IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 16 May 2024, accepted 8 July 2024, date of publication 16 July 2024, date of current version 29 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3429230

RESEARCH ARTICLE

Cooperative Deep Reinforcement Learning Policies for Autonomous Navigation in Complex Environments

VAN MANH TRAN AND GON-WOO KIM, (Member, IEEE)
Department of Intelligent Systems and Robotics, Chungbuk National University, Cheongju 28644, South Korea

Corresponding author: Gon-Woo Kim (gwkim@cbnu.ac.kr)

This work was supported in part by the Innovative Human Resource Development for Local Intellectualization Program through the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (MSIT), under Grant IITP-2024-2020-0-01462, 50%; and in part by the Technology Innovation Program (or Industrial Strategic Technology Development Program-ATC+) (Development of Service Robot Core Technology that can Provide Advanced Service in Real Life) funded by the Ministry of Trade, Industry and Energy (MOTIE, South Korea), under Grant 20009546.

ABSTRACT A critical part of achieving robust and safe navigation for mobile robots is selecting the right navigation policies trained through simulation to operate effectively in real-world situations. Simulation-trained policies often struggle for mobile robot settings deployed in real-world navigation tasks, leading to policy degradation and increased risk manners. To address these challenges, a cooperative deep reinforcement learning policies (CDRL) framework is proposed, ensuring safe exploration and deployment in unknown complex environments. The CDRL framework cooperates with exploration and exploitation policies based on a policy-switching mechanism, which efficiently helps the robot escape the local optima. Instead of transferring a single navigation policy, CDRL leverages cooperative navigation policies with diverse reward functions, enabling them to adapt to unknown complex environments. The proposed technique is based on an exploration distributional soft actor critic (E-DSAC) and soft actor critic (SAC) algorithms, which enhances training efficiency. The deep reinforcement learning (deep RL) models in this framework are represented by a mobile service robot that reaches target positions without requiring a map presentation. Experimental results show that the proposed framework is proven to have safe and fast motions in terms of navigation time and success rates. The sim-to-real transfer process of mobile service robots can be found (https://youtu.be/vIxRqXidKIM).

INDEX TERMS Autonomous navigation, sim-to-real transfer, soft actor critic, distributional reinforcement learning, service robot.

I. INTRODUCTION

Mobile service robots have gained popularity in modern life, where they are utilized widely in handling a wide spectrum of tasks from simplicity to superior human-level decision-making. A fundamental challenge for these robots is achieving autonomous navigation, particularly within intricate indoor environments like museums, shopping malls, and offices. To ensure safe and efficient navigation in such real-world scenarios, in-depth research on autonomous navigation technologies is crucial for service robots [1].

A method known as map-based navigation, used for autonomous navigation in specific environments where a predefined map has been deployed, is well-established [2]. Various technologies and algorithms are integrated into the map-based navigation system, including path planning, Simultaneous Localization and Mapping (SLAM), and control [3]. However, the integrated system carries inherent high potential risks, such as low map reliability in dynamic environments, accumulative errors, and the demand for

The associate editor coordinating the review of this manuscript and approving it for publication was Laura Celentano.

intensive expert knowledge in integration, hyperparameter-tuning, and implementation [4], [5]. Unlike conventional map-based navigation, map-free navigation using deep RL is a promising alternative that overcomes these shortcomings associated with map dependence [6], [7].

Mappless navigation for mobile robots using deep RL involves training a policy in simulated environments and then deploying a policy in real-world scenarios. Instead of directly training the robot system in real environments, a deep RL model is commonly trained within virtual environments, offering advantages in training time, efficiency, and safety. However, the policy may fail to transfer into real-world scenarios due to the inherent disparity between simulation data and real-world environments (sim-to-real gap) or the agent learning ability through reinforcement learning algorithms in simulation. To bring the reality gap, the randomization domain technique [8], [9] is utilized for randomized parameters during training, which is commonly deployed for transferring the deep RL model. On the other hand, recent mapless navigation studies (e.g SAC algorithm) [10], [11] leverage the backbone of the actor-critic algorithm with entropy regularization for training the agent in simulation, increasing transferability to real-world scenarios. Furthermore, distributional reinforcement learning [12], [13] also achieves significant advancement and has the potential to enhance the deep RL model for practical navigation tasks.

After in-depth investigations, a wide range of navigation studies above focus on single policy transfers to learn multiple navigation skills in environments, which connects to the multi-objective task. Meanwhile, a single learning method to transfer only a policy mastering all skills simultaneously remains challenging [14]. Despite advancements made in previous research, the challenge of the policy transfer from simulation to real-world environments persists for robot navigation missions. This necessitates the development of a new strategy to address the remaining limitations of this transferability issue.

To enhance the capabilities of the mapless navigation tasks, the learned policy takes advantage of experience from the training environment to operate in other environments, which exemplifies the exploitation of robots [15], [16], [17]. Nevertheless, new environments include dead ends (local optima) that may cause the robot to get stuck or trapped. Therefore, the robot needs an additional policy to explore its surroundings, enabling it to escape these dead ends. In terms of exploration, a variety of previous methods are used to increase learning ability during training, such as epsilon greedy [18], upper confidence bound [19], or Boltzmann exploration [20], and random network distillation (RND) [21]. In particular, RND has achieved success in exploration-demanding environments. In RND, a curiosity measure proves to be a valuable technique for designing the intrinsic reward for mobile robot navigation [22], [23]. Nonetheless, exploration strategies might help the robot avoid dead ends in unfamiliar situations, but this might not lead it to the target quickly. Hence, after escaping the stuck areas, the robot could reach targets directly by using robot exploitation experiences. For real-world navigation tasks, the robot system should ideally exist in exploration and exploitation modes for maximum efficiency. One of the interesting strategies is hybrid hierarchical reinforcement learning [24], which combines the exploration and exploitation modes into hierarchical structures. However, this method should be taken into account for continuous action space while continuously training in new environments. Therefore, it is essential to create a mechanism that seamlessly combines exploration and exploitation policies to avoid constant retraining when realistic environments become larger and more complex.

In light of the aforementioned benefits and limitations, a hybrid policy approach is proposed, the cooperative deep reinforcement learning policies framework to address mapless navigation challenges. Therefore, the technique utilizes distributional soft actor-critic (DSAC) combined with the intrinsic reward [21] encourages exploration to train the exploration policy, which is called the E-DSAC algorithm. The contributions of this paper are summarized as follows:

- The cooperative deep reinforcement learning policies framework combines exploration and exploitation policies, which not only prevent retraining as realistic environments grow in size and complexity but also improve the overall navigation performance of sim-to-real transfer.
- This work presents a training strategy that integrates a distributional value function with intrinsic reward. Building upon the DSAC algorithm [25], the intrinsic reward is designed by matching the loss between the random target network and the predictor network to enhance the exploration policy's curiosity within complex navigation tasks.
- The CDRL framework utilizes cooperative policies transferred simultaneously from simulation with diverse navigation skills in unknown environment structures, ensuring the framework's applicability in realistic environments when deployed on physical robots operating in unknown real-world environments.

The paper begins by describing related work in Section II, and then the original work including SAC and DSAC algorithms is presented in Section III. After that, Section IV presents the proposed framework, including the design of distinct rewards for each policy and the mechanism for transitioning between exploration and exploitation based on the local optima detection algorithm. Section V details the training process, employing the SAC algorithm to train the exploitation policy and the E-DSAC algorithm to train the exploration policy. The results of experiments conducted in both simulated and real-world environments are presented in Section VI. Finally, Section VII provides concluding remarks for this study.

II. RELATED WORK

A. SENSOR-BASED NAVIGATION BASED ON DEEP RL

The actor-critic method has been widely utilized across various fields of robotics, including mapless navigation of mobile robots [26], [27]. For instance, Fan et al. [6] proposed the deep RL model with low-dimensional laser ranges to observe local spaces and navigate to the target relative to the mobile robot coordinate. Marchesini and Farinelli [28] dealt with the mapless navigation problem by optimizing deep deterministic policy gradient (DDPG) and proximal policy optimization (PPO) algorithms. The model-free RL algorithms continue to develop significantly as the inclusion of the entropy term in the objective function eases training complexity and enhances exploration compared to DDPG or PPO [17], [29]. Besides, training deep RL agents is an indispensable procedure in transferring policies into real-world scenarios. The majority of studies are centered on the improvement of navigation performance during the training phase [30], [31]. However, the policy transferring finds it difficult to evaluate unfamiliar environments, particularly in real-world environments [32] like unstructured environments or corridor environments. To overcome the above issues, the sparse and dense reward functions combined with the distributional value function are designed appropriately for navigation policies. Although the reward shaping technique is not new within the realm of reinforcement learning (RL), its application, in conjunction with distributional RL, represents an underexplored avenue to achieve exceptional learning capabilities in mapless navigation.

B. DISTRIBUTIONAL REINFORCEMENT LEARNING FOR MOBILE ROBOT NAVIGATION

Recent advances in distributional reinforcement learning have achieved state-of-the-art performance in arcade game environments [33], [34] and robotic benchmarks [13], [35], [36]. An exemplary instance of this is the combination of distributional RL and SAC, resulting in significant advancement in risk-sensitive control tasks. For distributional RL, Rezaee et al. [37] estimated the distribution of stochastic outcomes to handle uncertainty for motion planning. Choi et al. [38] designed a distributional RL agent to learn an uncertainty-aware policy, which reduces cost-of-collision compared with conventional RL. By taking advantage of a distributional framework, Liu et al. [13] utilized the value of conditional value at risk (CVaR) forecasting intrinsic uncertainty to perform drone autonomous navigation tasks. Those previous studies adjusted risk policy levels based on CVaR methods (e.g. α subset), which is difficult to optimize parameters in large environments [39]. Regularizing risk-sensitive policies in distributional RL poses a significant challenge due to the requirement of intensive expert knowledge under real practical, real-world scenarios that encompass both static obstacles and unexpected dynamic obstacles (e.g. human motions). It can be seen that the extensive expansion of the DSAC algorithm's application to mapless navigation for mobile robots is a key challenge that should be carefully taken into account. Therefore, instead of regularizing risk-sensitive policy, the deployment of multiple policies is a new proposed approach to solve autonomous navigation tasks in large complex environments.

III. PRELIMINARIES

A. PROBLEM FORMULATION

In mapless navigation populated with unknown obstacles, the robot requires a decision-making process to reach the target position. This framework must leverage the robot's internal state information alongside sensor data from its surrounding environment. The environment can be represented as a Markov decision process. At time $t = 1, 2, \ldots, T$, the robot chooses action a_t according to state s_t, and receives observed reward $r(s_t, a_t)$. The interaction between the robot and the environment can be represented by a tuple $(s_1, a_1, s_2, a_2, \ldots, s_T)$, where the terminal state s_T happens when the robot reaches the target or the distance from the obstacle is less than r_m. The objective in traditional RL is to find the optimal collision-free policy π^* that maximizes the expected cumulative return. The action value function Q^π can be represented as follows:

$$Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{1}$$

B. SOFT ACTOR CRITIC

The off-policy SAC algorithm is described according to the actor-critic framework [40] with the entropy [41]. For mapless navigation, the deep RL model is trained with an emphasis on entropy to facilitate the training process. For this, we choose SAC [42] as one of the algorithms to train mobile robots for mapless navigation. To maximize a trade-off between the expected return and entropy $\mathcal{H}(\pi(\cdot \mid s_t)) = -\int_{|A|} \pi(a \mid x_t) \log \pi(a \mid x_t)\, da$ simultaneously in continuous action space. The objective of the policy is:

$$J(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \left[R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot \mid s_t))\right]\right] \tag{2}$$

the factor $\alpha > 0$ regulates the impact of the entropy. In SAC, three networks are used, of which two critic Q networks, parameterized by φ_1 and φ_2, and the output of the actor network generates the action. The past state transitions (s_t, a_t, r, s', d) are stored in replay buffer $\mathcal{D}$. The loss function minimizes the mean squared bootstrapped estimate (MSBE):

$$\mathcal{L}(\varphi_{1,2}) = \frac{1}{|\mathcal{M}|} \sum_{(s_t, a, r, s_{t+1}, d) \in \mathcal{M}} \left(Q_{\varphi_{1,2}}(s_t, a_t) - \hat{y}(r, s_{t+1})\right)^2 \tag{3}$$

where $\mathcal{M}$ presents the mini-batch sampling from $\mathcal{D}$. The target $\hat{y}(r, s_{t+1})$ includes clip Q value and entropy term:

$$\hat{y}(r, s_{t+1}) = r + \gamma \Big(\min_{j=1,2} Q_{\hat{\varphi}_j}(s_{t+1}, \tilde{a}_{t+1}) - \alpha \log \pi_\phi(\tilde{a}_{t+1}|s_{t+1}) \Big) \quad (4)$$

where $\tilde{a}_{t+1} \sim \pi_\phi(\tilde{a}_{t+1}|s_{t+1})$ is sampled from current policy π_ϕ. The policy is a Gaussian distribution that has been re-parameterized, incorporating a squashing function to ensure actions fall within a specified range:

$$a_\phi(s_t, \xi_t) = tanh(\mu_\phi(s_t) + \sigma(s_t) \cdot \xi_t)) \quad (5)$$

where $\xi_t \sim \mathcal{N}(0, I)$ is independent sampled noise and $\mu_\phi(s_t)$ and $\sigma_\phi(s_t)$ are the mean and standard deviation of policy. The parameters of the policy are updated by maximizing the future return with entropy regularization.

$$\max_\phi \mathop{\mathbb{E}}_{s\sim\mathcal{D},\xi\sim\mathcal{N}} \Big(\min_{j=1,2} Q(s_t, a_\phi(s_t, \xi_t)) - \alpha \log \pi_\phi(a_\phi(s_t, \xi_t)) \mid s_t) \Big) \quad (6)$$

C. DISTRIBUTIONAL SOFT ACTOR CRITIC

Whereas traditional RL aims to maximize the expected cumulative rewards, distribution RL focuses on distributional information of value function [43], which is based on the distributional Bellman equation:

$$Z^\pi(s, a) \overset{D}{=} R(s, a) + \gamma Z^\pi(s', a') \quad (7)$$

where $U \overset{D}{=} V$ indicates that random variables U and V have identical distributions. Equation (8) defines a recursive relationship among three random variables $Z^\pi(s', a')$, $Z^\pi(s, a)$ and $R(s, a)$. The distributional Bellman operator, denoted by $\mathcal{T}_D^\pi$, can be expressed as follows:

$$\mathcal{T}_D^\pi Z(s_t, a_t) :\overset{D}{=} R(s, a) + \gamma Z^\pi(s', a') \quad (8)$$

By combining the distributional RL and entropy framework, distributional SAC [25] relies on a distribution over the returns, using quantile regression to estimate value distribution. Moreover, DSAC utilizes not only the random return $Z^\pi : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$, but also the soft random return based on SAC, given by:

$$Z^\pi(s, a) :\overset{D}{=} \sum_{t=0}^{\infty} \gamma^t \left[R(s_t, a_t) - \alpha \log \pi_\phi(a_{t+1} \mid s_{t+1}) \right] \quad (9)$$

For the two quantiles $\hat{\tau}_i$ and $\hat{\tau}_j$, the temporal difference error is achieved:

$$\delta_{ij}^t = r_t + \gamma \left[Z_{\bar{\theta}_{1,2}}^{\hat{\tau}_i}(s_{t+1}, a_{t+1}) - \alpha \log \pi_{\bar{\phi}}(a_{t+1} \mid s_{t+1}) \right] - Z_{\theta_{1,2}}^{\hat{\tau}_j}(s_t, a_t) \quad (10)$$

where $Z_{\theta_{1,2}}^{\hat{\tau}_j}$ is the critic network output, which is an estimate of the τ-quantile of $Z^\pi(s, a)$, and $Z_{\bar{\theta}_{1,2}}^{\hat{\tau}_i}$ is the target critic. $\bar{\theta}$ and $\bar{\phi}$ are target critic network and target policy network parameters. To train the critic, the quantile fractions $\tau_i, \tau_j \sim U([0, 1])$ are sampled independently to minimize the Huber quantile regression loss:

$$\rho_\tau^\lambda(\delta_{ij}) = \left|\tau - \mathbb{I}\{\delta_{ij} < 0\}\right| \frac{\mathcal{L}_\lambda(\delta_{ij})}{\lambda}$$
$$\mathcal{L}_\lambda(\delta_{ij}) = \begin{cases} \frac{1}{2}\delta_{ij}^2, & \text{if } |\delta_{ij}| \le \lambda \\ \lambda\left(|\delta_{ij}| - \frac{1}{2}\lambda\right), & \text{otherwise} \end{cases} \quad (11)$$

where $\mathbb{I}$ is the indicator function, $\rho_{\hat{\tau}_j}$ is weighted by the target distribution fractions $(\tau_{i+1} - \tau_i)$ and λ is a smooth coefficient for gradient-clipping. The critic network is trained to minimize the loss function:

$$\mathcal{L}_Z(\theta) = \sum_{i=0,j=0}^{N-1} (\tau_{i+1} - \tau_i)\, \rho_{\hat{\tau}_j}^\lambda\left(\delta_{ij}^t\right) \quad (12)$$

where N represented independent quantiles sampled for both target and local networks. The action value function is achieved by taking expectations as:

$$Q(s, a) = \frac{1}{N}\sum_{i=0}^{N-1}(\tau_{i+1} - \tau_i) \min_{k=1,2} Z_{\theta_k}^{\hat{\tau}_i}(s_t, a_t) \quad (13)$$

The loss function policy network is updated by using gradient descent:

$$\mathcal{L}_\pi(\phi) = \frac{1}{|\mathcal{M}|} \sum_{s_t\in\mathcal{M},\xi_t\sim\mathcal{N}} \Big(\min_{j=1,2} Q(s_t, a_\phi(s_t, \xi_t)) - \alpha \log \pi_\phi(a_\phi(s_t, \xi_t)) \mid s_t) \Big) \quad (14)$$

The target networks are updated with a smoothing factor ι:

$$\bar{\phi} \leftarrow \iota\phi + (1 - \iota)\bar{\phi} \quad (15)$$
$$\bar{\theta}_{1,2} \leftarrow \iota\theta_{1,2} + (1 - \iota)\bar{\theta}_{1,2} \quad (16)$$

IV. COOPERATIVE POLICIES TRANSFERRING FRAMEWORK

Mappless navigation presents significant challenges for mobile robots. The robot must simultaneously explore the unknown environment to avoid local optima (dead ends or corners) and move toward the goal without collision. Therefore, the CDRL framework is proposed to cooperate with policies that move the robot efficiently and safely in complicated environments. Figure 1 describes an end-to-end navigation system to take laser data, robot pose, and goal position as the system input. With pre-defined goal information, the framework processes laser scan data to create observation states for navigation policies. The exploration and exploitation policies are switched dynamically to compute a sequence of commands that navigate the robot to the goal in unaware environments.

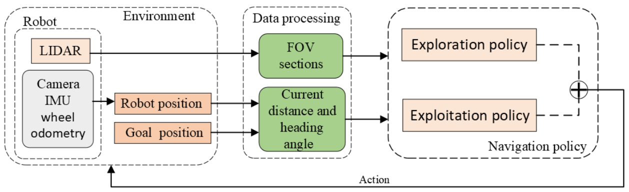

FIGURE 1. **The CDRL framework for a service robot. The exploitation policy is used for fast motion and simple target position while the exploration policy is aimed at safe obstacle avoidance and map exploration.**

A. EXPLOITAION POLICY

1) STATE SPACE

The state of the agent includes the number of 2D laser scans N, the updated distance d, and orientation angle α for more effective feature extraction. The ultimate state is defined as follows:

$$s = \{s_l \mid N, d, \alpha\} \in S \tag{17}$$

where d indicates the updated distance between the robot and the target in polar coordinates, and α depicts the orientation angle of the goal to the robot's current heading.

2) ACTION SPACE

The robot's actions can be defined as continuous choices. Continuous control commands for a differential velocity controller are defined as: $a_t = (v_t, w_t)$, where $v_t \in [0, 0.5]$ is the linear velocity of the agent, and $w_t \in [-1, 1]$ is the angular velocity.

3) REWARD FUNCTION

The behaviors of the agent are reshaped as the reward function including sparse and dense reward components. Based on the previous work [44], the reward function of the exploitation policy is redesigned to ensure navigation efficiency, which is comprised of the dense reward component and sparse reward component. The exploitation reward can be defined as:

$$R_{exploit}(s_t) = R_{goal}(s_t) + R_{col}(s_t) + R_d(s_t) + R_{oriented}(s_t) \tag{18}$$

The sparse reward component is calculated by returning the goal reward $R_{goal}(\mathbf{s}_t) = +500$ if the robot reaches the radius of the goal and a large punishment score with $R_{col}(\mathbf{s}_t) = -500$ is given if the distance between the robot's center and obstacles less than fixed safety distance r_i. $\mathbf{p}^t$ and $\mathbf{p}^{t-1}$ are the robot's pose at the current and previous timestamp, and $\mathbf{g}$ is the position of the goal. The dense reward includes R_d and $R_{oriented}$ described in equation (21) and equation (22), which reflects both distance and orientation aspects.

$$R_d(s_t) = c_1\left(\left\|\mathbf{p}^{t-1} - \mathbf{g}\right\| - \left\|\mathbf{p}^t - \mathbf{g}\right\|\right) \tag{19}$$

$$R_{oriented}(s_t) = k\left(\frac{\pi - \|\alpha\|}{\pi} + \cos(\alpha)\right) \tag{20}$$

where $\mathbf{c}$ is constant, π is a mathematical constant. If the robot gets to the target or the robot collides with obstacles. We choose $r_i = 0.4$ m, $c_1 = 0.2$, $k = 2$.

4) NETWORK ARCHITECTURE

The architecture of the actor network comprises 107 inputs including 105 laser scans from the 2D LiDAR sensor, deviation angles, and updated current distance calculated by the odometry module. At the input of neural network structures, the state of the network is normalized, and the layer normalization technique is used for stabilizing the training process. The normalized state is fed via three full layers with rectified linear unit activation functions. Inspired from [44], The actor network shown in Figure 2 is constructed by decreasing the network size over each layer, leading to optimizing computation and keeping the simple network and rich presentations. After using a re-parameterized trick, the output of the actor network is the distribution of the bound angular and linear velocity.

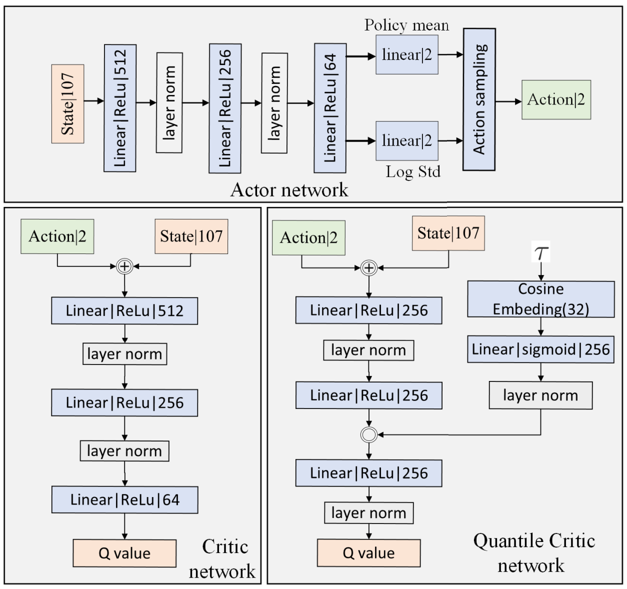

FIGURE 2. **Actor network, critic network, and quantile critic network.**

B. EXPLORATION POLICY

State Space, Action Space, Network Architecture: Transferring the exploitation policy directly to real-world environments increases collision rates during navigation toward the target position. Thus, an additional exploration policy is proposed within the CDRL framework to solve this challenge. This complementary policy fosters safe navigation by enabling the agent to explore the unseen environment and adapt to unforeseen obstacles, while still keeping efficient movement toward the destination. Both the exploitation and exploration policies utilize identical representations for the state space, action space, and neural networks. However, the objectives of the individual navigation policies are distinct. The exploitation policy prioritizes achieving the goal efficiently, favoring shorter paths. Conversely, the exploration policy focuses on ensuring safe curiosity-driven exploration. The reward function of the exploration policy can be formulated as follows:

$$R_{explore}(s_t) = R^i(s_t) + R^e(s_t) \tag{21}$$

While the extrinsic reward $R^e(\mathbf{s}_t)$ based on external environmental knowledge remains important, intrinsic reward $R^i(s_t)$ plays a crucial role in predicting current states. This capability fosters the agent's curiosity about the environment, driving further exploration. To design intrinsic rewards for the explorative policy, we utilize the two networks to define the error between the network $\bar{f}_\psi$ that predicts features of the observations and a fixed randomly initialized neural network f_ψ, using this error as an exploration bonus in RND [21]. These network structures are the same as the actor network. The exploration bonus can be defined as:

$$R^i(s_t) = \left\| f_\psi(s_{t+1}, a) - \bar{f}_\psi(s_{t+1}, a) \right\|_2^2 \tag{22}$$

For the external reward, the safe behaviors of the agent are reshaped to ensure safety and navigation efficiency by adding safety reward (R_s) from equation (20).

$$\begin{aligned} R^e(s_t) = R_{goal}(s_t) + R_{col}(s_t) + R_d(s_t) \\ + R_{oriented}(s_t) + R_s(s_t) \end{aligned} \tag{23}$$

The safety reward can be defined as:

$$R_s(s_t) = -c_2\left(1 - \frac{r_l}{2r_i}\right), \text{ if } r_i < r_l < 2r_i \tag{24}$$

where c_2 is a hyper-parameter, r_l is the minimum distance from the robot's centroid to its surrounding environments measured by the LiDAR. We choose $c_2 = 30$.

C. NAVIGATION STRATEGY BASED ON POLICY-SWITCHING MECHANISM

While an exploitation policy of the robot system, capitalizing on past experiences, can achieve efficient navigation in familiar settings, the robot may struggle with unforeseen obstacles like local optima (e.g., long walls or dead zones). To address this challenge, the framework employs a mechanism to select the right navigation policy for each situation. This could involve an exploration policy prioritizing the safe exploration of new environments when the exploitation policy encounters limitations.

$$\pi_{\text{CDRL}} = \begin{cases} \pi_{\text{exploit}}, & \text{if } \delta_{explore} \leq 0.5 \\ \pi_{\text{explore}}, & \text{otherwise} \end{cases} \tag{25}$$

A policy-switching mechanism is designed, which depends on the probability of selecting the exploration policy given the current state and the detection of potential local optima. The probability can be defined as:

$$\delta_{explore} = \frac{1}{1 + e^{-\beta(S_{local\_optima} - \mu)}} \tag{26}$$

where β is the coefficient, μ is the threshold value. S_{local_optima} indicates the local optima score, which fluctuates from 0 to 100, $(\beta, \mu, S_{local\_optima} \in \mathbb{R}_+^{n\times 1})$. The local optima score is calculated based on safety distance (d_{min_safe}), open path distance (d_{open_path}), and maximum change of heading angle (α_{max}). d_{open_path} indicates an open path distance when the robot sees the potential path. The open path zone is defined as an angle of 30 degrees in front of the robot. The above parameters were validated through a series of trials to ensure their appropriateness. When S_{local_optima} is below the threshold μ, the robot is likely not in local optima, and the exponent becomes negative, resulting in $\delta_{explore}$ close to zero, favoring exploitation behavior and effectively transitioning the policy towards exploitation. On the other hand, when $\delta_{explore}$ surpasses the threshold, potential local optima has been identified, promoting exploration behavior, effectively resulting in a switch to the exploration policy. The coefficient β controls the steepness of the transition between the two policies. A higher value of β makes the transition sharper, while a lower value results in a smoother transition. The local optima detection is illustrated in algorithm 1.

Algorithm 1 Local-Optima Detection

```
1: // Extract relevant information from the state: $L$, $\Delta$, $\alpha$
2: $L = [l_1, l_2, \ldots, l_n]$    ▷ LiDAR has $n$ number of laser beams
3: $\Delta_d = \|\mathbf{p}^t - \mathbf{g}\|$    ▷ distance from robot to the goal
4: $\alpha$    ▷ heading_angle
5: // Define threshold: $d_{min\_safe}$, $\alpha_{max}$, $d_{open\_path}$
6: $S_{local\_optima} \leftarrow 0$    ▷ Initialize local optima score
7: if $S_{local\_optima} \leq S_{local\_optima\_max}$ then
8:     if $\min(L) < d_{min\_safe}$ then
9:         $S_{local\_optima} \leftarrow S_{local\_optima} + 2$    ▷ Increase score if obstacle is detected
10:    end if
11:    if $\min(L_{open\_path}) < d_{open\_path}$ then
12:        $S_{local\_optima} \leftarrow S_{local\_optima} - 1$    ▷ Increase score if obstacle is detected
13:    end if
14:    // Analyze agent's behavior (heading angle)
15:    if $\alpha \geq \alpha_{max}$ then
16:        $S_{local\_optima} \leftarrow S_{local\_optima} + 1$
17:    end if
18: else
19:    $S_{local\_optima} = S_{local\_optima\_max}$
20: end if
21: if $S_{local\_optima} \leq 0$ then
22:    $S_{local\_optima} = 0$
23: end if
24: return $S_{local\_optima}$
```

V. LEARNING NAVIGATION POLICY

In this section, we will investigate how to train two navigation policies (exploitation and exploration policy), using different RL algorithms. To train this exploitation policy, we employ the SAC algorithm, which is well-suited for leveraging experience from the simulation environment. For exploration policy, we combine the distributional value function with intrinsic and extrinsic rewards to encourage the exploration of novel states and guide the agent toward the goal position.

1) EXPLOITATION POLICY

To train the exploitation policy for efficient navigation, we carefully leverage the reward function with the SAC algorithm. The specific implementation utilizes the off-policy SAC framework, which incorporates entropy bonus [41] within the actor-critic architecture [40]. For mapless navigation, the deep RL model is trained with a self-adjusting entropy coefficient, allowing it to balance exploration and exploitation and thereby facilitating the training process. To train the policy for efficient navigation to the destination regarding both navigation time and distance, the SAC algorithm proves highly effective in training robots in simulated environments. However, the policy inherently poses a risk of high collision rates when transferring to environments with new features (section VI-B). Additionally, employing the SAC algorithm to train the agent helps mitigate issues related to approximating value distribution of the DSAC algorithm in real-world scenarios, as mentioned in work [25]. In terms of the critic network shown in Figure 2, the network structure is the same as the actor network, and the Q-value is generated through a linear activation function.

2) DISTRIBUTIONAL EXPLORATION POLICY

To train the exploration policy, we use the soft action-state distributional value function to encourage the exploration of actions produced by the actor network. Based on the DSAC algorithm [25], we propose the Exploration-DSAC (E-DSAC) framework. In DSAC, when updating the parameter of the target value network based on the actions generated by the target policy network, the actions are generated based on previous experience with a soft update mechanism. However, this approach can lead to the policy being overly reliant on stereotypical actions and losing the stochasticity and curiosity for exploration during training. To train the exploration policy, the E-DSAC algorithm introduces two key modifications compared to the DSAC algorithm. First, we replace the soft update of the target policy network with actions produced by a random policy network. This eliminates randomness in the target value updates and encourages the exploration of diverse states. Second, we update the target distributional value function based on random actions with extrinsic and intrinsic rewards. The intrinsic reward is calculated by matching the loss between the target network and the predictor network, where the target network is randomly initialized and the predictor network is trained on data from the agent. This ensures the target value function remains aligned with the exploration goals and guides the agent towards novel experiences. The E-DSAC algorithm is illustrated in Algorithm 2.

Algorithm 2 E-DSAC for Mapless Navigation

```
Input: $s, a, s_0, \gamma \in (0,1), N, \kappa$
 1: Initialize parameters of policy network $\phi$
 2: Initialize two quantile Z-function $\theta_1, \theta_2$ and target
      quantile Z-function parameterized $\bar{\theta}_1, \bar{\theta}_2$
 3: Initialize RND predictor and prior parameter $\psi$
 4: Quantile fractions $\tau_i, i = 0, \dots, N, \tau_j, j = 0, \dots, N$
 5: Initialize replay memory $\mathcal{D}$
 6: while size($\mathcal{N}$) < MEMORY_SIZE do
 7:     The robot interact with environment $a_0 \sim \pi_\phi(\cdot|s_0)$
 8:     Get next observation $s_{t+1}$ and reward $R$
 9:     Store the transition $(\mathcal{S}_k, \mathcal{A}_k, \mathcal{R}_k, \mathcal{S}_{k+1})$ to $\mathcal{D}$
10: end while
11: for epoch = 0, training batch do
12:     Sample a mini-batch $B = (s, a)$ from $\mathcal{D}$
13:     Update RND predictor weights $\psi$ with gradient
          descent using loss function of equation (29)
14: end for
15: for epoch = 0 training batch do
16:     Sample a batch of transitions $\mathcal{M}$ from $\mathcal{D}$
17:     Get $\delta_{ij}^t$ using equation (10)
18:     Update $\theta_k$ by minimizing loss $\mathcal{L}_Z(\theta)$ in equation (12)
19:     Calculate $Q(s, \tilde{a})$ using equation (13)
20:     Update $\phi$ minimizing loss $\mathcal{L}_\pi(\phi)$ using equation (14)
21:     Soft update to target network: $\bar{\theta}_k$ using equation (17)
22: end for
23: Store neural network weight $\phi$
```

The predictor neural network $\bar{f}_\psi$ parameterized by ψ is trained by performing gradient descent to minimize the expected MSE:

$$\mathcal{L} = \frac{1}{B} \sum_{s \in \mathcal{D}} \left\| f_\psi(s, a) - \bar{f}_\psi(s, a) \right\|_2^2 \tag{27}$$

The architecture of the quantile critic network used by implicit quantile network (IQN) [33] is shown in Figure 2. The outputs of the quantile critic network are sets of quantile values that indicate the return distribution. Compared to the critic network used for the SAC algorithm, the quantile critic network uses multiplicative form, which combines state feature and the cosine function to embed τ_k with $k \in \{1, \dots N = 32\}$. The cosine function is defined as:

$$\phi_j(\tau_k) := ReLU(\sum_{i=0}^{n-1} cos(\pi i \tau_k) w_{ij} + b_j) \tag{28}$$

The Hadamard product is implemented between the embedding $\phi_j(\tau_k)$ and the second fully connected layer output of state features. Additionally, layer normalization is added to keep the training process more stable.

VI. EXPERIMENTS AND RESULTS

In this section, we validate our methodology through a comprehensive analysis with an existing approach to find the balance between safe, fast behaviors and the high success rate of navigation tasks. SAC and E-DSAC algorithms are trained for the single policy for mapless navigation. Then, the CDRL framework is deployed and compared with each pure single policy in simulation. After that, a comparison of our CDRL framework with the Entropy-threshold policy is evaluated for sim-to-sim scenarios. Finally, we implement several real-world experiments to demonstrate the potential efficient navigation for real-time systems.

A. TRAINING SETUP

While training the agent in a simplified simulation offers significant time efficiency compared to a realistic setting [28], it comes with the drawback of removing natural elements. This removal makes the transfer of policies in the real world more challenging and could even fail. We therefore choose the Gazebo simulation integrated with Robot Operating System (ROS) Noetic for the comparative analysis. For the analysis, arbitrary targets are configured randomly in the Gazebo environments including visible and occlusion targets in complex scenarios for robot mapless navigation. a service robot and a training environment are depicted in Figure 3. Each target position is generated randomly after the end of the episode. The whole training process is conducted on a Gazebo simulation with a 10Hz control frequency. The simulation resets (after 0.1s) when the robot's collision occurs in the virtual environment. The training is conducted with two platform robot models (the service robot and the pioneer P3DX) equipped with a planar LiDAR for approximately 1300 episodes. The simulation is conducted on a computer with core™ i7-8700 CPU and Geforce RTX 2060 Ti GPU.

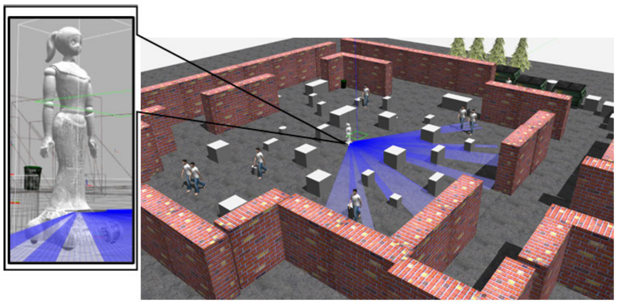

FIGURE 3. The service robot model (left image) and the virtual environment in 3D robotics simulator for training SAC and DSAC algorithms.

To investigate the impact of reward shaping on the navigation of mobile robots, we train exploitation policy with the SAC algorithm [42] and exploitation policy with the E-DSAC algorithm. Additionally, we also train the DSAC algorithm [25] for the single policy with the reward function listed in [44], which primarily consists of sparse reward and does not include safety reward R_s. Figure 4 shows that the learning process of the exploitation policy outperformed the other two methods, but it exhibited the phenomenon of "overfitting" after a long training time. One can observe that applying the distributional value function

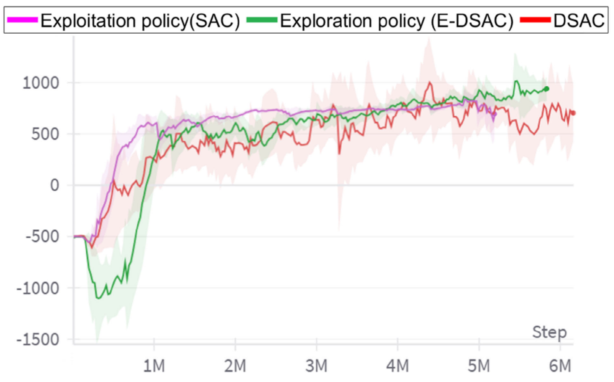

FIGURE 4. The evolution of learning curve of SAC and E-DSAC algorithms under different reward functions.

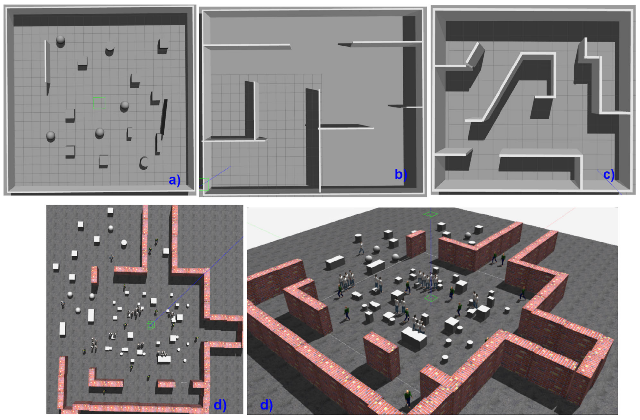

FIGURE 5. Static environment structures for testing including environment (a), (b), (c) from left to right side respectively. Environment (d) is a dynamic environment.

TABLE 1. Success rate and collision rate for 50 runs in testing environment (success rate/collision rate).

	pioneer model		service robot model	
Method	env(a)	env(b)	env(c)	env(d)
SAC	0.88/0.12	–	–	0.59/0.41
E-DSAC	0.902/0.10	0.98/0.02	–	0.45/0.55
CDRL	0.94/0.06	0.90/0.10	0.98/0.02	0.92/0.08

in a complex training environment makes policy learning slower due to the approximation issue. For the exploration policy trained by the E-DSAC algorithm, the agent can learn the concept of "curiosity" about environments without relying on previous experience. Another important note is that the reward components should be carefully designed due to the impact of agent learning during the training process. The parameters for training utilized are described in Table 3 with the simulation time progressing at three times the real-time simulation speed.

B. SIMULATION EVALUATION

1) CDRL FRAMEWORK AND SINGLE POLICY

The proposed CDRL framework could achieve a success rate in reaching the target, and reduce the collision rate

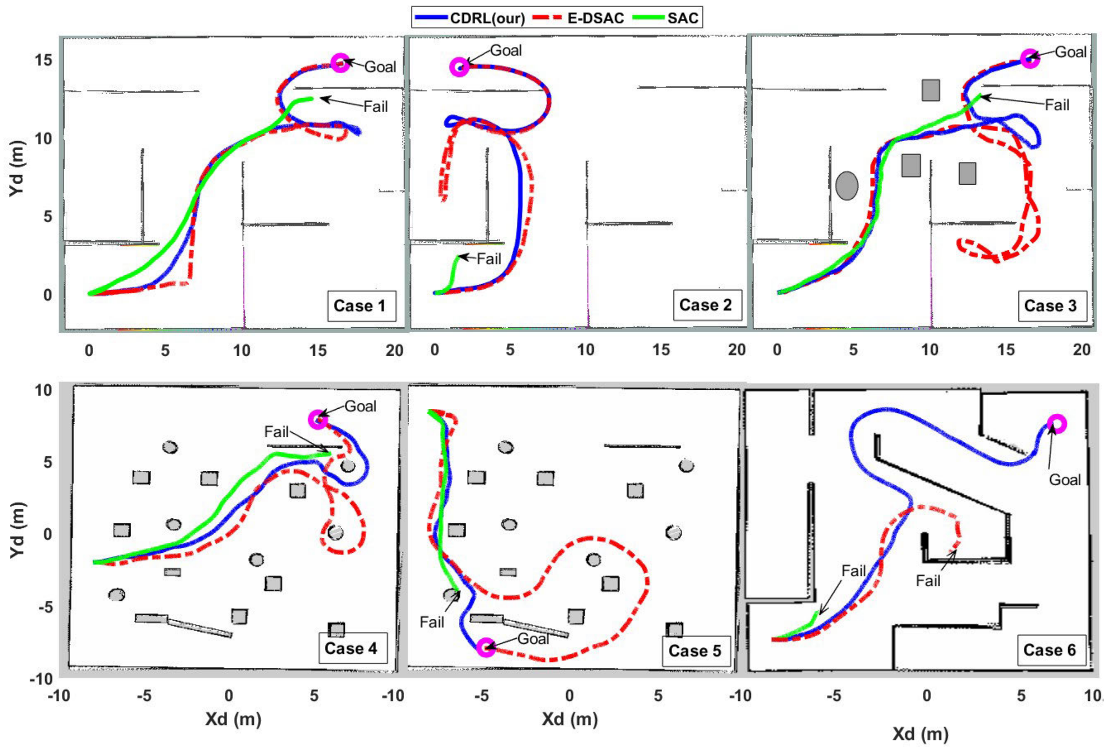

FIGURE 6. The trajectories visualization of SAC, E-DSAC, and CDRL is presented in 6 cases from (a)-(f). When transferring the policy to another simulation environment, the path of SAC shows navigation task failures while E-DSAC and CDRL paths complete navigation to the target position. Visually, CDRL demonstrates a shorter path than E-DSAC.

through a comparison with two native policies (exploitation policy and exploration policy) sampled by SAC and E-DSAC algorithms respectively. We tested 2 robot models (pioneer and service robot) in 3 static complex environments and a dynamic environment shown in Figure 5 over 50 runs. The first environment (a) is most similar to the training environment. Environment (b) presents different structures, while environment (c) is a complex environment, including corridors and dead-end corners. The comparison results are shown in Table 1, where the CDRL framework used for the pioneer model and service robot model achieved the highest success rate over the environment (a) (94%), (c) (98%), and (d) (92%) while also exhibiting the lowest collision rate. The exploitation policy used for the service robot and pioneer models could not reach the target in environments (b) and (c). Although the exploration policy obtained the highest success rate in the environment (b), the exploration policy could not be deployed in the environment (c) and got only a 45% success rate in the environment (d). Moreover, the difference in structure between the testing environment (c) and the training environment makes the single policy (trained by SAC and E-DSAC algorithms) difficult to navigate toward the goal position.

As shown in Figure 6, the visualized trajectories for each environment demonstrate the trade-off between safety and efficiency. The exploration policy navigates safely to the target. However, it tends to be overly conservative to create a longer trajectory. In contrast, the exploitation policy focuses on finding the shortest path with fewer states than the exploration policy to the goal. However, this focus often led to collisions during navigation. We observed that the CDRL framework generated a shorter free-collision path compared with the exploration policy and still guaranteed safe navigation. Table 2 presents the navigation time and navigation distance corresponding to six cases in Figure 6. In most cases, the time and distance traveled by the CDRL framework are more efficient (shorter is better) than those of the exploration policy when navigating to the same destination. While the exploitation policy shows the shortest path and travel time, it results in collisions during the movement process.

TABLE 2. Simulation results of 6 cases in static environments with navigation time(NT) and navigation distance(ND).

	SAC		E-DSAC		CDRL	
	NT(s)	ND(m)	NT	ND	NT	ND
Case 01	39	19.7	80	35.5	**74**	**33.6**
Case 02	2.7	10	96	43.3	**79**	**34.5**
Case 03	52	25.6	86	41	**74**	**36.77**
Case 04	34	12.8	78	39.6	**17.8**	**36.1**
Case 05	40	17.05	96	33.8	**52**	**24.78**
Case 06	7.8	17	70	18.8	**53**	**19.5**

2) CDRL FRAMEWORK AND ENTROPY-THRESHOLD POLICY

In this section, we evaluate our method with an Entropy-Threshold Policy called H-Entropy [45] as the baseline in

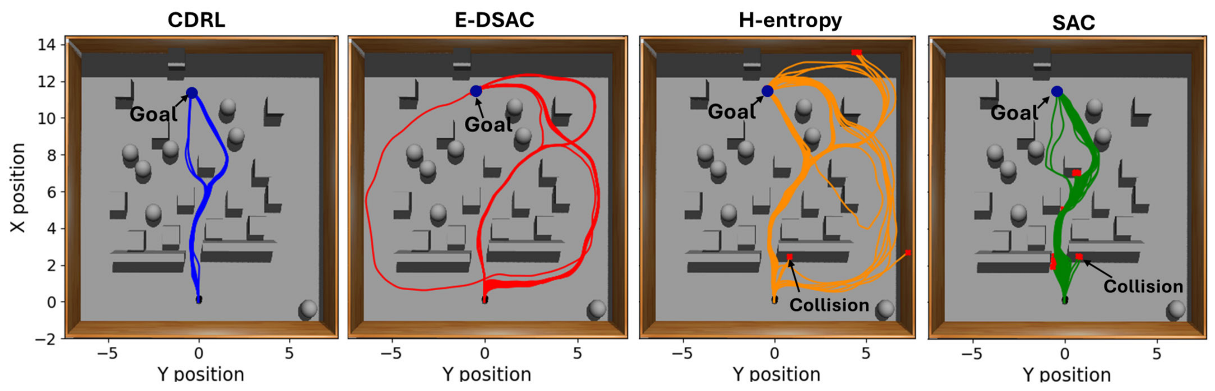

FIGURE 7. Trajectories of CDRL framework, H-entropy method, exploitation policy trained by SAC, and exploration policy trained by E-DSAC in a dense static obstacles environment.

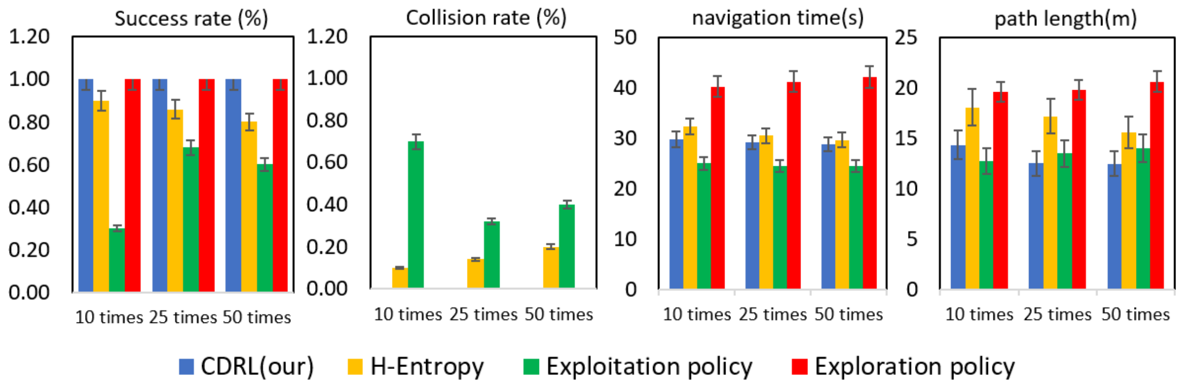

FIGURE 8. The quantitative results in the dense static obstacles environment with 10, 25, and 50 navigation times. The proposed method is evaluated with four metrics: success rate, collision rate, navigation times, and path length.

comparison to two typical environments, including the dense static obstacle environment and the dynamic environment. Entropy is directly proportional to the action space, indicating the stochastic nature of actions undertaken by an agent navigating in unfamiliar environments. Higher entropy signifies increased action unpredictability of the action, enabling policies to explore uncharted regions instead of getting stuck in local optima due to new features (e.g. long walls or wall corners). Following equation (29), the entropy can be defined:

$$\mathcal{H}_k = -\ln \frac{1}{|A|} \sum_a \pi_k (a \mid s_t) \ln \pi_k (a \mid s_t) \tag{29}$$

where $|A|$ is the cardinality of the state space. Then, if $\mathcal{H}_{exploit}$ is less then $\mathcal{H}_{explore}$, the $\pi_{exploit}$ is selected, otherwise the action will be generated by $\pi_{explore}$:

$$\pi_k (a \mid s_t) = \begin{cases} \pi_{exploit} (a \mid s_t) & \text{if} \quad \mathcal{H}_{exploit} < \mathcal{H}_{explore} \\ \pi_{explore} (a \mid s_t) & \text{otherwise} \end{cases} \tag{30}$$

Figure 7 shows the trajectories of not only the CDRL and H-entropy frameworks but also exploration and exploitation policies trained by E-DSAC and SAC algorithms respectively. Figure 8 indicates that the CDRL framework achieved a high success rate with the H-entropy method, and also got better navigation performance regarding navigation time and path length compared with exploration policy.

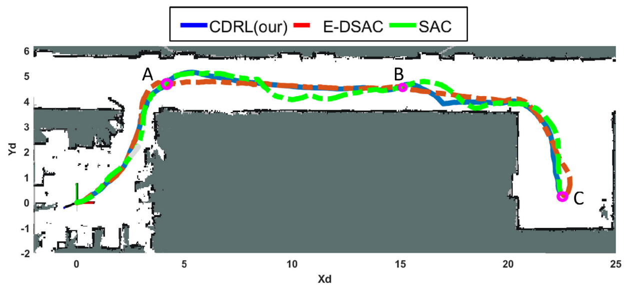

FIGURE 9. Path trajectories of CDRL framework, E-DSAC, and SAC policy in a corridor environment (23 × 8 m^2).

C. TESTING REAL-WORLD SCENARIOS AND TASK DESCRIPTION

The robot utilizes a low-cost YDLIDAR G6 sensor, which is sensitive to light and temperature. The robot utilizes 120 degrees of field of view (FOV) with a maximum measuring 5 m, and an angular resolution of 0.25°.

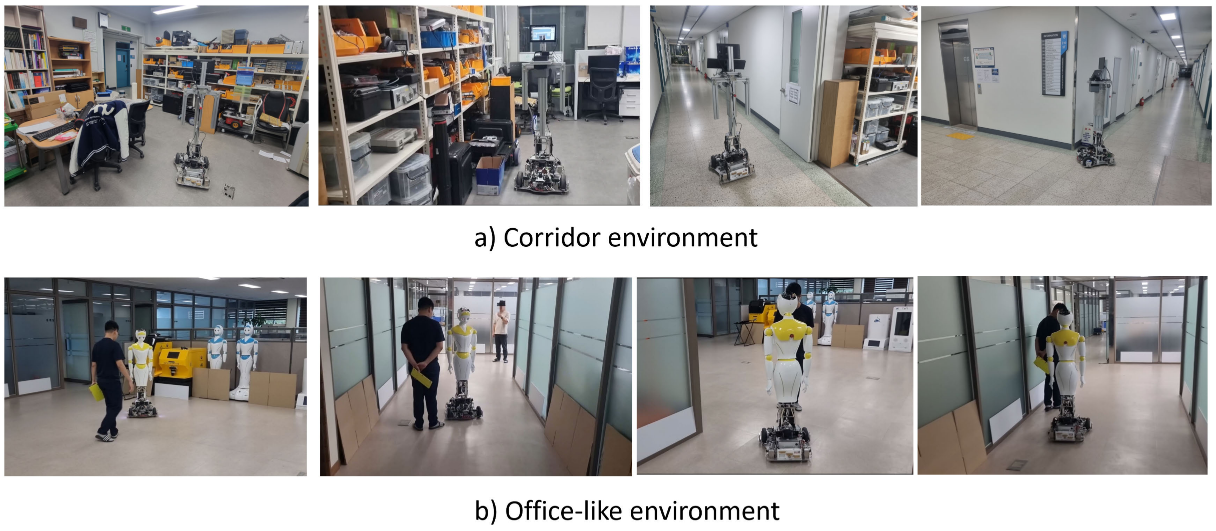

FIGURE 10. Real-world experiment, (a): a corridor scenario; (b): an office-like environment with natural human motions.

TABLE 3. Training hyperparameters and CDRL framework parameters.

Parameter	Value	Parameter	value
linear velocity	[0.0,0.5] m/s	Discount factor	0.99
FOV	120°	Batch size	256
angular velocity	[-1.0,1.0]m/s	$d_{min\_safe} = r_i$	0.4 m
Maximum steps/1 episode	1200	d_{open_path}	2 m
frequency	10 Hz	β	0.05
Optimizer	Adam	μ	50
Learning rate	3.10^{-4}	l	3.0 m
Buffer size	3.10^{6}	α_{max}	$\frac{\pi}{2}$

TABLE 4. Testing results in two real-world environments in terms of total navigation distance(TND), total navigation time(TNT), and the number of reaching goals (NoG).

Method	Corridor environment			Office-like environment		
	SAC	EDSAC	**CDRL**	SAC	EDSAC	**CDRL**
TND (m)	51.38	50.92	**49.4**	20.17	77.4	**69.54**
TNT (s)	147.6	235.6	**160**	123	501.2	**445**
NoG	3/3	3/3	3/3	1/4	3/4	4/4

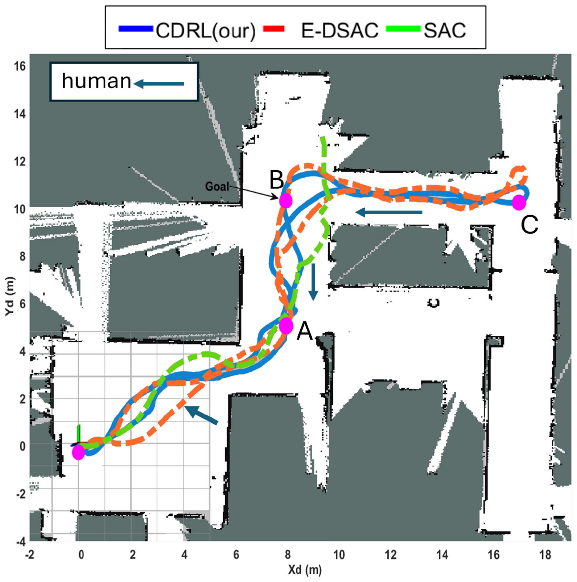

FIGURE 11. Path trajectories of CDRL framework, E-DSAC, and SAC policy in an office-like environment (20×20 m^2).

Target localization: RTAB-Map [46] is used to build a grid map of the testing scenario and ROS AMCL [47] is to localize the robot in this map. It is imperative to emphasize that this map is solely employed for calculating the target position and is not employed by motion planning. The target position in the robot frame could be calculated using the robot and goal coordinates on the map. The computer mounted on the robot is a mini PC with an Intel Core i7 5700U Processor.

We implemented the CDRL framework on the mobile service robot on ROS2 foxy [48] to validate sim-to-real transfer and generalize the performance of the navigation tasks. The robot is tested in two real-world scenarios including an office-like environment and a corridor environment shown in Figure 10. While Figure 9 depicts the robot's navigation from a dense-static obstacles room to the lobby in the corridor environment, Figure 11 shows the process robot's navigation toward the target with dynamic obstacles (e.g. human motions) in the office-like environments. To mark the target position accurately on the map, we used teleoperation to navigate the robot to the wanted points A, B, and C, and then come to the starting point. At each target, the coordinate is collected for the mapless navigation process. The target error is 0.2 meters. After reaching one target, the robot would wait for 1 second to show it successfully reached this target rather than occasionally traversing this target when moving to the other target. The testing result is illustrated in Table 4, which proves the promising results of the CDRL framework in terms of navigation time, navigation distance, and success rate.

VII. CONCLUSION

To implement efficient navigation tasks, the paper tackles mapless navigation in unaware scenarios by introducing the CDRL framework that enables the mobile service robot to escape dead ends and reach the target position faster with a high success rate. Here, we lay greater emphasis on each policy that is appropriate to the complexity of the environment. The CDRL framework chooses the right deep RL policy to navigate safely and efficiently in environments without re-training. The actions generated from the CDRL framework also encourage exploration when transferring the deep RL models to unaware environments. With evaluation in simulation and real-world scenarios, the framework shows more close-to-optimal trajectories than a naive policy transfer and fast motion by considering surrounding obstacles. However, our method still has a limitation. Mathematically, the policy-switching equation is not flexible because of the hyperparameter tuning. Expert knowledge is used to set the threshold μ based on environment complexity, its impact across varying environments warrants further exploration. We aim to address the varying μ as an open subject for future research.

REFERENCES

[1] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Sci. Technol.*, vol. 26, no. 5, pp. 674–691, Oct. 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235516106

[2] N. Buniyamin, W. A. J. W. Ngah, N. Sariff, and Z. Mohamad, "A simple local path planning algorithm for autonomous mobile robots," *Int. J. Syst. Appl. Eng. Develop.*, vol. 5, no. 2, pp. 151–159, 2011.

[3] D. H. T. Kim, T. N. Manh, C. N. Manh, N. D. Nguyen, D. P. Tien, M. T. Van, and M. P. Xuan, "Adaptive control for uncertain model of omni-directional mobile robot based on radial basis function neural network," *Int. J. Control, Autom. Syst.*, vol. 19, no. 4, pp. 1715–1727, Apr. 2021.

[4] L. C. Santos, A. S. Aguiar, F. N. Santos, A. Valente, J. B. Ventura, and A. J. Sousa, "Navigation stack for robots working in steep slope vineyard," in *Proc. SAI Intell. Syst. Conf.*, vol. 1. Cham, Switzerland: Springer, 2021, pp. 264–285.

[5] K. Zheng, "RoS navigation tuning guide," in *Robot Operating System (ROS)*, vol. 6. Springer, 2021, pp. 197–226.

[6] T. Fan, X. Cheng, J. Pan, D. Manocha, and R. Yang, "Crowd-Move: Autonomous mapless navigation in crowded scenarios," 2018, *arXiv:1807.07870*.

[7] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 856–892, Jun. 2020.

[8] C. Xiao, P. Lu, and Q. He, "Flying through a narrow gap using end-to-end deep reinforcement learning augmented with curriculum learning and Sim2Real," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2701–2708, May 2023.

[9] X. Chen, A. Ghadirzadeh, J. Folkesson, M. Björkman, and P. Jensfelt, "Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 3110–3116.

[10] J. Choi, K. Park, M. Kim, and S. Seok, "Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5993–6000.

[11] T. Chaffre, J. Moras, A. Chan-Hon-Tong, and J. Marzat, "Sim-to-real transfer with incremental environment complexity for reinforcement learning of depth-based robot navigation," 2020, *arXiv:2004.14684*.

[12] X. Lin, J. McConnell, and B. Englot, "Robust unmanned surface vehicle navigation with distributional reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 6185–6191.

[13] C. Liu, E.-J. van Kampen, and G. C. H. E. de Croon, "Adaptive risk-tendency: Nano drone navigation in cluttered environments with distributional reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 7198–7204.

[14] S. Takamuku and R. C. Arkin, "Multi-method learning and assimilation," *Robot. Auto. Syst.*, vol. 55, no. 8, pp. 618–627, Aug. 2007.

[15] J. Gao, W. Ye, J. Guo, and Z. Li, "Deep reinforcement learning for indoor mobile robot path planning," *Sensors*, vol. 20, no. 19, p. 5493, Sep. 2020.

[16] N. D. Toan and K. G. Woo, "Mapless navigation with deep reinforcement learning based on the convolutional proximal policy optimization network," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Jan. 2021, pp. 298–301.

[17] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. D. S. L. Cuadros, and D. F. T. Gamarra, "Soft actor-critic for navigation of mobile robots," *J. Intell. Robotic Syst.*, vol. 102, no. 2, p. 31, Jun. 2021.

[18] L. Shani, Y. Efroni, and S. Mannor, "Exploration conscious reinforcement learning revisited," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5680–5689.

[19] R. Kumaraswamy, M. Schlegel, A. White, and M. White, "Context-dependent upper-confidence bounds for directed exploration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–7.

[20] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[21] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," 2018, *arXiv:1810.12894*.

[22] L. Pan, A. Li, J. Ma, and J. Ji, "Learning navigation policies for mobile robots in deep reinforcement learning with random network distillation," in *Proc. 5th Int. Conf. Innov. Artif. Intell.*, Mar. 2021, pp. 151–157.

[23] H. Yin, S. Su, Y. Lin, P. Zhen, K. Festl, and D. Watzenig, "Random network distillation based deep reinforcement learning for AGV path planning," 2024, *arXiv:2404.12594*.

[24] Y. Zhou, E.-J. van Kampen, and Q. Chu, "Hybrid hierarchical reinforcement learning for online guidance and navigation with partial observability," *Neurocomputing*, vol. 331, pp. 443–457, Feb. 2019.

[25] X. Ma, L. Xia, Z. Zhou, J. Yang, and Q. Zhao, "DSAC: Distributional soft actor critic for risk-sensitive reinforcement learning," 2020, *arXiv:2004.14547*.

[26] H. Shi, L. Shi, M. Xu, and K.-S. Hwang, "End-to-end navigation strategy with deep reinforcement learning for mobile robots," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2393–2402, Apr. 2020.

[27] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," 2016, *arXiv:1611.03673*.

[28] E. Marchesini and A. Farinelli, "Discrete deep reinforcement learning for mapless navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 10688–10694.

[29] L. Yang, J. Bi, and H. Yuan, "Dynamic path planning for mobile robots with deep reinforcement learning," *IFAC-PapersOnLine*, vol. 55, no. 11, pp. 19–24, 2022.

[30] 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 *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2021, pp. 144–149.

[31] F. Leiva and J. Ruiz-del-Solar, "Robust RL-based map-less local planning: Using 2D point clouds as observations," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5787–5794, Oct. 2020.

[32] M. Luong and C. Pham, "Incremental learning for autonomous navigation of mobile robots based on deep reinforcement learning," *J. Intell. Robotic Syst.*, vol. 101, no. 1, p. 1, Jan. 2021.

[33] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1096–1105.

[34] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 2892–2901.

[35] C. Bodnar, A. Li, K. Hausman, P. Pastor, and M. Kalakrishnan, "Quantile QT-opt for risk-aware vision-based robotic grasping," 2019, *arXiv:1910.02787*.

[36] V. M. Tran and G.-W. Kim, "Mapless navigation with distributional reinforcement learning," *J. Korea Robot. Soc.*, vol. 19, no. 1, pp. 92–97, Feb. 2024.

[37] K. Rezaee, P. Yadmellat, and S. Chamorro, "Motion planning for autonomous vehicles in the presence of uncertainty using reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 3506–3511.

[38] J. Choi, C. Dance, J.-E. Kim, S. Hwang, and K.-S. Park, "Risk-conditioned distributional soft actor-critic for risk-sensitive navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 8337–8344.

[39] J. Martin, M. Lyskawinski, X. Li, and B. Englot, "Stochastically dominant distributional reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6745–6754.

[40] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[41] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. Int. Conf. Mach. Learn.*, Aug. 2017, pp. 1352–1361.

[42] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

[43] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 449–458.

[44] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 31–36.

[45] A. Sestini, A. Kuhnle, and A. D. Bagdanov, "Policy fusion for adaptive and customizable reinforcement learning agents," in *Proc. IEEE Conf. Games (CoG)*, Aug. 2021, pp. 1–8.

[46] M. Labbé and F. Michaud, "RTAB-map as an open-source LiDAR and visual simultaneous localization and mapping library for large-scale and long-term online operation," *J. Field Robot.*, vol. 36, no. 2, pp. 416–446, Mar. 2019.

[47] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, "ROS-based localization of a race vehicle at high-speed using LiDAR," *E3S Web Conf.*, vol. 95, Feb. 2019, Art. no. 04002.

[48] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, May 2022, Art. no. eabm6074.

VAN MANH TRAN received the B.S. degree in automation and control from Hanoi University of Science and Technology, in 2020. He is currently pursuing the master's degree with the Department of Control and Robot Engineering, Chungbuk National University, South Korea. His research interests include robotics, reinforcement learning, motion planning, and control systems.

GON-WOO KIM (Member, IEEE) received the M.S. and Ph.D. degrees from Seoul National University, South Korea, in 2002 and 2006, respectively. He is currently a Professor with the Department of Intelligent Systems and Robotics, Chungbuk National University, South Korea. His research interests include navigation, localization, and SLAM for mobile robots and autonomous vehicles.