LiDAR-Based End-to-End Active SLAM Using Deep Reinforcement Learning in Large-Scale Environments

Jiaying Chen, Keyu Wu, Minghui Hu, Ponnuthurai Nagaratnam Suganthan, *Fellow, IEEE*, and Anamitra Makur

***Abstract*—Autonomous exploration in expansive and complicated environments poses a significant challenge. When the dimensions of the environment expand, exploration algorithms encounter substantial overhead, which can overpower the computational capacity of mobile platforms. In this paper, we propose a novel 3D LiDAR-based end-to-end autonomous exploration network architecture, which allows mobile robots to learn to explore autonomously in expansive environments through deep reinforcement learning. Specifically, we utilize both scans from the LiDAR sensor and maps obtained by SLAM as exploration information to predict the robot's linear and angular actions simultaneously. Furthermore, in order to enhance exploration capability, intrinsic rewards are also used during training. Compared to the existing methods, our proposed approach demonstrates improved learning efficiency and adaptability for various environments. Moreover, the proposed method can complete exploration in unknown environments with a shorter trajectory length than state-of-the-art methods. Additionally, experiments are conducted on the physical robot. which indicates that the trained network can be seamlessly transferred from the simulation to the real world.**

***Index Terms*—Deep reinforcement learning, automatic exploration, LiDAR active SLAM, collision avoidance.**

I. Introduction

SIMULTANEOUS Localization and Mapping (SLAM) is a fundamental area of robotics research. It involves a robot simultaneously building a map of its surroundings and simultaneously determining its own location based on onboard sensors. With the ability to offer highly precise and long-range 3D measurements of surrounding environments, 3D LiDARs are becoming essential sensors used in SLAM systems, commonly known as LiDAR-based SLAM. Active SLAM (A-SLAM) refers to the integration of active decision-making or control strategies within the SLAM framework [1]. In traditional SLAM, robots passively collect data and build maps with external controllers or human efforts. A-SLAM, however, enables mobile robots to autonomously explore and map environments, which is widely applied in practices, such as search, rescue, and sterilization.

Due to the growing interest from diverse domains, several notable contributions have emerged in the field. These include the development of frontier-based methods [2], [3], [4], [5], sampling-based methods [6], [7], [8], and information-based methods [9], [10], [11], [12], [13]. The idea of frontier-based approaches is to discover all frontiers which are boundaries between explored and unexplored areas. Subsequently, one of these frontiers is chosen as the next target. However, the majority of these approaches greedily select frontiers to explore, resulting in significant computational overhead as the environment size increases. The mobile system's computing capacity may be overloaded by this overhead. Moreover, the frequency of exploration planning is limited, impeding the robot's ability to promptly respond to environmental changes and sometimes requiring it to pause and await the completion of planning processes. Sampling-based methods primarily involve the random sampling of some viewpoints, followed by the computation of sampled paths' gains to pick the optimal one with the most ability to observe unknown areas. However, these methods need numerous computational resources to calculate the viewpoints' gains, and there is a risk of getting trapped in local regions due to insufficient global consideration while exploring expansive environments. Information-based methods utilize Shannon entropy [14] to characterize the environmental map's uncertainty and formulate optimization problems. The methods optimize the mutual information gain throughout the following several actions during the automatic exploration process. However, the methods used to construct optimization problems often result in scenarios where a mathematical optimal solution is not attainable in most cases. Then, some researchers combine information-based methods with the other two methods. [15], [16] employ information gain to evaluate frontier points and determine the optimum point for the subsequent move. However, as the size of the exploration area expands, the computational time required for this decision-making process grows substantially. Generally, the conventional methods mentioned above primarily focus on identifying the best next point based on current points and the explored map. However, these methods

Manuscript received 23 November 2023; revised 9 April 2024; accepted 20 May 2024. Date of publication 27 May 2024; date of current version 17 October 2024. The review of this article was coordinated by Dr. Ying He. *(Corresponding author: Jiaying Chen.)*

Jiaying Chen, Minghui Hu, and Anamitra Makur are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: jiaying001@e.ntu.edu.sg; e200008@e.ntu.edu.sg; eamakur@ntu.edu.sg).

Keyu Wu is with the Agency for Science, Technology and Research (A*STAR), Singapore 138632 (e-mail: wu_keyu@i2r.a-star.edu.sg).

Ponnuthurai Nagaratnam Suganthan is with the KINDI Center for Computing Research, College of Engineering, Qatar University, Doha, Qatar (e-mail: p.n.suganthan@qu.edu.qa).

Digital Object Identifier 10.1109/TVT.2024.3405483

suffer from reduced adaptability due to their heavy reliance on expert features of maps. Additionally, it is challenging to develop a generic termination mechanism that balances computational cost and exploration efficiency.

Recently, Deep Reinforcement Learning (DRL) has garnered substantial attention and observed significant development to address the aforementioned problem. Since Mnih et al. [17] apply the Deep Q-Network (DQN) and gain superior performance with video games like Atari than human players, DRL has become incredibly popular in addressing problems involving sequential decision-making. Within the framework of DRL, decision-making strategies are acquired through iterative trial-and-error interactions with the surrounding environments, which closely resembles how humans learn from experience and make decisions. Lately, DRL has achieved remarkable progress in large-scale real-time strategy games. Consequently, its application has extended to domains such as robotics exploration and navigation in unknown environments.

Currently, DRL methods are mostly combined with frontier points in LiDAR-based exploration. The DRL algorithm adapts its exploration strategy based on the observed frontier points and selects the one that maximizes the expected reward or information gain [18], [19], [20], [21]. Nonetheless, there are two disadvantages associated with frontier-based DRL methods. Firstly, the robot may encounter collisions with suddenly appeared obstacles while navigating towards the selected frontier points. These obstacles, which are not initially accounted for in the frontier points, can impede the robot's movement toward the selected points and potentially lead to collisions. Secondly, to address obstacle avoidance challenges, additional path planning and control algorithms are required. The integration of these algorithms complicates the overall system architecture and may increase the computational burden.

Motivated by these considerations, we develop an end-to-end automatic exploration system that can directly and effectively transform the raw LiDAR sensor observations to the robot control commands. Our framework utilizes a dueling DDQN (D3QN) [22] architecture consisting of two input streams designed to process different types of sensor data. The first input stream involves projected 2D depth images by encoding all the point cloud information. These depth images capture essential spatial information about the environment. The second input stream focuses on local occupancy grid map images, providing a structured representation of the surroundings. CNNs are employed in each stream to extract features from the input. Subsequently, the resultant features from both streams are merged and generate two separate Q-values by distinct noisy fully connected layers. By estimating these Q-values, the network can simultaneously generate angular and linear control commands, enabling the robot to navigate and explore the environment effectively. We also introduce a β-consistency action selection strategy, which considers the coherence of angular control commands during the selection process. By taking into account the previously executed velocity commands as well as the estimated Q-values, an angular velocity is determined. This approach significantly enhances the stability of the system by incorporating motion filtering. Additionally, the agent develops the ability to navigate out of dead ends by executing consistent actions. Furthermore, intrinsic rewards are integrated into the training procedure to enhance the system's exploration capacity. These intrinsic rewards are computed by random network distillation (RND) [23], which leverages the prediction error of a random network as exploration bonuses, aimed at providing higher rewards for unexplored states in contrast to those already encountered. In other words, it assigns higher rewards to states that are more dissimilar, thus encouraging efficient exploration. Therefore, the primary contributions include:

- We propose a novel end-to-end DRL-based automatic exploration framework, which considers both point cloud and map information. This framework can transform raw LiDAR sensor information into robot control commands, enabling efficient navigation and exploration.
- We develop an effective reward function designed specifically for the exploration framework. This function incorporates both extrinsic rewards, considering velocity adaptation and map exploration, and intrinsic rewards, resulting in an overall improvement in effectiveness.
- We introduce a β-consistency action selection strategy that improves system stability and enables the agent to navigate out of dead ends.
- Experiments have been conducted in simulated as well as real environments. The results illustrate the superiority of our A-SLAM system and show its remarkable transferability to real-world environments that include both static and dynamic objects.

II. Related Work

Active SLAM: The task of a robot exploring and mapping an unknown environment autonomously and efficiently is referred to as autonomous robot exploration, or A-SLAM. It can be categorized into two groups: conventional methods and DRL-based methods. In terms of conventional methods, Yamauchi [2] introduces the first frontier-based technique, wherein the robot advances to the nearest frontier. Sample-based methods aim to randomly sample certain viewpoints and subsequently choose the optimal one that can observe more unknown areas [7], [24], [25]. Information-based exploration methods often estimate the mutual information (MI) between the cells in an occupancy grid map and a robot's upcoming sensor observation to assess the next-view candidates under consideration. Bourgault et al. [9] present an information-based method, which considers the trade-off between reducing the pose uncertainty and optimizing MI within a robot's SLAM procedure. Bai et al. [13], [26] combine the Gaussian process and Bayesian optimization techniques to assess MI across action space and choose the one that optimizes the map's entropy. For autonomous exploration in expansive environments, Cao et al. [27] propose TARE that focuses on finding the shortest path that goes through viewpoints, either on a local or global scale, to ensure thorough exploration of the unknown area. FAEL [28] is proposed for autonomous exploration in expansive environments. It introduces a quick preliminary treatment of environment data involving viewpoints,

frontiers, information gains, and distances among viewpoints. It also provides a path optimization mechanism to efficiently determine the navigation path.

Deep reinforcement learning: Given the remarkable progress in DRL in the last few years, numerous researchers in the field of intelligent control attempted to use DRL to address the sequential decision-making problem by viewing robot exploration as an optimal control issue. DQN, initially proposed by Mnih et al. [17], demonstrates impressive results in diverse domains, such as video games, autonomous navigation, and robotic manipulation. Tai et al. [29] train their network through DQN for a mobile robot with an RGB-D camera to prevent obstacle collisions in unknown environments. Building upon this foundation, researchers further enhance the performance of DQN through extensions, for example, Double DQN (DDQN) [30]. DDQN solves the overestimation bias issue observed in traditional DQN, leading to improved stability and more accurate value estimations. Furthermore, the combination of DDQN and Prioritized Experience Replay (PER) [31] shows promising results in enhancing sample efficiency and overall learning performance. PER prioritizes experiences based on their impact on learning, allowing the agent to prioritize and learn from more informative experiences. Additionally, D3QN [22] emerges as a noteworthy development, separating the value and advantage streams to better represent the value function. Instead of relying on the commonly employed epsilon-greedy approach for exploration, the NoisyNet [32] introduces parameter noise to encourage exploration and promote a better exploration-exploitation trade-off during training. However, these algorithms cannot be used in action spaces with numerous dimensions [33], i.e., only allow for a limited number of discrete control outputs. To address this problem and enhance autonomous steering performance, we apply a branching noisy architecture within the D3QN framework to simultaneously derive linear and angular velocities as well as a β-consistency action selection strategy. Moreover, the proposed approach also leverages RND intrinsic rewards [23] together with the extrinsic rewards. In robot exploration, intrinsic reward plays a vital role due to the absence of a clear ground truth. Intrinsic reward facilitates more effective and adaptive exploration strategies by encouraging visiting novel states or learning the environment dynamics. In [34], intrinsic motivation is employed to reward signals to assist mobile robots in learning laser-based navigation policies.

DRL in A-SLAM: Due to the significant number of interactions required by DRL-based approaches to identify effective policies, simulators are commonly used to train policies in virtual environments. Consequently, many DRL-based methods leverage laser sensors due to their higher degree of transferability from simulation to the real world. For instance, [35] and [36] propose autonomous navigation systems for goal-driven exploration via using Twin Delayed Deep Deterministic Policy Gradient (TD3) [37]. Zheng et al. [38] present a hierarchical approach with Proximal Policy Optimization (PPO) [39] for mobile robot exploration in pedestrian crowds. Li et al. [18] construct a DQN-based technique to determine the subsequent target within a grid map and apply an additional path-planning algorithm to control the robot's movement. Nevertheless, three-dimensional scenarios cannot be adequately described by the limited sensory information supplied by laser scanners [40].

With the growing popularity of LiDARs, researchers have turned their attention to LiDAR-based approaches. Cao et al. [21] propose a LiDAR-based autonomous exploration network combining an attention-based neural network with Soft Actor-critic (SAC) [41], which selects one neighboring node in a collision-free graph as the robot's next target waypoint. As the robot proceeds to its next destination along a straight trajectory, there exists the inherent risk of collisions with unanticipated obstacles that may suddenly appear in its path. Hence, we propose an end-to-end DRL-based framework that enables simultaneous predictions of the robot's linear and angular actions, ensuring timely and adaptive response capabilities to effectively navigate and mitigate collision risks. Liu et al. [42] present an end-to-end DRL network incorporating asynchronous advantage actor-critic (A3C) [43] and imitation learning, which enables motion planning in crowded and cluttered environments by using LiDAR and grayscale stereo camera sensors. All the aforementioned methods use 2D occupancy grid maps as LiDAR input. Nevertheless, the occupancy grid map is not informative enough to describe three-dimensional environments since it does not capture the vertical dimension or detailed spatial information in the form of height, obstacles, or varying terrain levels. Consequently, relying solely on the occupancy grid map may lead to inadequate obstacle avoidance capabilities. Unlike 2D laser data that can be directly utilized as network input, 3D mechanical LiDAR poses a challenge due to its extensive point cloud containing tens of thousands of points that are sparse and unordered. In this paper, we encode the point cloud to a projected 2D image that can be fed into the network.

III. Methodology

The overview of our LiDAR-based Active SLAM system is illustrated in Fig. 1. The A-SLAM module comprises two sub-modules: one is the traditional SLAM, and the other one is the exploration DRL network. 3D LiDAR installed on the autonomous robot platform provides surrounding information. According to this information, the traditional SLAM builds the point cloud map and estimates the robot's localization in the map. This work builds upon a versatile LiDAR-inertial SLAM framework SE2LIO [44], which was proposed in our previous work. The SE2LIO SLAM system fuses LiDAR data, inertial measurement unit (IMU) measurements, and ground constraints within a bundle adjustment (BA) based optimization framework. It uses IMU pre-integration to de-skew the LiDAR scan points and provide an initial guess for the LiDAR odometry optimization. Similar to the LOAM [45], the de-skewed point clouds are then used to estimate the transformation relating consecutive scans by minimizing the distance from edge feature points to corresponding lines and planar feature points to corresponding planes. The obtained LiDAR odometry is subsequently used to update the bias of the IMU sensors. Additionally, the ground constraints are integrated into the BA-based optimization framework to reduce the pose drift. The extracted features are then

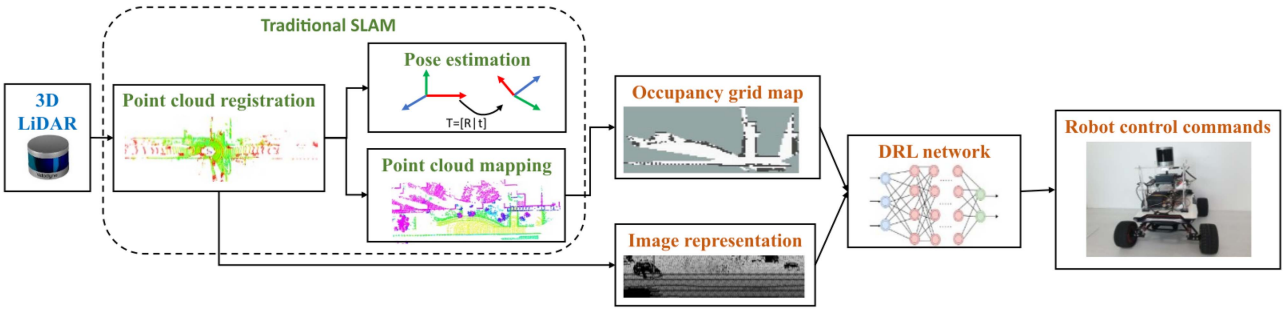

Fig. 1. Overview of our Active SLAM system.

registered to build a global point cloud map. This 3D map is projected into a 2D occupancy map representation, which serves as one type of input to the DRL network. The second type of input to the DRL network is the depth images created by encoding the sparse and unordered 3D point clouds. This depth information complements the 2D occupancy map, providing the DRL agent with a more comprehensive environmental representation. Finally, the DRL network generates control commands for the robot to explore areas that haven't been visited yet.

A. LiDAR Data Encoding

To restructure the original sparse and unordered point clouds into a format appropriate for network input, we utilize a cylindrical projection [46] to encode the LiDAR data into an image coordinate-parameterized representation. To derive the depth image representation, we apply a geometric mapping in the form of $\Phi : \mathbb{R}^{n\times 3} \rightarrow \mathbb{R}^{1\times H\times W}$, where H denotes the height of the image and is chosen to be the number of LiDAR vertical scan lines; W denotes the image width which should not be set larger than the horizontal resolution of LiDAR. In this paper, we use Velodyne VLP-16 LiDAR, resulting in $H = 256$ and $W = 16$. Each 3D point $p = (x_p, y_p, z_p)$ in LiDAR coordinate is mapped onto the 2D image coordinates (u, v) represented as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_h/2 - \arctan(y_p, x_p)/\delta_h \\ f_v/2 - \arctan(z_p, d_p)/\delta_v \end{pmatrix}, \tag{1}$$

where depth $d_p = \sqrt{x_p^2 + y_p^2}$; δ_h and δ_v are the horizontal and vertical resolution for pixel representation; f_v and f_h denote the vertical and horizontal field of view (FOV). In situations where multiple 3D points are mapped to identical pixel coordinates, we opt for the closest point to serve as the pixel value.

B. Deep Reinforcement Learning

1) Problem Formulation: We construct the task of automatic exploration as a Markov decision process (MDP) formulated in a quintuple:

$\langle S; A; R; P; \gamma \rangle$, where S and A symbolize the state space and action space; R, P, and $\gamma \in [0, 1]$ represent immediate reward, transition probabilities, and discount factor, respectively. In this framework, the agent interacts with the surroundings and acquires the policy π by maximizing the expected discounted return $R_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$.

2) Deep Q-Learning: The Q-value denotes the expected cumulative reward of a state-action pair for a given policy π:

$$Q_\pi = E_\pi[R_t \mid s_t, a_t], \tag{2}$$

In deep Q-learning, deep neural networks parameterized by θ are utilized to estimate the optimal Q-value: $Q^*(s, a) \approx Q(s, a; \theta)$. The optimal action a_t^* is selected by identifying the highest Q-value and is carried out at the state s_t. D3QN [22] splits the network into two branches to calculate the value function $V(s)$ and advantage function $A(s, a)$ separately. The Q function is the result of combining a value function and an advantage function: $Q(s_t, a_t; \theta, \theta_V, \theta_A) = V(s_t; \theta, \theta_V) + A(s_t, a_t; \theta, \theta_A) - \frac{1}{|N|}\sum_{a'} A(s_t, a'; \theta, \theta_A)$, where N is the number of actions; θ, θ_V and θ_A denote the parameters of the common layers, the value branch, and the advantage branch, respectively.

For model training, we adopt D3QN. However, instead of generating a single Q-value vector that includes both angular and linear actions, our proposed network generates two separate Q-value vectors corresponding to linear and angular velocities separately. To achieve this, we employ three separate branches in our model to predict the state-value function $V(s)$, as well as the advantage functions A_1 and A_2. Then, an aggregation layer integrates the state value with each advantage function to calculate the Q-value functions Q_1 and Q_2, respectively. As a result, each Q-value is calculated by:

$$Q_i(s, a_i) = V(s) + A_i(s, a_i) - \frac{1}{|N_i|}\sum_{a_i'} A_i(s, a_i'),\ i = 1, 2, \tag{3}$$

3) NoisyNets for Exploration: Classical DRL techniques employ a ϵ-greedy policy to arbitrarily perturb the agent's policy and generate diverse exploration behaviors. It is necessary to adjust the hyperparameters, though, and it is challenging to produce various behaviors for effective exploration due to the induced local dithering disturbances. As a result, we implement the concept of NoisyNets [32] in this work for efficient exploration. Consider a linear layer without noise, represented by $y = wx + b$, where y, w, x, and b represent the output, weight matrix, input, and bias, respectively. Gaussian noises are used to add uncertainty to accomplish the exploration, defined by:

$$y = (\mu^w + \sigma^w \odot \epsilon^w)x + \mu^b + \sigma^b \odot \epsilon^b, \tag{4}$$

where the parameters μ^w, μ^b, σ^w and σ^b are learnable whereas ϵ^w and ϵ^b are noise random variables. In this manner, the RL algorithm automatically adjusts the amount of noise introduced into the network, enabling state-conditioned and self-annealing exploration. Moreover, factorized Gaussian noises are applied to shorten the computing time of random number generation. Therefore, ϵ^w and ϵ^b are factorized as following:

$$\epsilon^w_{i,j} = f(\epsilon_i) f(\epsilon_j), \quad \epsilon^b_j = f(\epsilon_j), \tag{5}$$

where we use $f(x) = sgn(x)\sqrt{x}$ in our network. Each element $u_{i,j}$ is initialized by a sample from independent distribution in the interval $[-1/\sqrt{L}, 1/\sqrt{L}]$, where L represents the count of input. Each element $\sigma_{i,j}$ is initialized by a constant $\sigma_0/\sqrt{t}$, $\sigma_0 = 0.5$ in our experiments.

4) Intrinsic Reward and β-Consistency Strategy: During our research, we consider the RND bonuses [23] as intrinsic rewards to improve exploration during training. The RND bonus is chosen due to its computational efficiency, ease of implementation, and effectiveness in handling image-based inputs. Compared to other prediction error-based exploration bonuses, it also contributes to mitigating the interference resulting from unexpected stochastic transitions. There are two neural networks utilized in estimating the RND bonuses: a prediction network and a static target network. The prediction network is trained using gathered data, while the target network is initialized at random. In both networks, the LiDAR input at time step $t+1$ is processed through a sequence of CNN layers, succeeded by linear layers. The prediction network's parameters, represented as θ_p, are refined via minimizing the mean squared error, defined as:

$$L(\theta_p) = \|g_p(s_{t+1}; \theta_p) - g_t(s_{t+1})\|^2, \tag{6}$$

where g_p and g_t represent prediction and target networks' functions. The normalized prediction errors are treated as intrinsic rewards, motivating the exploration of unfamiliar states.

Moreover, to enhance the learning policy's performance, we present a β-consistency action selection strategy that can address the agent's left-right swing behavior and avoid dead ends effectively. The β-consistency approach functions as an added layer of motion filtering by evaluating the coherence in angular velocity commands. The selection of angular velocity takes into account both the previous step's executed command and the estimated Q-values:

$$a^*_{2,t} = \begin{cases} a^*_{2,t-1}, \text{ if } \frac{\exp\left(Q^*_2(s_t, a_{2,i})\right) - \exp\left(Q_2(s_t, a^*_{2,t-1}; \theta)\right)}{\sum_{i=1}^{N} \exp\left(Q_2(s_t, a_{2,i}; \theta)\right)} < \beta \\ \arg\max_{a_{2,i}} Q_2(s_t, a_{2,i}; \theta), \text{ otherwise} \end{cases}, \tag{7}$$

where β denotes the threshold and $a^*_{2,t-1}$ is the previouly executed action.

5) Prioritized Experience Replay: We also implement Prioritized Experience Replay (PER) [31]. PER addresses the inefficient sampling issue by emphasizing the replay of experiences that are more informative and valuable for learning. In PER, experiences are assigned priorities via the magnitude of their temporal difference error (TD-error). During the replay phase, experiences with higher priorities are sampled more frequently, allowing the agent to focus on important and informative transitions. PER utilizes stochastic prioritization, which generates the probability of the sampling transition k :

$$P(k) = \frac{p^a_k}{\sum_i p^a_i}, \tag{8}$$

where p_k denotes the priority of the k^{th} experience, a is used to reintroduce some randomness in the experience selection. The importance-sampling weights introduced for updating the network are computed as $w_k = (N \cdot P(k))^{-b} / \max_i w_i$, where b is to control how much these sampling weights affect learning. Finally, update the transition priority by $p_k = |\delta_k| + e$, where δ_k is TD-error, and e is a constant to prevent zero priority.

Overall, we propose an improved D3QN with NoisyNets, RND intrinsic reward, and PER for robot autonomous exploration. At time step t, the target Q-values can be computed as:

$$y_i = R_t + \gamma Q_i(s_{t+1}, \arg\max_{a_{i,t+1}} Q_i(s_{t+1}, a_{i,t+1}, \epsilon; \theta), \epsilon^-; \theta^-), \tag{9}$$

where $i = 1, 2$ indicates the linear and angular velocities branch, respectively; θ and θ^- denote the online and target network's parameters; ϵ and ϵ^- are the online and target network's noise variables. The definition of the loss function is given by:

$$\begin{aligned} L(\theta) = \mathbb{E}[&\alpha_1 (y_1 - Q_1(s_t, a_{1,t}, \epsilon; \theta))^2 \\ &+ \alpha_2 (y_2 - Q_2(s_t, a_{2,t}, \epsilon; \theta))^2 \\ &+ \alpha_3 (Q_1(s_t, a_{1,t}, \epsilon; \theta) - Q_2(s_t, a_{2,t}, \epsilon; \theta))^2], \end{aligned} \tag{10}$$

where α_1, α_2, and α_3 are weighting factors. Our model is outlined in Algorithm 1.

C. Implementation Details

State space: The first kind of observation is a pile of encoded LiDAR images from three successive steps. The size of the depth images, derived from point clouds, is 16×256. The advantage of using depth images is that they can be easily obtained in real-time from the LiDAR sensor and provide a more detailed representation of close obstacles compared to distant obstacles which are less important to the robot. The second kind of observation is a stack of 2D occupancy grid maps representing a local extract of the surroundings. The map is centered around the robot and rotated around its heading. Considering that the LiDAR's maximum measurement range is 80 meters, we establish the dimensions of the occupancy grid map as 256×256, with a 0.4 m/pixel resolution. The benefit of employing an occupancy grid map is that it encodes entire environments within a certain range, allowing the robot to take more optimal actions from a long-term perspective.

Action space: The action space of our network comprises robots' linear and angular velocities. In our experimentation, we select a range of nine discrete linear velocities, which include 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.5, and 1.8 m/s. Similarly, the set of nine values for angular velocity includes $-\pi/3$, $-\pi/4$, $-\pi/6$, $-\pi/12$, 0, $\pi/12$, $\pi/6$, $-\pi/4$, and $\pi/3$ rad/s.

Algorithm 1: Dual-Branch D3QN With NoisyNet (DBD3QNN) Model.

Initialize state-action value function Q, reply buffer, and target network $\theta \leftarrow \theta^-$
Initialize Prioritized Experience Replay B to capacity M
Set priority $p_1 = 1$, $a = 0.6$, $b = 0.4$
for *t=1 to T* **do**
  Select $a^*_{1,t} = \arg\max_{a_{1,i}} Q_1(s_t, a_{1,i}; \theta)$
  if *use* β*-consistency* (7) **and** $\frac{\exp(Q_2^*(s_t,a_{2,i})) - \exp(Q_2(s_t,a^*_{2,t-1};\theta))}{\sum_{i=1}^{N}\exp(Q_2(s_t,a_{2,i};\theta))} < \beta$ **then**
    Select $a^*_{2,t} = a^*_{2,t-1}$
  else
    Select $a^*_{2,t} = \arg\max_{a_{2,i}} Q_2(s_t, a_{2,i}; \theta)$;
  end
  Execute $a^*_{1,t}$ and $a^*_{2,t}$ simultaneously
  Observe s_{t+1}, and reward R_t
  Store $(s_t, a^*_{1,t}, a^*_{2,t}, R_t, s_{t+1})$ in B with priority $p_t = \max_{i<t} p_i$
  if $t > M$ **then**
    for *j=1 to batch size* N_B **do**
      Sample a batch of transitions $(s_t, a_{1,t}, a_{2,t}, R_t, s_{t+1})$ from B
      Sample noise variables $\epsilon_i, \epsilon_j, \epsilon_i^-, \epsilon_j^- \sim N(0,1)$
      if *episode terminates at* $j+1$ **then**
        $y_1 = R_t$ and $y_2 = R_t$
      else
        $y_1 = R_t + \gamma Q_1(s_{t+1}, \arg\max_{a_{1,t+1}} Q_1(s_{t+1}, a_{1,t+1}, \epsilon; \theta), \epsilon^-; \theta^-)$
        $y_2 = R_t + \gamma Q_2(s_{t+1}, \arg\max_{a_{2,t+1}} Q_2(s_{t+1}, a_{2,t+1}, \epsilon; \theta), \epsilon^-; \theta^-)$ (9)
      end
      Update transition priority: $p_j \leftarrow |\delta_j|$
    end
    Minimize loss function
    $L(\theta) = \mathbb{E}[\alpha_1(y_1 - Q_1(s_t, a_{1,t}, \epsilon; \theta))^2 + \alpha_2(y_2 - Q_2(s_t, a_{2,t}, \epsilon; \theta))^2 + \alpha_3(Q_1(s_t, a_{1,t}, \epsilon; \theta) - Q_2(s_t, a_{2,t}, \epsilon; \theta))^2]$ (10)
    if *t mod* N_t **then**
      Update target network $\theta \leftarrow \theta^-$
    end
  end
end

Reward: To promote effective exploration, the robot receives a reward involving three components after executing each movement action a_t:

$$R_t = \begin{cases} R_{finish}, & \text{if } \rho > 0.9 \\ R_{collision}, & \text{if collide} \\ \lambda_1 R_{map} + \lambda_2 R_{control}, & \text{otherwise} \end{cases}, \tag{11}$$

where

- $R_{finish} = 30$ is a positive reward when the ratio of the map's explored region $\rho > 0.9$.
- $R_{collision} = -30$ is the negative penalty when the robot collides with obstacles.
- R_{map} is the information gain reward. Inspired by the Shannon's entropy [14], We define it over an occupancy grid map m as follows:

$$H(m) = -\sum_i \sum_j p(m_{i,j}) \log p(m_{i,j}), \tag{12}$$

where $p(m_{i,j})$ represents the occupied probability of the grid cell in the i^{th} column and j^{th} row. Every cell can exist in three possible states: free, occupied, and unknown. Cells that have not been previously observed are assigned a value of $p(m_{i,j}) = 0.5$, providing a unit of entropy for every cell. Cells with complete information do not contribute any entropy to the sum. Mutual information is employed for evaluating the expected information gain reward, and it is characterized as follows:

$$R_{map} = H(m_{t-1}) - H(m_t), \tag{13}$$

- $R_{control}$ is defined as

$$R_{control} = vcos(\lambda_3 vw), \tag{14}$$

where w and v represent the angular and linear velocities. The coefficient λ_3 is utilized to control the robot's behavior. Specifically, when the angular velocity is low, the objective is for the robot to move ahead at maximum speed. However, if a larger angular velocity is required, the constraint on linear velocity is relaxed to ensure safety considerations. In this paper, we design an effective instant reward function that includes R_{map} for map exploration and $R_{control}$ for adaptive control strategy, which is crucial for ensuring smooth robot movement and effective exploration.

D. Network Architecture

The proposed network architecture referred to as the Dual-Branch D3QN with NoisyNet (DBD3QNN) is depicted in Fig. 2. It consists of two different channels that separately process LiDAR range information and occupancy map. For the LiDAR image channel, a series of three 16×256 depth images acquired from consecutive time steps are combined to form the input. After that, the raw depth input is fed into three convolutional layers with the ReLU activation function, to generate feature representation. The acquired feature representation is subsequently flattened and sent to a linear layer consisting of 1024 units, activated by the ReLU function. Similarly, in the other channel, a set of three 256×256 occupancy maps obtained from consecutive time steps is concatenated together to form the input, and then processed through three convolutional layers. The yielded feature maps are flattened and fed through a linear layer comprising 1024 units, activated by the ReLU function as well. The ultimate feature vector is formed by concatenating the output derived from the aforementioned processes.

The network then branches out into three streams. One branch focuses on estimating the state-value function, meanwhile, two

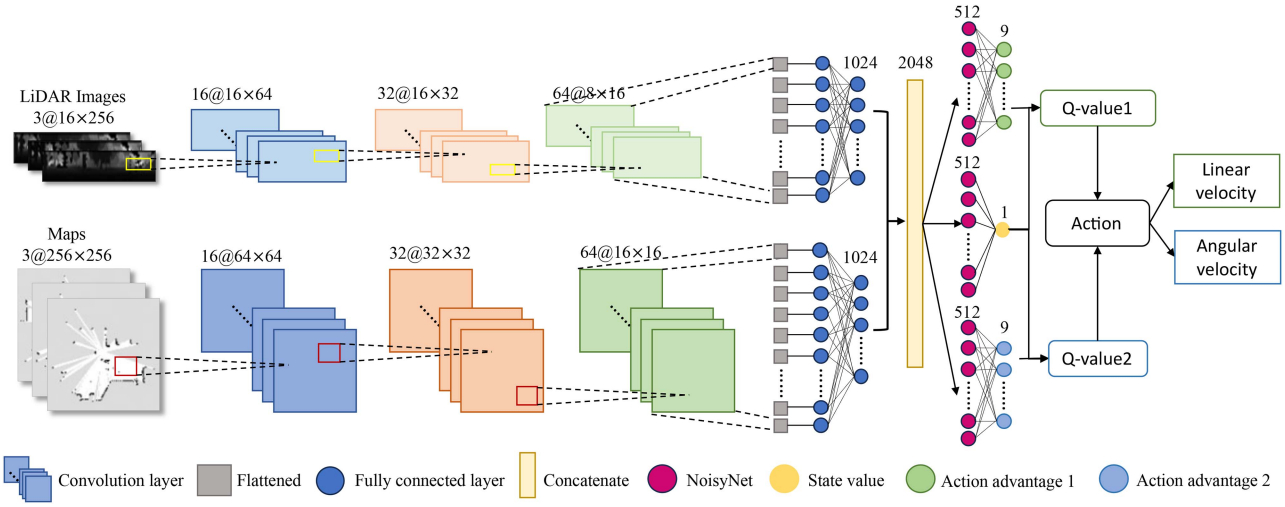

Fig. 2. The network architecture of DBD3QNN. The robot's state contains a series of depth images encoded from the LiDAR point cloud and occupancy grid maps. Features are extracted from these two inputs using separate CNN branches. The network then divides into three streams and maps the obtained feature representation to two advantage functions and a state value. The aggregation layers are subsequently used to calculate the Q-values. Both angular and linear control commands are simultaneously determined by using the estimated Q-values.

Fig. 3. Training environments.

branches are dedicated to estimating the advantage functions associated with angular and linear control commands. Specifically, to derive the state value, the 2048-D feature vector is processed through two NoisyNet layers comprising 512 neurons and 1 neuron, respectively. Similarly, to obtain action advantages related to linear actions, the identical feature vector is processed by two NoisyNet layers. The first layer consists of 512 neurons, and the second layer has N units, wherein N represents the amount of discretized actions. Likewise, additional NoisyNet layers with the same configuration are applied to map the extracted features to action advantages related to angular velocities.

IV. Experiments

The experimental results are presented in this section. To evaluate the effect of each component of the proposed network on its performance, we first conduct an ablation study. Then, we validate our approach through simulation and real-world experiments.

A. Training in Simulation

Setup: A large-scale 85 m $\times$ 32 m training environment with a variety of obstacles is built in Gazebo [47] as illustrated in Fig. 3. A simulated Scout-mini robot is furnished with a VLP-16 LiDAR that has an 80 m sensor range. Furthermore, the Robot Operating System (ROS) [48] is used as the robot communication and control middleware layer. TensorFlow [49] is utilized to train our network model. The computer outfitted with an Intel i9-10900K CPU, 32 GB RAM, and NVIDIA GeForce RTX 2080 Ti is employed for training and testing in simulation.

Ablation Study: Firstly, we conduct the training of Dual-Branch D3QN with NoisyNet (DBD3QNN) without the implementation of the β-consistency strategy, relying solely on extrinsic rewards. Subsequently, the β-consistency strategy is incorporated, resulting in a modified model referred to as DBD3QNN with β-Consistency. During the training process, the threshold β gradually increases from 0.001 to 0.05 over $2e5$ training iterations. Building upon this, we further train a variant of the network, denoted as DBD3QNN-RND with β-Consistency, by integrating the RND reward alongside the extrinsic reward.

For every training episode, the robot is arbitrarily oriented and starts at a random collision-free point in the training environment. All models are trained by Adam optimizer [50] with batch size $= 64$, the learning rate $= 10^{-4}$. The maximum training iteration is $4e5$ with the maximum episode length $= 500$ and the discount factor $\gamma = 0.99$. An episode ends when a collision occurs, the robot reaches the maximum step numbers, or the exploration rate $\rho > 0.9$. Each model undergoes an evaluation every 2000 iterations during the training process. The evaluation involves computing the average episodic reward over 5 episodes, where an episodic reward represents the cumulative extrinsic rewards obtained within a single episode. Every model undergoes three training sessions from scratch utilizing distinct random seeds to guarantee reliable statistical investigation. Table. I shows the hyperparameters setting.

Fig. 4 demonstrates the average episodic rewards achieved by all the models. It is evident that the rewards progressively increase during the training episodes and eventually converge

TABLE I
HYPERPARAMETERS

Hyperparameters	value
Batch size	64
Episode length	500
Training iteration	400,000
Discount factor γ	0.99
Learning rate	0.0001
NoisyNets initialization constant σ_0	0.5
Weighting factors α_1, α_2, α_3	0.4, 0.4, 0.2
Scaling factors λ_1, λ_2, λ_3	0.5, 0.02, 1.0

Fig. 4. The average episode rewards during training.

TABLE II
TRAINING RESULTS FOR DIFFERENT MODELS

model	average reward	average explored area (m^2)
DBD3QNN	319.6	1453
DBD3QNN w/ β-consistency	374.5	1774
DBD3QNN-RND w/o β-consistency	394.8	1976
DBD3QNN-RND w/ β-consistency	416.0	2112

to stable values. Comparing the results, the DBD3QNN model with the β-consistency strategy exhibits higher average episodic rewards with moderate fluctuations, demonstrating improved performance and enhanced training stability in comparison to the standard DBD3QNN. Similarly, the DBD3QNN-RND model with the β-consistency strategy also shows higher average episodic rewards compared to the standard DBD3QNN-RND, further affirming the benefits of this strategic approach. In addition, the proposed model demonstrates faster convergence and achieves even greater rewards by incorporating the RND-based intrinsic reward mechanism. These findings emphasize the effectiveness and potential of the proposed network in enhancing both the performance and efficiency of the training process.

Table. II illustrates the average rewards and the average explored areas of the training process during the last 1000 training iterations. It is noteworthy that the calculation of average explored areas does not take into account the condition

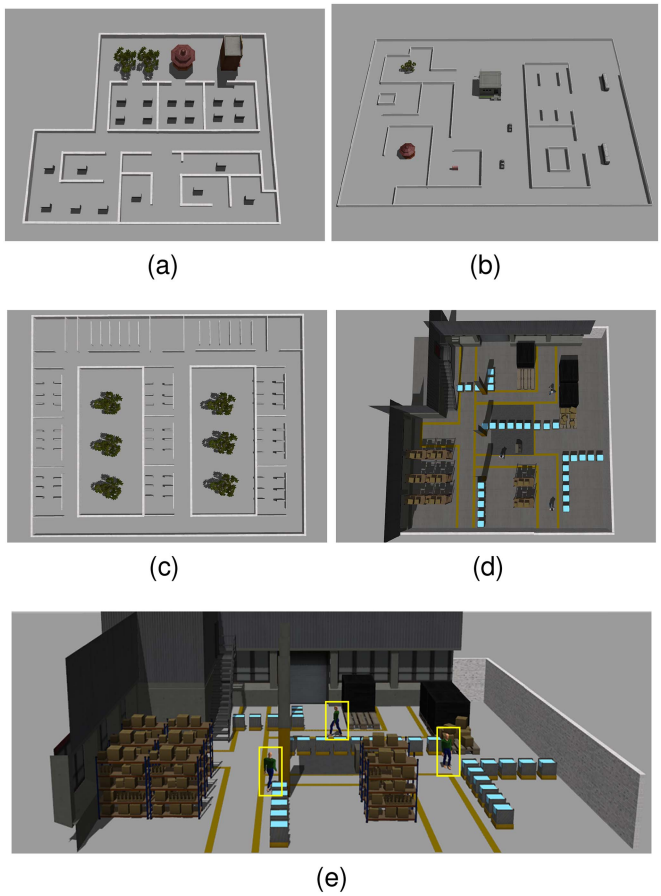

Fig. 5. Testing environments. (a) test env-1 ($85m \times 64m$), (b) test env-2($160m \times 120m$), (c) test env-3 ($200m \times 182m$), (d) test env-4 ($22m \times 22m$), and (e) walking people in test env-4.

where an episode ends upon the exploration rate exceeding 0.9. This procedure is carried out to derive the final explored areas. DBD3QNN-RND with β-consistency strategy model demonstrates accelerated convergence to the maximum average reward, completing the training procedure within the fewest number of steps. In the training environment, which has an explorable area of 2200 m^2, the robot's successful run is achieved when the ratio of explored regions on the map exceeds 0.9. Notably, only the DBD3QNN-RND with β-consistency strategy model achieves this desired level of exploration and ultimately succeeds. The ablation study indicates that all components play a critical role in the exploration's success.

B. Evaluation in Simulation

As shown in Fig. 5, four simulated environments are created to evaluate the performance of our exploration system. The fourth test environment is dynamic, with walking people moving at 0.7 m/s. We contrast our system with SOTA approaches (Nearest Frontier [2], TARE [27], FAEL [28]) with regard to exploration time, trajectory length, and success rate as illustrated in Table III. Achieving a successful run entails the robot meeting the termination condition: exploring over 90% of unknown areas without collision. The success rate is the percentage of robot runs that meet the termination condition within 25 runs, while

TABLE III
EVALUATION RESULTS IN SIMULATION ENVIRONMENTS

	exploration time(s)		trajectory length (m)		suceess rate (%)
	agv.	std.	agv.	std.	
test-env1					
Nearest frontier	>1500	—	>301.7	—	0
TARE	442.9	35.4	777.2	71.1	100
FAEL	457.0	34.7	561.2	34.4	96
Ours	475.4	31.0	448.2	29.3	88
test-env2					
Nearest frontier	>2200	—	>926.0	—	0
TARE	916.3	44.3	1737.0	60.2	96
FAEL	865.0	67.2	1525.5	58.3	76
Ours	574.0	32.7	626.7	34.6	92
test-env3					
Nearest frontier	>2000	—	>469.6	—	0
TARE	488.3	38.5	727.1	43.8	96
FAEL	349.3	20.3	446.4	21.7	88
Ours	445.8	33.1	435.4	29.8	68
test-env4					
Nearest frontier	252.31	41.5	131.7	19.8	64
TARE	141.1	14.78	143.0	30.7	72
FAEL	260.2	23.5	94.4	14.7	72
Ours	203.8	9.7	91.3	12.84	88

the average trajectory length and exploration time are calculated from successful runs. All methods are tested on the same desktop with Ubuntu 18.04 and ROS Melodic system.

As demonstrated by Table III, the nearest frontier method fails to achieve a successful run within the time limit in the first three expansive simulation environments. This is because it greedily selects frontiers and applies the A^* algorithm to determine the exploration path, which is prone to getting trapped in the local area. Our approach shows lower success rates in test env-1 and env-3 compared to TARE and FAEL, which are better suited for such environments due to their ability to detect and select viewpoints in free space and navigate through cluttered areas. Nonetheless, compared to the FAEL method, our approach achieves a higher success rate in test env-2, which is larger and less dense. This is because our proposed method fully leverages the 80 m LiDAR range, whereas FAEL only focuses on seeking frontiers within 12 m to mitigate computation costs. The average speed of our approach is slower than the TARE and FAEL methods in test env-1, env-2, and env-3. As a result, for the same travel distance, our method takes longer to explore the environment. This difference can be attributed to our use of a larger negative collision penalty in the reward function. This penalty prioritizes safety by encouraging the robot to move cautiously, leading to a slower average speed. In contrast to SOTA approaches, our proposed system consistently achieves the shortest trajectory lengths across all test environments. The nearest frontier and TARE methods neglect information gain, potentially leading to the robot ignoring high-gain areas initially and necessitating return visits. While the FAEL method considers the information gain for all viewpoints, it encounters revisits to the same sites due to equivalent gains between these viewpoints. In addition, our approach displays superior performance compared to standard

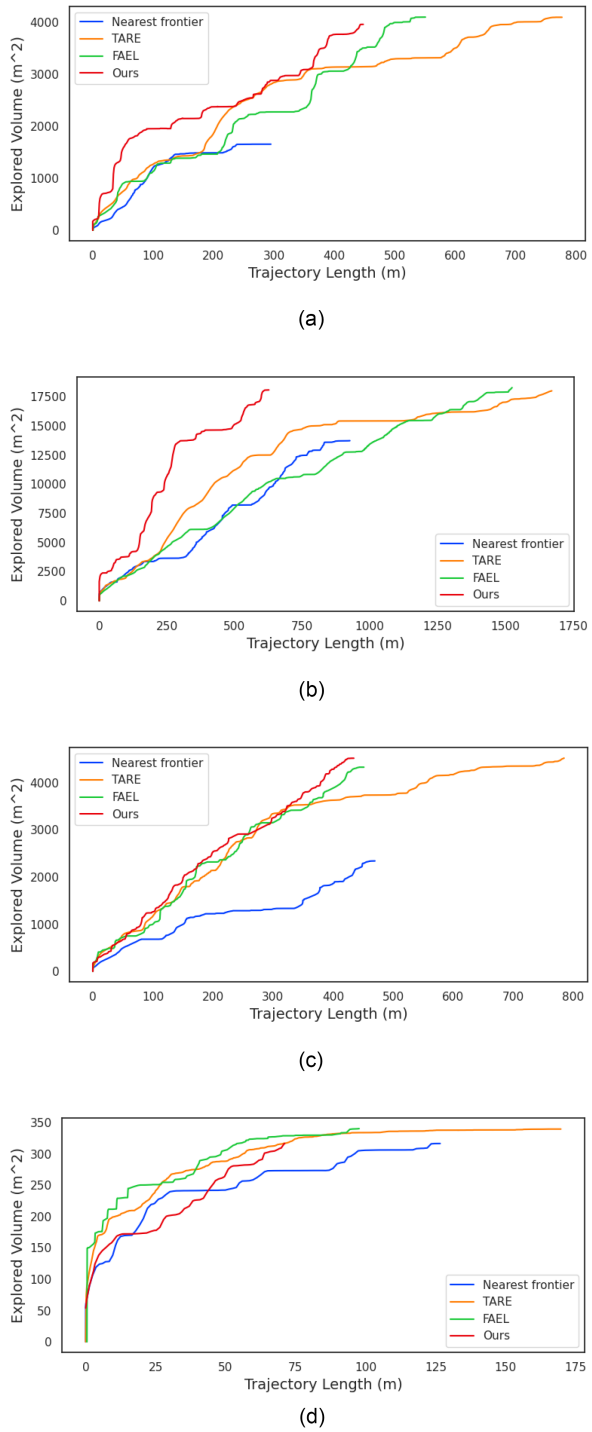

Fig. 6. The results show explored areas *vs.* trajectory length. The explorable area is all areas within the environment that are visible to the robot. (a) test env-1 (explorable area: 4146 m^2), (b) test env-2 (explorable area: 18215 m^2), (c) test env-3 (explorable area: 4582 m^2), and (d) test env-4 (explorable area: 351 m^2).

benchmark methods, particularly in attaining higher success rates within dynamic simulation environments. Both nearest frontier and FAEL methods relying on the A^* algorithm for local path planning pose a risk of collision in dynamically changing environments where frequent path recalculations are required. The path plan for the TARE approach updates at 1 Hz,

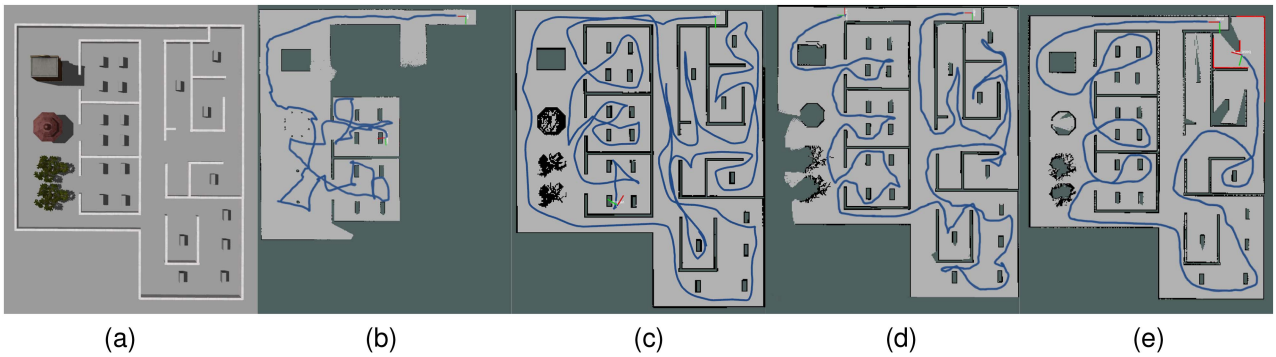

Fig. 7. The visual comparison of trajectories and occupancy grid maps in test env-1. (a) test env-1, (b) nearest frontier, (c) TARE, (d) FAEL, and (e) ours.

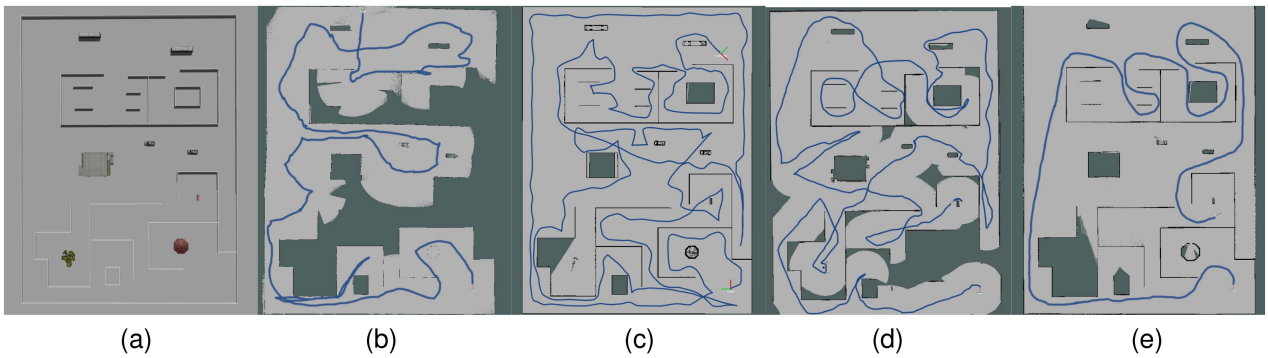

Fig. 8. The visual comparison of trajectories and occupancy grid maps in test env-2. (a) test env-2, (b) nearest frontier, (c) TARE, (d) FAEL, and (e) ours.

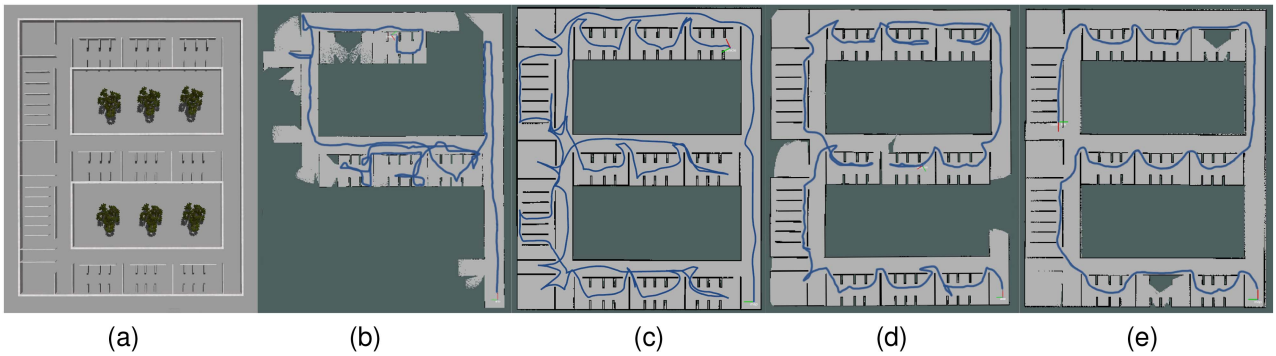

Fig. 9. The visual comparison of trajectories and occupancy maps in test env-3. (a) test env-3, (b) nearest frontier, (c) TARE, (d) FAEL, and (e) ours.

which increases the potential risk of collision with fast-moving obstacles. Conversely, our method's operation at 10 Hz, aligning with the LiDAR update frequency, facilitates real-time obstacle avoidance. To evaluate the computation cost, we calculate the average run time for our proposed method compared to the TARE and FAEL methods. The average run time for our method is 0.077 s, while the TARE approach has an average runtime of 0.716 s and FAEL is 0.041 s. This means our method achieves approximately a 10x speedup in runtime compared to TARE. Although FAEL has a slightly faster average runtime than our approach, it's important to note that the FAEL runtime grows exponentially as the LiDAR range is increased from 12 m to 80 m. This increased computational overhead could potentially overload the mobile system's computing capacity. In contrast, our A-SLAM system maintains a good exploration rate, the shortest movement distance, and low computational requirements, all within a relatively short timeframe. To better demonstrate the superiority of our framework, Fig. 6 shows the results of the exploration area over trajectory length for each method, underscoring our approach's superior exploration efficiency. Furthermore, Figs. 7–10 depict examples of experiments in four distinct test environments. According to the results, the proposed system plans more efficient trajectories and avoids unnecessary movements.

C. Evaluation in Real-World Environments

A Scout-mini robot equipped with an RSLidar Helios-16P and an Intel NUC mini-PC with i7-8559 U CPU is utilized as the robot platform in our experiments, as shown in Fig. 11. The

Fig. 10. The visual comparison of trajectories and occupancy maps in test env-4. (a) test env-4, (b) nearest frontier, (c) TARE, (d) FAEL, and (e) ours.

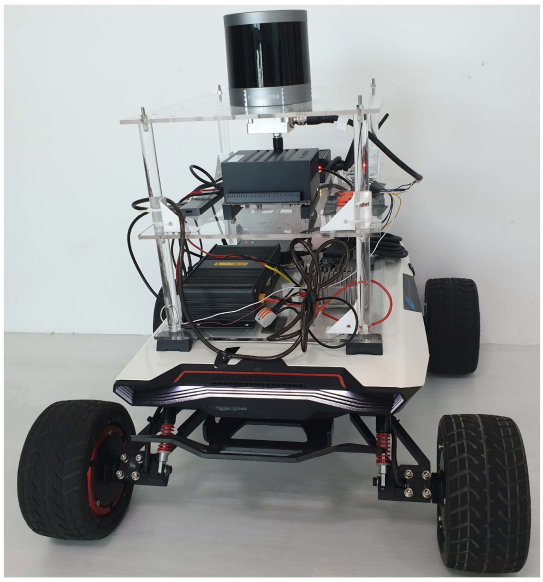

Fig. 11. Scout-mini robot equipped with LiDAR.

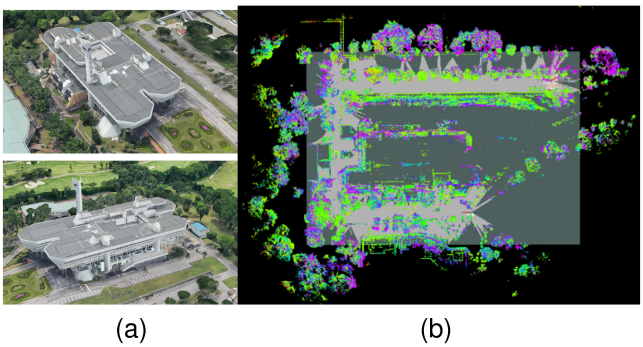

Fig. 12. The result of the real-world experiments undertaken within the large-scale open environment. (a) The area to be explored. (b) The explored area.

motion commands are updated every 0.1 s. The experiments are conducted in several scenarios: (a) an open environment with complex terrain and roads without guardrails; (b) a large landscape sky terrace featuring vegetation; (c) a crowded environment with moving people; (d) a large-scale mixed-use development with a canopy cover. Notably, the trained model is adopted in real-world experiments requiring no additional fine-tuning.

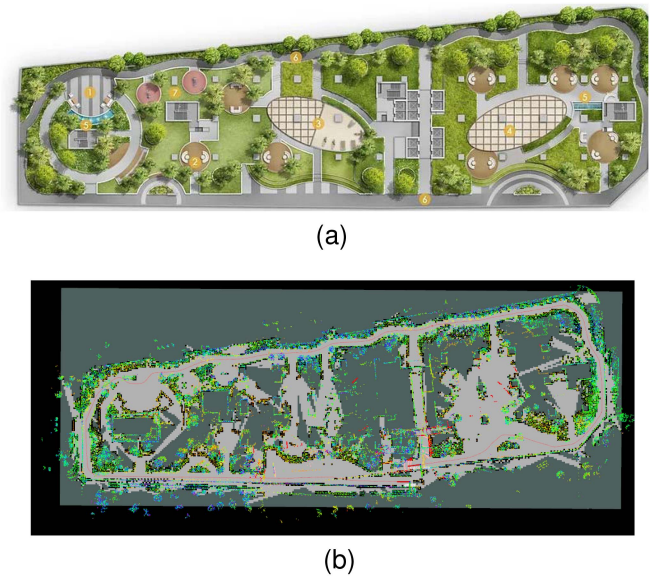

Fig. 13. The result of the real-world experiments undertaken within the large landscape sky terrace. (a) The area to be explored. (b) The explored area.

In the beginning, our system is developed in an expansive open environment. Fig. 12 illustrates the reconstructed environment, including a 3D Octomap and 2D occupancy grid map. The robot explores over 5000 m^2 in 321 seconds and moves 448 meters at an average velocity of 1.4 m/s. Furthermore, the model is evaluated in a large landscape sky terrace as illustrated in Fig. 13. The robot explores about 2000 m^2 in 280 s. During this exploration, the robot travels a distance of 281 meters, maintaining an average velocity of 1.0 m/s, which indicates that the proposed A-SLAM system is capable of efficiently navigating unknown environments, even in regions that are not easily accessible. Additionally, the proposed system is assessed in a challenging scenario with moving individuals. To guarantee pedestrians' safety, the robot's maximum speed is configured to be 0.8 m/s. As the walker approaches the robot, the robot decelerates its speed and adjusts its orientation by using the higher angular velocity to prevent any potential collision with the walker, as shown in Fig. 14. Finally, the A-SLAM system is evaluated in a 35,000 m^2 large-scale mixed-use development with a canopy cover as shown in Fig. 15. In this real-world setting, the robot navigates a total distance of 957 m at an average speed of 1.35 m/s, which demonstrates the capability of

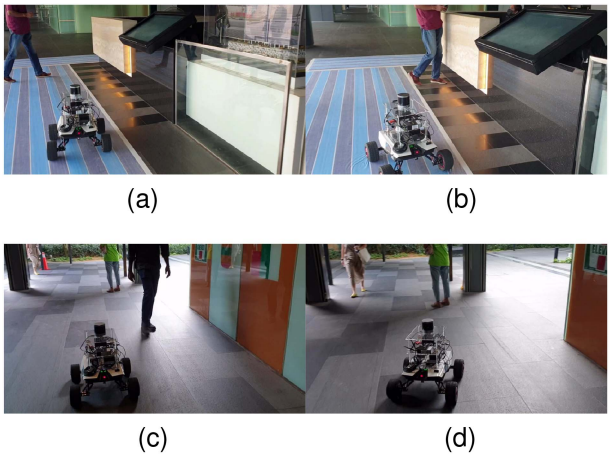

Fig. 14. Intermediate steps in the crowded environment. The subcaption includes the received linear and angular velocities. (a) $-\pi/12$ rad/s, 0.2 m/s, (b) $\pi/4$ rad/s, 0.2 m/s, (c) $\pi/12$ rad/s, 0.1 m/s, and (d) $-\pi/4$ rad/s, 0.2 m/s

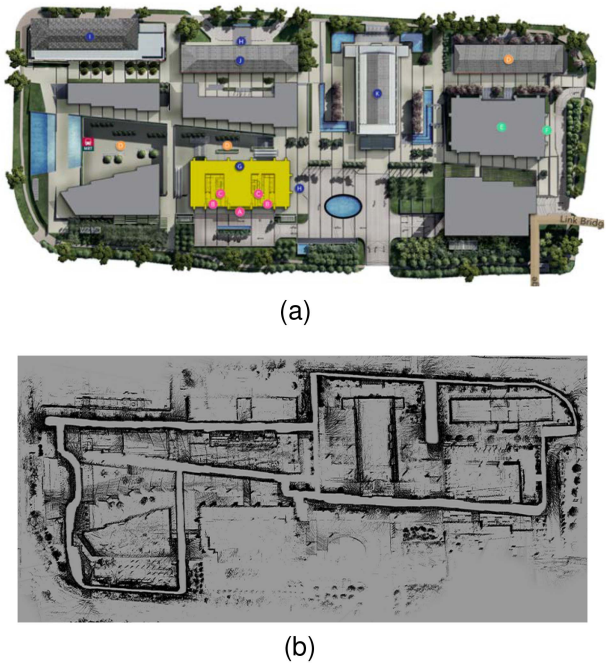

Fig. 15. The result of the real-world experiments undertaken within the large-scale mixed-use development. (a) The area to be explored. (b) The explored area.

our A-SLAM system to effectively explore and map large-scale unknown environments. A video of experiments is available at https://www.youtube.com/watch?v=lXXOuViXp24.

V. Conclusion

This paper presents an end-to-end LiDAR-based Active SLAM system by DRL, to realize efficient autonomous robot exploration. Specifically, we first propose a LiDAR data encoder to convert the original sparse and unordered point clouds to an image representation. LiDAR images and occupancy grid maps are fed into the CNN streams. Afterward, the extracted features are merged and passed through distinct branches of the NoisyNet architecture to generate two sets of Q-value vectors, enabling the simultaneous determination of angular and linear velocities. Furthermore, we adopt the β-consistency action selection approach, which considers the angular velocity's consistency during the decision-making procedure. To enhance exploration capability, we also incorporate RND bonuses as intrinsic rewards. The experimental results reveal that our system outperforms SOTA methods, with regard to episode rewards, success rates, and, notably, its obstacle avoidance proficiency. Furthermore, the proposed approach can be seamlessly transferred from the simulator to intricate real-world environments without requiring specific fine-tuning, demonstrating its impressive generalization capability.

In our future work, we aim to further validate the robustness and efficiency of our method in real-world environments by conducting more experiments and comparisons with benchmark methods. Simultaneously, we plan to optimize both the algorithm and hardware to address more complex exploration scenarios. Additionally, we intend to integrate an active loop-closing strategy into our exploration approach to reduce the robot's pose uncertainty and attain more precise mapping results.

References

[1] J. A. Placed et al., "A survey on active simultaneous localization and mapping: State of the art and new frontiers," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1686–1705, Jun. 2023.
[2] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Automat.*, 1997, pp. 146–151.
[3] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *Proc. IEEE ISR 41st Int. Symp. Robot., 6th German Conf. Robot. VDE*, 2010, pp. 1–8.
[4] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," *Adv. Robot.*, vol. 27, no. 6, pp. 459–468, 2013.
[5] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2135–2142.
[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
[7] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon"next-best-view" planner for 3D exploration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1462–1468.
[8] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *J. Field Robot.*, vol. 37, no. 8, pp. 1363–1388, 2020.
[9] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, "Information based adaptive robotic exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, pp. 540–545.
[10] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," *Robot. Sci. Syst.*, vol. 2, pp. 65–72, 2005.
[11] B. J. Julian, S. Karaman, and D. Rus, "On mutual information-based control of range sensing robots for mapping applications," *Int. J. Robot. Res.*, vol. 33, no. 10, pp. 1375–1392, 2014.
[12] W. Tabib, M. Corah, N. Michael, and R. Whittaker, "Computationally efficient information-theoretic exploration of pits and caves," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 3722–3727.
[13] S. Bai, J. Wang, F. Chen, and B. Englot, "Information-theoretic exploration with Bayesian optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1816–1822.
[14] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
[15] M. G. Jadidi, J. V. Miró, R. Valencia, and J. Andrade-Cetto, "Exploration on continuous Gaussian process frontier maps," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 6077–6082.

[16] M. G. Jadidi, J. V. Miro, and G. Dissanayake, "Mutual information-based exploration on continuous occupancy maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 6086–6092.
[17] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[18] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2020.
[19] F. Chen, S. Bai, T. Shan, and B. Englot, "Self-learning exploration and mapping for mobile robots via deep reinforcement learning," *Aiaa Scitech Forum*, 2019, Art. no. 0396.
[20] F. Chen et al., "Zero-shot reinforcement learning on graphs for autonomous exploration under uncertainty," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 5193–5199.
[21] Y. Cao, T. Hou, Y. Wang, X. Yi, and G. Sartoretti, "ARiADNE: A reinforcement learning approach using attention-based deep networks for exploration," in *Proc. IEEE Int. Conf. Robot. Automat.*, London, U.K., 2023, pp. 10219–10225.
[22] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.
[23] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," in *Proc. 7th Int. Conf. Learn. Representations*, 2019, pp. 1–17.
[24] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty-aware receding horizon exploration and mapping using aerial robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4568–4575.
[25] T. Dang, C. Papachristos, and K. Alexis, "Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2526–2533.
[26] S. Bai, J. Wang, K. Doherty, and B. Englot, "Inference-enabled information-theoretic exploration of continuous action spaces," *Robot. Res.*, vol. 2, pp. 419–433, 2018.
[27] C. Cao, H. Zhu, H. Choset, and J. Zhang, "TARE: A hierarchical framework for efficiently exploring complex 3D environments," *Robot. Sci. Syst.*, vol. 5, 2021.
[28] J. Huang et al., "FAEL: Fast autonomous exploration for large-scale environments with a mobile robot," *IEEE Robot. Automat. Lett.*, vol. 8, no. 3, pp. 1667–1674, Mar. 2023.
[29] L. Tai and M. Liu, "Towards cognitive exploration through deep reinforcement learning for mobile robots," 2016, *arXiv:1610.01733*.
[30] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.
[31] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.
[32] M. Fortunato et al., "Noisy networks for exploration," 2017, *arXiv:1706.10295*.
[33] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, pp. 4131–4138.
[34] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning," 2018, *arXiv:1804.00456*.
[35] R. Cimurs, I. H. Suh, and J. H. Lee, "Goal-driven autonomous exploration through deep reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 730–737, Apr. 2022.
[36] H. Jiang et al., "ITD3-CLN: Learn to navigate in dynamic scene through deep reinforcement learning," *Neurocomputing*, vol. 503, pp. 118–128, 2022.
[37] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[38] Z. Zheng, C. Cao, and J. Pan, "A hierarchical approach for mobile robot exploration in pedestrian crowd," *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 175–182, Jan. 2022.
[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[40] K. Wu, H. Wang, M. Abolfazli Esfahani, and S. Yuan, "BND*-DDQN: Learn to steer autonomously through deep reinforcement learning," *IEEE Trans. Cogn. Develop. Syst.*, vol. 13, no. 2, pp. 249–261, Jun. 2021.
[41] T. Haarnoja et al., "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.
[42] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5671–5677.
[43] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[44] J. Chen, H. Wang, M. Hu, and P. N. Suganthan, "Versatile LiDAR-inertial odometry with SE (2) constraints for ground vehicles," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3486–3493, Jun. 2023.
[45] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," *Robot. Sci. Syst.*, vol. 2, no. 9, pp. 1–9, 2014.
[46] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1907–1915.
[47] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IEEE Cat. No 04CH37566)*, 2004, vol. 3, pp. 2149–2154.
[48] M. Quigley et al., "ROS: An open-source robot operating system," in *Proc. IEEE Int. Conf. Robot. Automat. Workshop Open Source Softw.*, 2009, vol. 3, p. 5.
[49] M. Abadi et al., "*TensorFlow*: A system for *Large-Scale* machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Des. Implementation*, 2016, pp. 265–283.
[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

Jiaying Chen received the B.Eng. degree from the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, where she is currently working toward the Ph.D. degree in electrical and electronics engineering. Her research interests include indoor positioning system, simultaneous localization and mapping (SLAM), loop closure detection, and active SLAM.

Keyu Wu received the B.Eng. degree from the National University of Singapore, Singapore, and the Ph.D. degree from Nanyang Technological University, Singapore. She is currently a Scientist with the Institute for Infocomm Research, A*STAR, Singapore. Her research interests include reinforcement learning, transfer learning, deep learning, autonomous navigation, path planning, and trajectory generation.

Minghui Hu received the MSc. degree in electric and electrical engineering in 2019 from Nanyang Technological University (NTU), Singapore, where he is currently working toward the Ph.D. degree with the School of Electrical and Electronic Engineering. He is a Research Scientist with Temasek Lab, NTU. His research interests include generative models and multi-modality learning.

Ponnuthurai Nagaratnam Suganthan (Fellow, IEEE) received the B.A degree, Postgraduate Certificate, and the M.A degrees in electrical and information engineering from the University of Cambridge, Cambridge, U.K., in 1990, 1992 and 1994, respectively, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He was a Predoctoral Research Assistant with the Department of Electrical Engineering, University of Sydney, Sydney, NSW, Australia, during 1995–1996, and a Lecturer with the Department of Computer Science and Electrical Engineering, University of Queensland, Brisbane, QLD, Australia, during 1996–1999. In 1999, he joined the School of Electrical and Electronic Engineering, Nanyang Technological University, as an Assistant Professor, then he became an Associate Professor. Since August 2022, he has been with the KINDI Center for Computing Research, Qatar University, Doha, Qatar, as a Research Professor. His research interests include evolutionary computation, pattern recognition, bioinformatics, biometrics, multiobjective evolutionary algorithms, applications of evolutionary computation, and neural networks. He is an Associate Editor for IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *Pattern Recognition Journal*.

Anamitra Makur received the B.Tech. degree from Indian Institute of Technology, Kharagpur, India, and the M.S. and Ph.D. degrees from the California Institute of Technology, Pasadena, CA, USA. He was with the Indian Institute of Science, Bangalore, India, till 2002 in various capacities, the most recent being a Professor. He also held visiting positions with University of California at Santa Barbara, Santa Barbara, CA, the University of Kaiserslautern, Kaiserslautern, Germany, Griffith University, Brisbane, QLD, Australia, and Keio University, Yokohama, Japan. Since 2002, he has been an Associate Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where he is currently the Deputy Head of the Division of Information Engineering. His research interests include multirate signal processing, signal/image/video compression, and image processing. He was the recipient of several awards, notable being the 1998 Young Engineer award from the Indian National Academy of Engineering, and the best paper award in the IEEE APCCAS 2006 conference.