key points is represented as follows:

$$e_{i,j} = x_{i,j} - \pi_{\mathrm{i}}\left(T_{iw}, X_{w,j}\right) \tag{1}$$

where $e_{i,j}$ is the reprojection error of a map point j in a keyframe i, $x_{i,j}$ represents the matched keypoint, $T_{iw} \in \mathrm{SE}(3)$ is the pose of keyframe i, which is a rigid body transformation that transforms points from the world w to the camera coordinate system, $X_{w,j} \in \mathbb{R}^3$ represents the 3D pose of map point j in the world coordinate system w, and π_{i} denotes the projection function.

Then, to obtain the camera pose, a cost function C needs to be minimised:

$$C = \sum_{i,j}\left(\rho_{\mathrm{h}}\left(e_{i,j}^{\mathrm{T}}\omega_{i,j}^{-1}e_{i,j}\right)\right) \tag{2}$$

where ρ_{h} represents the Huber robust cost function, and $\omega_{i,j}$ denotes the covariance matrix linked to the scale at which the keypoint was identified.

3.2 | Reinforcement Learning

3.2.1 | Markov Decision Process

The mapless navigation problem can be considered as a Markov decision process (MDP), which can be represented by a tuple $\langle S, A, R, p, \gamma, \rho_0\rangle$. S and A represent the state and action spaces, respectively. $R(s,a)$ specifies the reward function. $p(s'|s,a)$ represents the system transition model. γ is the discount factor and ρ_0 is the initial state distribution. A policy $\pi(a|s)$ is characterised as a mapping of a state to an action. A state value function $V^{\pi}(s)$ represents the expected value of the cumulative rewards, following policy π from state s. Mathematically, it is expressed as $V^{\pi}(s) = \mathrm{E}_{a\sim\pi,s\sim p}\left[\sum_{t=0}^{T}\gamma^t R(s_t,a_t)\right]$, where T is the terminal timestep and $t \in [0,T]$. A state-action value function, denoted as $Q^{\pi}(s,a)$, encapsulates the analogous quantity when an action a is executed at state s. The objective of RL is to find an optimal policy that maximises the value function [22].

3.2.2 | Deep Deterministic Policy Gradient

We use the deep deterministic policy gradient (DDPG) [23] RL algorithm to train our policies. DDPG falls into the category of actor-critic methods, representing a hybrid approach that combines value and policy-based methods. This actor-critic architecture makes it suitable to work with a continuous action space. It consists of two networks, an actor network $\pi(s_t|\theta^{\pi})$ and a critic network $Q(s_t, a_t|\theta^{Q})$, deterministically estimating the action a_t given s_t and the Q-value of them respectively. The actor and critic networks are parametrised by θ^{π} and θ^{Q}, respectively. DDPG is affected by an overestimation of Q-values and, to handle the problem, we can use a target network and a second critic network [23].

3.3 | State Representation

Our method is based solely on an RGB-D camera. A compact and informative state representation is essential for effective and efficient training. Directly using raw RGB-D observations is impractical due to their high redundancy and dimensionality. Our proposed representation O_t comprises four components denoted by O_{dis}, O_{tgt}, O_{dep}, O_{vel}.

O_{dis} contains essential information about the spatial distribution of tracked map points that are crucial for the error reprojection optimisation process, formulated in Equation (1). Previous studies show that tracking sufficient features or map points between consecutive frames is essential for reliable SLAM performance [10–12]. To ensure consistent map points being tracked, the robot would directly benefit from the awareness of the spatial distribution of the tracked map points. In our work, we partition the map points into two categories: (1) those that are currently tracked, denoted as T; and (2) those that are previously tracked and stored in the map, denoted as M.

The T distribution enables the robot to assess the current view's feature distribution for decision-making. As an example (Figure 2), if T's distribution is predominantly concentrated on the right side of the robot, opting for a left turn might be deemed as a less favourable action, as the robot will lose track of these features. M provides contextual information of the map points of the surroundings. Such contextual information endows the robot with the predictive capability of future observations. With the same example above, if a significant quantity of M is located on the left side of the robot, but untracked currently. Executing a left rotation may not result in SLAM failure due to the pre-tracked features.

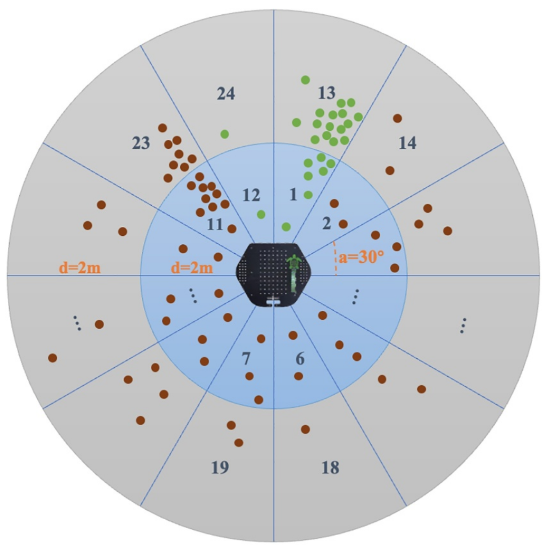

FIGURE 2 | Illustration of spatial distribution representation of map points. The green points represent the currently tracked map points (distributed within regions 1, 12, 13 and 24). The brown points represent those previously tracked and may be distributed across any region.

To effectively encapsulate the map point spatial distribution in a structured format, we partition the surrounding area into 24 regions in the polar coordinate frame centred on the robot, as illustrated in Figure 2. Each map point is assigned to a corresponding region based on its location with respect to the robot's frame. Given that the horizontal angle of the RGB-D camera (Xtion PRO LIVE, ASUS Taipei, China) is 58°, points in T are primarily distributed in regions 1, 12, 13 and 24. However, the points in M can be in all regions (1 − 24). Subsequently, we extract the distribution information of T and M map points in each region, represented as follows:

$$\eta(p) = [N, \sigma_0, \sigma_1, \sigma_2] \tag{3}$$

where $\eta(\cdot)$ denotes the distribution information, and p represents the map points of interest. N is the number of p, and σ denotes the variances of the map point coordinates in three axes, respectively. The reason for including the variance is that they represent the distribution of p across three axes. A large variance means the data range is large on the corresponding axis; while a smaller value means the data are concentrated. Therefore, O_{dis} can be formulated as follows:

$$\begin{aligned} O_{\text{dis}} = \ & [\eta(T_1), \eta(T_{12}), \eta(T_{13}), \eta(T_{24}) \\ & \eta(M_1), \eta(M_2), \ldots, \eta(M_{24})] \end{aligned} \tag{4}$$

where $T_{1,12,13,24}$ represents the T map points in region 1, 12, 13 and 24 respectively. Similarly, M_{1-24} is the M map points in region 1–24.

O_{tgt}, O_{dep}, and O_{vel} are used to generate the policy that controls the robot's move to the target location and avoid collisions. O_{tgt} is the relative position of the target, represented in the polar coordinates with respect to the robot's frame ($O_{\text{tgt}} = [d, \theta]$). The pose is obtained from ORB-SLAM2 and is converted to the polar coordinates.

Regarding obstacle avoidance, many works show that the task can be effectively achieved using only range sensors [5–7]. The RGB-D camera, capable of extracting depth information, can be considered as a range sensor. Since the depth image $\boldsymbol{D}$ is represented as an $m \times n$ matrix, to be effective in training the policy, rather than using the raw depth data, we converted the matrix into a 1D vector $\boldsymbol{k}$, with the dimension of $1 \times n/p$, simulating a 2D range sensor.

$$\boldsymbol{k} = \begin{bmatrix} \min(D[a:b, 0:(p-1)]) \\ \min(D[a:b, p:(2p-1)]) \\ \min(D[a:b, 2p:(3p-1)]) \\ \vdots \\ \min(D[a:b, (((n/p)-1)p):((n/p)p-1)]) \end{bmatrix}^{\mathrm{T}} \tag{5}$$

where min returns the lowest number of an array, and $D[a:b, \cdots]$ returns the sub-matrix, ranged from index a to b. Here, $a = m/2$ and $b = 3m/4$.

Last, O_{vel} denotes the robot velocity, including both the linear and angular velocities, which are essential in guiding the policy to decide its subsequent steps.

Therefore, the state representation is formulated as follows:

$$O_t = \{O_{\text{dis}} || O_{\text{tgt}} || O_{\text{dep}} || O_{\text{vel}}\} \tag{6}$$

where || denotes vector concatenation.

3.4 | Reward Function and Network Structure

In our work, in addition to typical mapless navigation rewards, such as rewarding the agent on its arrival at the target or penalising actions of collision, we also included two other localisation-related rewards. The agent will be penalised when the estimated pose is diverging from the true trajectory, or when SLAM fails to track its trajectory.

$$R = \begin{cases} r_{\text{arrive}} & \text{if } d_{\text{s}} \le \delta_{\text{s}} \text{ and } d_{\text{t}} \le \delta_{\text{d}} \\ r_{\text{collision}} & \text{if collision} \\ r_{\text{pose}} & \text{if } e_t \ge \delta_{\text{p}} \\ r_{\text{lost}} & \text{if SLAM fails} \\ r_{\text{approach}} & \text{otherwise} \end{cases} \tag{7}$$

A positive reward, r_{arrive}, is assigned if the robot reaches the target. A two-threshold criterion is introduced. d_{s} and d_{t} represent the distances between the robot and the target, where d_{s} is estimated using SLAM, and d_{t} is the ground-truth position. δ_{s} and δ_{d} specify the corresponding distance thresholds. Allowing for a certain amount of error during ORB-SLAM2 runs, δ_{d} is slightly larger than δ_{s}. If the robot collides with an obstacle, a negative reward, $r_{\text{collision}}$, is assigned.

The reward r_{pose} is introduced to reward or penalise the agent based on the impact on the SLAM performance. RPE, which measures the drift of pose estimates from the ground-truth over a fixed time span, is used to evaluate the performance of SLAM. RPE offers immediate feedback by measuring localisation errors over a fixed number of frames, directly capturing the impact of actions on SLAM accuracy. This real-time reward mitigates delayed penalties, enabling efficient policy adjustments. Moreover, RPE is less affected by cumulative noise, leading to more stable training. By incorporating RPE into the reward function, the model can learn a more effective navigation policy while avoiding the drawbacks of delayed rewards. Assuming that the ground-truth relative poses between $Q_{i-\Delta t}$ and Q_i is $Q_{i-\Delta t}^{-1} Q_i$ and the estimated relative poses between $P_{i-\Delta t}$ and P_i is $P_{i-\Delta t}^{-1} P_i$. The RPE is formulated below:

$$e_i = |\text{trans}\left(\left(Q_{i-\Delta t}^{-1} Q_i\right)^{-1} \left(P_{i-\Delta t}^{-1} P_i\right)\right)| \tag{8}$$

where trans denotes the translation error, which is considered here.

RPE is employed as the representation of the error induced by the corresponding action. In our work, we consider the RPE error in one timestep, $\Delta t = 1$. When RPE surpasses a predefined threshold, it indicates that the last action has led to inaccurate pose estimation, and will receive a penalty. However, determining a reliable threshold is challenging, due to the substantial variation across different environments. Instead of setting a fixed threshold, we employ a dynamic threshold approach, by checking if the current RPE lies within an acceptable range. Specifically, we calculate the mean value and standard deviation of the RPEs of the last τ timesteps to provide an estimated range of RPE distribution of the environment. Then, the threshold, δ_{p}, is defined as follows:

$$\delta_{\mathrm{p}} = \mu(E) + \alpha\, \sigma(E), \tag{9a}$$

$$E = (e_{t-\tau}, e_{t-(\tau-1)}, \ldots, e_{t-1}) \tag{9b}$$

where α is a constant, $\mu(\cdot)$ represents the mean value and $\sigma(\cdot)$ denotes the standard deviation.

If e_t exceeds the threshold δ_{p}, this is considered as surpassing the acceptable range. This approach endows the agent with the capability to adapt to diverse environments by providing the agent with an adaptive estimation of localisation performance in the current environment. When the RPE exceeds δ_{p}, a negative reward r_{pose} is applied. The calculation of δ_{p} involves only basic statistical operations (mean and standard deviation), which are negligible in terms of computational cost. Given that these statistics are updated incrementally and used for threshold adjustment rather than real-time SLAM correction, they do not introduce any noticeable slowdown to the system. In addition, if SLAM encounters a complete failure, that is, the number of currently tracked features is 0, r_{lost} is applied to the agent.

The reward r_{approach} is defined as the distance difference from the target to the robot between the last timestep and the current timestep, $r_{\mathrm{approach}} = c_{\mathrm{r}}(d_{t-1} - d_t)$, where c_{r} is a hyperparameter and d_t is the distance between the robot and the target at timestep t. This encourages the robot to move towards the goal.

Regarding the network structure, as mentioned, a DDPG-based method is deployed in our work. The actor network in the DDPG framework consists of three multi-layer perceptron (MLP) layers, each with a uniform size of 512. The critic network also comprises three MLP layers. The size of the first and last layers is 512, and the dimension of the second layer is 514, incorporating two additional dimensions for the action. ReLU activation is applied to each layer in both the actor and the critic networks, except for the output layers. The actor network utilises hyperbolic tangent activation for the last layer, whereas the critic network has no activation applied to the output.

The design of our neural network structure in the DDPG framework is based on both theoretical principles and empirical findings in DRL. MLPs are chosen for their universal approximation capability, ensuring effective policy and value function learning. The three-layer architecture provides sufficient capacity, with 512 neurons per layer in both networks, while the critic's second layer has 514 dimensions to incorporate action information. ReLU activation is applied to hidden layers to mitigate vanishing gradients and enhance training efficiency. The actor network employs a tanh activation in the output layer to constrain actions within a bounded range, ensuring stable exploration, whereas the critic network uses a linear output to allow unrestricted Q-value estimation. These design choices align with established RL practices, optimising learning stability and performance in continuous control tasks.

4 | Experiments

4.1 | Training in Simulation

The simulation environment, iGibson, is used in our work [24], providing a range of photorealistic 3D indoor environments. We use a TurtleBot, which has a differential drive configuration. The training procedure of our model is performed in a localisation-challenging environment, such as Env 1 in Figure 3.

The rewards are set as follows: $r_{\mathrm{arrive}} = 20$, $\delta_{\mathrm{s}} = 0.5$, $\delta_{\mathrm{d}} = 0.7$, $r_{\mathrm{collision}} = -5$, $r_{\mathrm{lost}} = -5$. For Equation (9), α is 3, τ is 50. c_{r} is set as 100 in r_{approach}. These parameters are empirically determined through trial and error.

Regarding α in Equation (9), it determines how much tolerance we allow beyond the typical error range before applying penalties. The choice of α is determined empirically through multiple preliminary experiments, where we test different values ($\alpha \in [1, 5]$) and observe that $\alpha = 3$ provides the most stable and effective performance. For $\alpha \leq 2$, δ_{p} becomes too small, leading to frequent and often unnecessary penalties, even when the SLAM system is performing well within expected bounds. This caused instability in reinforcement learning since the agent was excessively penalised, reducing its ability to learn an optimal policy. For $\alpha \geq 4$, δ_{p} becomes too large, allowing SLAM errors to persist without triggering penalties. This led to slower convergence and less robust performance in the final trained policy. Our empirical findings indicate that $\alpha = 3$ provides a reasonable trade-off between stability and responsiveness in SLAM error control, and this choice is effective in maintaining system performance.

The action space is continuous. The linear velocity is set to be positive, ranging from 0 to 0.06 m/s, whereas the angular velocity varies within the range of -0.1 to 0.1 rad/s.

We train our policy for 5 million steps. In each episode, the start location is randomised and the target is also randomly sampled in the environment, with the condition that the target must be 1–5 m away from the start location. The episode terminates when any of the following conditions are met: (1) SLAM fails, that is, insufficient valid map points can be effectively tracked; (2) The error in either the estimated x-coordinates or y-coordinates from the ground-truth exceeds 0.15, or the error in the estimated orientation is greater than 0.5; (3) The robot

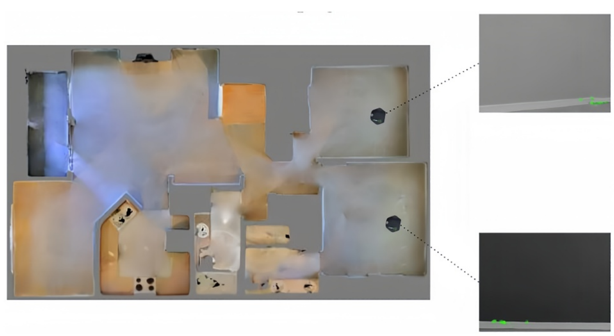

FIGURE 3 | Env 1: Aloha and two examples of the robot visual features.

collides or surpasses the time limit (3000 timesteps); and (4) The robot successfully reaches the target location. Successful arrival at the goal is determined by the following two conditions: (1) the distance from the estimated position to the target is less than 0.5 m, and (2) the distance from the ground-truth position to the goal is less than 0.7 m. The work was performed on a workstation with an Intel i9-10900X CPU (3.7 GHz × 20) and an NVidia RTX-2080 TI GPU, which takes about 145 h. Our trained policy requires the computational time of 0.07 s to compute the velocity action.

4.2 | Evaluation

4.2.1 | Comparison of Layout Configurations for Model Performance

In this section, we explore the impact of different layouts of distributing regions around the robot in the model input on its performance. In addition to the 24 regions we proposed earlier, we designed two additional configurations with 16 and 36 regions (as shown in Figure 4). After training the model using these three configurations, we test each method over 100 episodes. The success rates for the three methods are shown in Table 1.

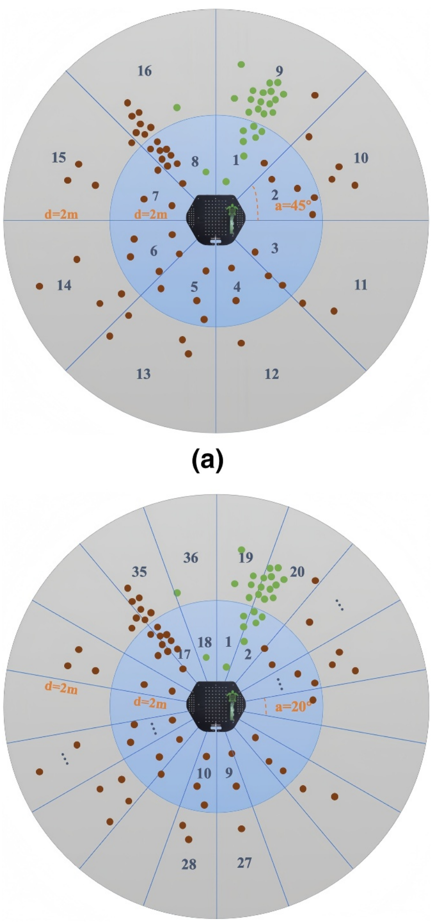

FIGURE 4 | Two different layout configurations with (a) 16 regions and (b) 36 regions.

As shown in the table, the method with 24 regions exhibits the highest success rate. In contrast, the method utilising 36 regions achieves the lowest success rate of 23%. An increased number of regions leads to higher computational complexity. Specifically, as the number of regions increases, the dimensionality of the model inputs also rises. We represent the number and distribution of map points in each region using a 4-dimensional vector, requiring a total of 144 dimensions for the map points that are previously tracked and stored in the map. For the map points currently being tracked, there are 8 regions that need to be processed, corresponding to 32 dimensions. In total, this results in 176 dimensions of data, significantly increasing computational overhead. Moreover, excessively high input dimensions may introduce redundant information, complicating the training process.

On the other hand, using fewer regions reduces the input dimensionality but may lead to information loss. When the number of regions is small, each region encompasses a larger spatial area, leading to a reduced density of map points. This reduction in density adversely affects the model's ability to capture local features and impedes the robot's capacity to accurately perceive environmental changes in certain scenarios. Given that the method with 24 regions performs better, this section will focus on this method in the subsequent experiments.

4.2.2 | Baseline

To validate the effectiveness of our method, we select several approaches as the baselines.

- Naive policy (NP-GT). We train a simple mapless navigation model, which uses the ground-truth as the robot pose, obtained from the simulation environment. Since ground-truth poses are used, this policy does not need to consider the accuracy of the poses. Therefore, the input representation contains O_{tgt}, O_{dep}, and O_{vel}. We use the same reward function proposed in ref. [7].
- Naive policy (NP-SLAM). In contrast to the NP-GT, the robot poses are supplied by ORB-SLAM2. The rest remains the same as the method above.
- Policy with image input (P-Image). In ref. [10], raw image observation is employed as the RL input and subsequently identifies which action may lead to SLAM failures. Inspired by this approach, we utilise the same neural network structure to extract image features as inputs for our method. For a fair comparison, the reward function is the same as ours.

TABLE 1 | Test success rates of different configurations in Env 1.

Configuration	Success rate
16 regions	31%
24 regions	49%
36 regions	23%

- Policy trained by ATE-based reward function (P-ATE). In ref. [14], a reward metric is proposed to provide feedback on actions that affect localisation accuracy. The difference between the ATE errors at the previous and the current timesteps is utilised for the reward. Additionally, the agent incurs a substantial penalty if the current error surpasses a pre-defined threshold. For comparison, we train a model using its reward function, but the state representation is the same as ours for a fair comparison.

4.2.3 | Results

In the process of training, we test each method every 100,000 training steps, with 100 episodes per test. The success rates are shown in Figure 5. NP-GT achieves the highest success rate, eventually stabilised at around 65%. The success rate of NP-SLAM is significantly lower and shows a declining trend after 4 million training steps, reaching about 30%. In contrast, our approach demonstrates a notably higher success rate, approximately 20% higher than that of NP-SLAM. The success rates of P-ATE and P-Image do not increase with the number of training steps, and ended up at about 5% and 3%, respectively.

After training, we conducted a performance evaluation on our method and the baselines in Env 1. For each method, 100 tests were performed. The initial position of the robot and the target are generated randomly for each test. The same start and goal positions are used for different algorithm configurations to ensure fair comparisons. The success rates of the testing tasks are shown in Table 2.

It is obvious that the highest success rate of 61% is achieved when the robot pose is precise. In contrast, when relying on SLAM for pose estimation, the success rate exhibits a considerable decrease, falling to 33%. The decrease in the test success rate of NP-SLAM is related to the increased complexity of the scenarios during training, as well as the limitations of the input representation. In the initial stages of training, the robot operated in relatively simple environments where direct paths to the goal were sufficient. However, as training progressed, the robot encountered more complex scenarios, such as navigating to goals located in different rooms, necessitating advanced path planning. Our model employs polar coordinates to represent the target location, but this representation fails to capture the spatial relationships in such complex environments. As a result, the robot may select more complicated paths, increasing the likelihood of SLAM failures and reducing the success rate. This suggests that the robot's movements, generated by NP-SLAM, fail to ensure accurate SLAM localisation, consequently, resulting in navigation failure. Our proposed method has the highest success rate among the related baselines, surpassing the NP-SLAM by 16%. Our policy demonstrates the capability to enhance the robot pose estimation. In addition, we record the robot's localisation errors at the end of episodes that did not end due to collisions. The errors were calculated using the Euclidean distance between the SLAM-predicted pose and the ground-truth pose, which can be formulated as follows:

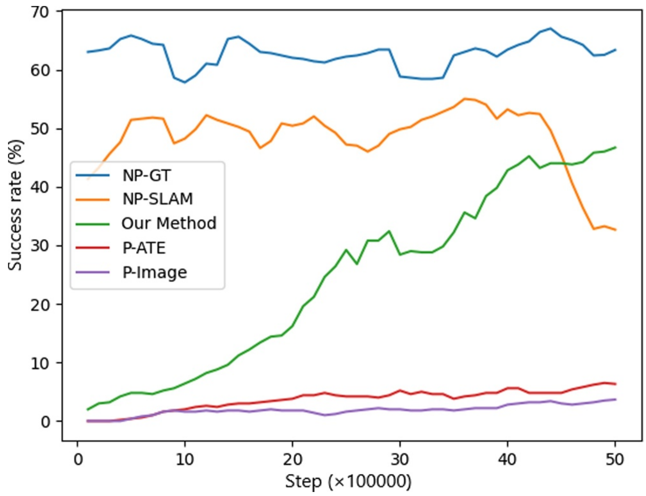

FIGURE 5 | Success rates achieved by different methods.

TABLE 2 | Test success rates in Env 1.

Method	Success rate
NP-GT	61%
NP-SLAM	33%
Our method	49%
P-image	2%
P-ATE	5%

$$e = \sqrt{\left(x_s - x_{gt}\right)^2 + \left(y_s - y_{gt}\right)^2} \quad (10)$$

where x_s and y_s represent the pose predicted by SLAM, x_{gt} and y_{gt} denote the ground-truth pose. The average error of our method is 0.127, whereas the average error of NP-SLAM is as high as 0.372. These results demonstrate that our method significantly enhances localisation accuracy. Furthermore, we record the pose error at the end of successful episodes for both methods. The average error of our method is 0.0173, while that of NP-SLAM is 0.0183. The result indicates that even in successful episodes, the error associated with the NP-SLAM method remains higher than that of our proposed method.

Figure 6 shows two instances where our method successfully completes the task, while the NP-SLAM leads to SLAM failure. In the first example (Figure 6a,b), the left and right sides of the environment are walls with poor features, while the upper and lower parts contain various environmental features. The NP-SLAM directs the robot to opt for the shortest path. However, when confronted with an entire wall that lacks sufficient features, SLAM failure occurs. In contrast, with our method, the robot traverses a path that may not be the shortest, but allows the robot to observe more environmental features, thereby ensuring the efficacy of SLAM.

In the second example (Figure 6c,d), the initial heading direction is opposite to the target. The policy initially directs the robot to execute a large-angle rotation in place, before it

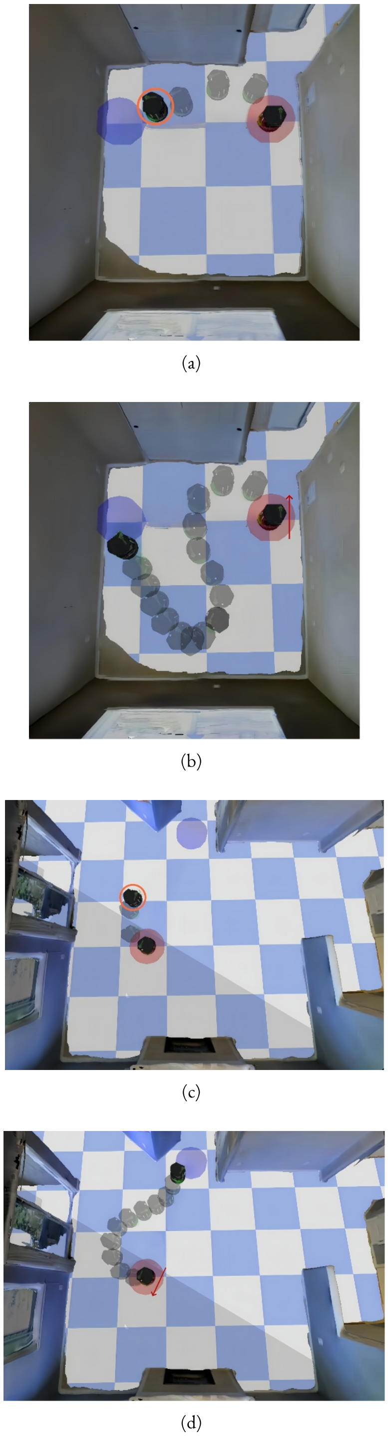

FIGURE 6 | Two examples of SLAM failures caused by the NP-SLAM. (a) Case 1 of NP-SLAM, (b) Case 1 of our method, (c) Case 2 of NP-SLAM, (d) Case 2 of our method. The red and blue circles denote the starting and target locations, respectively. The arrow represents the initial direction of the robot. In panels (a) and (c), the orange circle represents the locations of the SLAM failures.

proceeds to move towards the target. However, the substantial changes in environmental perception during the rotation result in a significant error in SLAM localisation. Consequently, the task fails due to the considerable localisation error incurred during navigation. In contrast, our approach avoids the large rotation and guides the robot along a path closer to the door, leveraging a greater number of visual features to enhance the accuracy of its localisation.

Furthermore, utilising the ATE-based reward function proposed in [14] yields a mere 5% success rate, suggesting its ineffectiveness in training the policy. Penalising the agent based on the discrepancy between the current estimated pose and the ground-truth is inappropriate. This is because the error may not arise solely from the last executed action but could be accumulated from a sequence of past actions executed. Moreover, applying a uniform criterion to assess SLAM performance limits its generalisability, as the performance of SLAM varies significantly across diverse environments. Judging whether the localisation fails or not using identical criteria is imprecise. The experimental results further validate the efficacy of our proposed reward function.

Additionally, in accordance with our expectations, our proposed method outperforms the method using raw images as inputs in terms of success rate. This is mainly due to the high dimensionality of raw RGB images, where the policy struggles to extract useful information for the task. In addition, this may require more training steps and resources for training. The result suggests that our proposed input representation is more effective in training the policy.

To verify the capacity of generalisation of each method, we test each in two unknown environments (Env 2 and Env 3 as shown in Figure 7). Each method is tested for 100 episodes, and the corresponding success rates are presented in Table 3.

Given that Env 2 and Env 3 exhibit simpler layouts than Env 1, both methods demonstrate higher success rates in these environments. Moreover, the success rate is highest when the robot's pose is accurate, and it decreases significantly when the pose is derived from SLAM. Notably, our method consistently achieves the highest success rate, outperforming NP-SLAM by 18% and 16% in Env 2 and Env 3, respectively. We also record the pose errors at the end of the test episodes in both Env 2 and Env 3. The average error of our method is 0.133 and 0.144 in these environments, respectively, while the average error for NP-SLAM is 0.417 and 0.463, respectively. Furthermore, our method outperforms the method employing the ATE-based reward function by 50% and 45% in these environments, and the approach utilising raw images by 52% and 47% respectively. The experiments in virtual environments reveal the remarkable performance of our proposed method in terms of success rate, localisation errors and generalisation capability.

4.3 | Study on RPE Dynamic Threshold

In this section, we study the effect of our proposed dynamic threshold δ_p. Specifically, we train another model with a pre-set threshold. When the RPE exceeds the threshold, r_{pose} in Equation (7), is used. The rest of the reward function remains the same. To set a suitable threshold, we run the robot for 10,000 timesteps randomly and record the RPE generated at each timestep. Then, we rank them in descending order. Subsequently, we select the smallest RPE value (0.0026) that falls

(a)

(b)

FIGURE 7 | Experiment environments for testing. (a) Env 2 (Arona), (b) Env 3 (Benicia).

TABLE 3 | Test success rates in Env 2 and Env 3.

Method	Success rate	
	Env 2	Env 3
NP-GT	65%	63%
NP-SLAM	42%	37%
Our method	60%	53%
P-image	8%	6%
P-ATE	10%	8%

within the top 10% of the ranked list as the threshold. After training, we also conduct tests in Env 1, Env 2 and Env 3. The success rates are shown in Table 4.

The results show that our method has significantly higher success rates in both environments. It suggests that pre-setting the threshold is not suitable for training an effective policy. In contrast, our proposed dynamic threshold significantly enhances the performance of the model.

5 | Conclusion

In this paper, we introduce a novel DRL-based mapless navigation method without relying on the assumption of the availability of localisation. Our method uses RGB-D-based ORB-SLAM2 for localisation in indoor environments. Our trained policy not only directs the robot towards the target location but also enhances robot pose estimation by considering the quality of the observed features along the selected paths. We propose a compact state representation based on the spatial distributions of map points, thereby enhancing the robot's awareness of areas with reliable map points. Additionally, RPE is incorporated into the reward function instead of ATE. This allows the policy's responsiveness to each single action. Furthermore, a dynamic threshold is introduced to enhance the policy's adaptability to variations in SLAM performance across different environments. The experiments in localisation-challenging environments demonstrate that our method outperforms existing methods in terms of success rate. In future studies, pedestrian detection and classification will be incorporated into the network to implement our method within dynamic environments. Ultimately, we will aim to implement this research on real robots and assess its performance in real-world scenarios. Although iGibson provides high-fidelity simulation, it differs from real-world environments in texture diversity, dynamic elements, and lighting variations, which can impact SLAM performance. Future work could enhance realism by introducing dynamic scenes and real-world sensor noise. Moreover, sim-to-real transfer remains a challenge; exploring domain adaptation techniques such as fine-tuning or self-supervised learning could improve model robustness in real-world applications.

TABLE 4 | Test success rates in Env 1/Env 2/Env 3.

Method	Success rate		
	Env 1	Env 2	Env 3
Pre-set threshold	4%	4%	5%
Our method	49%	60%	53%

Acknowledgements

Yan Gao thanks the Chinese Scholarship Council (CSC) for providing the living stipend for his Ph.D. programme (No. 202008230171).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The authors have nothing to report.

References

1. B. Patle, L. Ganesh Babu, A. Pandey, D. Parhi, A. Jagadeesh, et al., "A Review: On Path Planning Strategies for Navigation of Mobile Robot," *Defence Technology* 15, no. 4 (2019): 582–606, https://doi.org/10.1016/j.dt.2019.04.011.

2. T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic Approaches in Robot Path Planning: A Survey," *Robotics and Autonomous Systems* 86 (2016): 13–28, https://doi.org/10.1016/j.robot.2016.08.001.

3. 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), 6582–6588.

4. V. Mnih, A. P. Badia, M. Mirza, et al., "Asynchronous Methods for Deep Reinforcement Learning," in *International Conference on Machine Learning* (PMLR, 2016), 1928–1937.

5. E. Marchesini and A. Farinelli, "Discrete Deep Reinforcement Learning for Mapless Navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020), 10688–10694.

6. 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).

7. 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), 31–36.

8. 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), 3357–3364.

9. R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source Slam System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics* 33, no. 5 (2017): 1255–1262, https://doi.org/10.1109/tro.2017.2705103.

10. K. Naveed, M. L. Anjum, W. Hussain, and D. Lee, "Deep Introspective Slam: Deep Reinforcement Learning Based Approach to Avoid Tracking Failure in Visual Slam," *Autonomous Robots* 46, no. 6 (2022): 705–724, https://doi.org/10.1007/s10514-022-10046-9.

11. V. Prasad, K. Yadav, R. S. Saurabh, et al., "Learning to Prevent Monocular Slam Failure Using Reinforcement Learning," in *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing* (2018), 1–9.

12. F. Lin, Z. Ji, C. Wei, and H. Niu, "Reinforcement Learning-Based Mapless Navigation With Fail-Safe Localisation," in *Annual Conference Towards Autonomous Robotic Systems* (Springer, 2021), 100–111.

13. F. Lin, Z. Ji, C. Wei, and R. Grech, "Localisation-safe Reinforcement Learning for Mapless Navigation," in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2022), 1327–1334.

14. Y. Chen, Q. Qiu, X. Liu, et al., "Deep Reinforcement Learning for Localizability-Enhanced Navigation in Dynamic Human Environments," *arXiv preprint arXiv:2303.12354* (2023).

15. W. Chen, W. Li, A. Yang, and Y. Hu, "Active Visual Slam Based on Hierarchical Reinforcement Learning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2023), 7155–7162.

16. H. Taheri and Z. C. Xia, "SLAM; Definition and Evolution," *Engineering Applications of Artificial Intelligence* 97, no. 104 (2021): 032, https://doi.org/10.1016/j.engappai.2020.104032.

17. L. Xie, *Reinforcement Learning Based Mapless Robot Navigation*. Ph. D. thesis (University of Oxford, 2019).

18. J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning," *IEEE Transactions on Vehicular Technology* 69, no. 12 (2020): 14413–14423, https://doi.org/10.1109/tvt.2020.3034800.

19. H. Niu, Z. Ji, F. Arvin, B. Lennox, H. Yin, and J. Carrasco, "Accelerated Sim-To-Real Deep Reinforcement Learning: Learning Collision Avoidance From Human Player," in *2021 IEEE/SICE International Symposium on System Integration (SII)* (IEEE, 2021), 144–149.

20. D. M. Saxena, V. Kurtz, and M. Hebert, "Learning Robust Failure Response for Autonomous Vision Based Flight," in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2017), 5824–5829.

21. S. Rabiee and J. Biswas, "IV-SLAM: Introspective Vision for Simultaneous Localization and Mapping," in *Conference on Robot Learning* (PMLR, 2021), 1100–1109.

22. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT press, 2018).

23. T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al., "Continuous Control With Deep Reinforcement Learning," *arXiv preprint arXiv:1509.02971* (2015).

24. 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), 7520–7527.