CRAR: Coherent Risk-Aware Regulation for DRL-based Mapless Robot Navigation

Shaojie Jian[1], Changyun Wei*[1], Shunyu Tian[1], Xiangyu Wu[1] and Ze Ji[2]

***Abstract*—Navigating without prior maps poses significant challenges for robots, especially in complex environments populated with pedestrians and obstacles. In such settings, achieving a balance between safety, socially compliant behavior, and path efficiency is critical. To address this, we propose a novel deep reinforcement learning-based approach, termed Coherent Risk-Aware Regulation (CRAR), tailored for Unmanned Ground Vehicles (UGVs). Our method incorporates a risk prediction network that enables the robot to anticipate potential hazards arising from pedestrians or obstacles. Furthermore, by integrating risk assessment and correction mechanisms, CRAR facilitates rapid learning of avoidance strategies, thereby reducing training duration. Consequently, the robot can identify navigation paths that are both safer and more efficient. Simulation results demonstrate that, in open environments, CRAR achieves an average success rate improvement of 25.33% and reduces navigation time by 10.85% compared to baseline methods. In dense environments, these benefits are amplified, with a success rate increase of 30.46% and a navigation time reduction of 21.75%. Additionally, CRAR ensures greater safety margins from obstacles and smoother trajectories, underscoring its efficacy in navigating dense and dynamic settings. Finally, we validate the performance and practicality of the proposed method through real-world experiments. A video demo is available at https://youtu.be/j0-pEd7_gr8.**

***Impact Statement*—Autonomous robot navigation in unstructured, dynamic environments is a grand challenge, particularly in balancing safety, social compliance, and efficiency. Existing methods, such as ORCA and traditional deep reinforcement learning approaches, often fall short in these aspects. Our CRAR framework addresses these limitations by integrating risk prediction, assessment, and correction mechanisms into a DRL-based system. This enables UGVs to proactively anticipate and mitigate hazards, improving both safety and efficiency. By bridging DRL with risk-aware decision-making, CRAR sets a new benchmark for autonomous navigation and paves the way for future advancements in socially adaptive robotics.**

***Index Terms*—Safe navigation, unmanned ground vehicles, dense environments, risk-aware mechanisms.**

I. Introduction

AN increasing number of robots are deployed across diverse applications, notably in autonomous navigation [1]. Historically confined to structured settings like factories and warehouses, robots now face demands to operate in dynamic, human-centric environments such as smart cities [2], transportation hubs [3], and healthcare facilities [4]. Navigating these complex, crowded spaces poses significant challenges, requiring efficient, safe, and socially aware solutions.

Dense environments present robots with diverse obstacles—walls, pedestrians, other robots, and vehicles—whose movements demand continuous perception and prediction for safe operation [5]. Reactive methods, such as Reciprocal Velocity Obstacles (RVO) [6] and Optimal Reciprocal Collision Avoidance (ORCA) [7], adjust trajectories in real time based on local data, offering rapid responses in dynamic contexts. Yet, in highly crowded scenarios, these approaches risk "robot freezing" [8]. In contrast, predictive methods [9] forecast obstacle trajectories to enhance safety, but their dependence on precise, single-step predictions [10] limits adaptability in unpredictable, dense settings, often yielding shortsighted decisions and poor generalization.

Deep Reinforcement Learning (DRL) has emerged as a transformative approach to navigation, learning optimal policies through environmental interaction and excelling in dynamic contexts [11], [12]. Unlike rule-based methods, DRL captures complex robot-human interactions via deep neural networks, adeptly handling multi-objective tasks. However, challenges remain: conventional DRL often oversimplifies obstacle models, ignoring interaction effects and compromising safety in dense settings [13], while prioritizing short-term actions over long-term planning, risking erratic behavior or collisions [14]. Although some studies integrate risk mechanisms for safety, they rarely address real-world uncertainties like prediction errors for dynamic obstacles [15].

To address these gaps, we propose the Coherent Risk-Aware Regulation (CRAR) framework, a DRL-based method for mapless navigation in dense environments. CRAR enhances efficiency and safety through robust risk prediction, measurement, and correction, avoiding shortsighted or intrusive strategies. Our key contributions are:

- A risk prediction network that fuses spatio-temporal sensory data to quantify future risks posed by pedestrians and obstacles, thereby enabling proactive collision avoidance and facilitating multi-step planning.
- Real-time risk assessment via LiDAR, paired with a safety-focused reward function, to identify secure, efficient trajectories.
- The Risk Correction module improves efficiency while ensuring robot safety by adjusting actions in high-risk scenarios, and promotes rapid mastery of obstacle avoidance.

[1] Shaojie Jian, Changyun Wei, Shunyu Tian and Xiangyu Wu are with the College of Mechanical and Electrical Engineering, Hohai University, Changzhou, 213200, China

[2]Ze Ji is with the School of Engineering, Cardiff University, Cardiff CF24 3AA, United Kingdom

This work was supported in part by the National Natural Science Foundation of China under Grant 52371275 and in part by the National Key Research and Development Project of China under Grant 2024YFC3211001.

*Corresponding author: Changyun Wei (e-mail: c.wei@hhu.edu.cn)

- A mapless navigation system is implemented in the Robot Operating System (ROS), outperforming baseline methods in simulations and validated in real-world scenarios.

The remainder of this paper is organized as follows. Section II reviews related work in robotic navigation. Section III outlines the theoretical foundations of our approach. Section IV provides an overview of the system and a detailed exposition of the CRAR approach. Section V presents experimental results validating the method, and Section VI offers the conclusion and an outlook on future work.

II. Related Work

Autonomous robot navigation has emerged as a cornerstone of robotics, with applications spanning healthcare [16], manufacturing [17], and logistics [18]. In complex and densely populated environments, such as hospitals, factories, and urban settings, robots must perform critical tasks like goods transportation and inspections while ensuring safety and efficiency. Decades of research have advanced navigation strategies for such scenarios, broadly categorized into three paradigms: reactive methods, trajectory prediction-based approaches, and learning-based techniques [19], [20].

A. Reactive Methods

Reactive navigation methods enable real-time obstacle avoidance by adjusting a robot's trajectory based on immediate sensory inputs. The Social Force (SF) model [21], for instance, simulates pedestrian dynamics by treating interactions as repulsive forces, allowing robots to adapt their speed and direction to avoid collisions [22]. Similarly, the ORCA framework [7] models surrounding agents as velocity obstacles, computing collision-free velocities under reciprocal assumptions. These methods excel in dynamic settings due to their computational simplicity and responsiveness. However, their reliance on local observations often leads to shortsighted decisions, manifesting as oscillatory behavior or the "robot freezing" problem in crowded environments [8]. Moreover, by focusing solely on pairwise interactions, reactive approaches may fail to capture broader crowd dynamics, resulting in suboptimal or unnatural navigation paths.

B. Trajectory Prediction-Based Methods

Prediction-based methods enhance navigation by forecasting the future states of dynamic obstacles, such as pedestrian trajectories. Huang et al. [23] have employed spatio-temporal graph attention networks to predict pedestrian movements, enabling robots to plan proactive routes. Similarly, Filotheou et al. [24] integrates Model Predictive Control (MPC) to coordinate multiple agents, leveraging local perception and communication for safe navigation. While these approaches improve long-term planning over reactive methods, their efficacy hinges on prediction accuracy. Error accumulation over extended horizons often degrades performance, particularly in unpredictable or dense settings [10]. Additionally, MPC's dependence on precise system models renders it vulnerable to unexpected behaviors when modeling assumptions fail, limiting its robustness in real-world applications.

C. Learning-Based Methods

Learning-based techniques offer a paradigm shift by enabling robots to adaptively acquire navigation strategies. Imitation Learning (IL) trains robots to mimic expert demonstrations, bypassing extensive environmental interaction [25]. Inverse Reinforcement Learning (IRL), a related approach, infers reward functions from expert behavior to derive optimal policies [26]. While effective, these methods depend heavily on the quality and diversity of demonstration data, posing scalability challenges. In contrast, DRL autonomously learns policies through trial-and-error interactions, optimizing navigation strategies for complex, multi-objective tasks [11]. DRL's ability to model intricate robot-human interactions and design end-to-end frameworks makes it particularly promising. However, existing DRL approaches often oversimplify obstacle dynamics or prioritize short-term actions, leading to unsafe or erratic behavior in dense environments [13], [14]. Efforts to incorporate risk awareness [15] have improved safety but frequently overlook real-world uncertainties, such as prediction errors, constraining their practical utility.

D. Safe Reinforcement Learning

Safe Reinforcement Learning (SRL) aims to prevent unsafe behaviors by modifying the reward function or introducing constraints, focusing on ensuring safety within the learning process itself [27]. For example, instead of solely maximizing expected cumulative rewards, some approaches consider the risk in the reward distribution to embed safety considerations [28]. Others employ methods such as Constrained Markov Decision Processes (CMDPs) to restrict exploration in risky states and enhance safety [29]. In contrast, our method focuses on real-time safe navigation, optimizing the robot's actions through risk prediction and correction to minimize collision risk in dynamic environments.

E. Mapless Navigation

In unfamiliar or dynamic settings, map-based navigation becomes impractical, driving interest in mapless approaches. Tai et al. [30] have developed a DRL-based planner using sparse LiDAR data, reducing reliance on preconstructed maps and enhancing deployment flexibility. Dobriborsci et al. [31] combines RL with MPC, employing stacked Q-functions to evaluate future actions in real time, further advancing mapless navigation. These methods demonstrate robustness in unknown environments, yet challenges remain in balancing safety and efficiency amidst dense, dynamic obstacles.

F. Positioning of This Work

While prior work has made significant strides, limitations persist: reactive methods lack foresight, prediction-based approaches falter under uncertainty, and learning-based strategies struggle with generalization and safety in dense settings. Our proposed CRAR framework addresses these gaps by integrating a risk prediction network, real-time LiDAR-based risk assessment, and a correction module within a DRL architecture. Unlike existing methods, CRAR further takes into account future risk and crowd dynamics. It then guides

the robot toward low-risk areas, achieving safer and more efficient navigation without the need for prior mapping. By leveraging lightweight sensors and a tailored reward function, our approach bridges the gap between theoretical robustness and practical applicability.

III. Preliminary

A. Problem Formulation

We consider a robot operating in a complex, dynamic factory environment, tasked with inspection or material transportation. The goal is to derive a navigation policy π that governs the robot's linear velocity ν and angular velocity ω, allowing the robot to reach a designated target safely and efficiently. In such settings, preconstructed maps are unreliable due to real-time changes in obstacle positions, rendering traditional map-based navigation impractical. To address this, we employ a DRL framework, modeling the mapless navigation task as a Partially Observable Markov Decision Process (POMDP) defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \Theta, \gamma \rangle$. Here, $\mathcal{S}$ represents the state space, encapsulating the robot's environment; $\mathcal{A}$ denotes the action space, comprising continuous controls $a = [\nu, \omega]$; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability function; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, providing immediate feedback for taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$; Ω is the observation space, with observations $o \in \Omega$ reflecting partial environmental information; $\Theta : \mathcal{S} \to \mathcal{P}(\Omega)$ is the observation probability distribution, where $o \sim \Theta(s)$; $\gamma \in [0, 1]$ is the discount factor, balancing short-term and long-term rewards. This formulation captures the uncertainty and partial observability inherent in mapless navigation, enabling the robot to learn adaptive strategies through interaction with the environment.

B. Reinforcement Learning for Navigation Policy

In an unknown, dynamic environment, the robot achieves mapless navigation by continuously perceiving its surroundings and exploring viable paths. Starting from an initial state s_1, the robot aims to reach a target state s_T while avoiding collisions. Its trajectory τ is represented as a sequence of state-action pairs over a finite horizon T: $\tau = [s_1, a_1, s_2, a_2, \cdots, s_{T-1}, a_{T-1}, s_T]$. For a given policy π_θ, the distribution of robot's navigation from s_1 to s_T can be expressed by:

$$P(\tau) = p(s_1) \prod_{t=1}^{T-1} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t, s_t), \tag{1}$$

where $p(s_1)$ is the initial state distribution, $p(s_{t+1}|s_t, a_t)$ is the transition probability, and $\pi_\theta(a_t|s_t)$ is the policy's action distribution conditioned on s_t.

At each time step t, the robot observes $o_t \in \Omega$, selects an action a_t, and receives a reward $r_t = \mathcal{R}(s_t, a_t)$. The goal is to optimize π_θ to maximize the expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim P(\tau)} \left[\sum_{t=1}^{T-1} \gamma^t \mathcal{R}\left(s_t, a_t\right) \right]. \tag{2}$$

This objective drives the robot to learn a collision-free, efficient navigation policy without reliance on a global map.

C. Environmental Perception

Accurate, real-time environmental perception is fundamental to autonomous navigation. While sensors such as cameras and ultrasonic devices are widely used, 2D LiDAR stands out for its high-precision distance measurements and robustness in dynamic settings. In this work, the robot is equipped solely with a lightweight 2D LiDAR sensor.

The LiDAR provides raw observations $o_l = [l_1, l_2, \ldots, l_{N_l}]$, where N_l is the number of angle-distance pairs in a 360° scan. Each $l_i \in [l_{\min}, l_{\max}]$ represents the distance at angle i, bounded by the sensor's minimum ($l_{\min}$) and maximum ($l_{\max}$) range. To facilitate decision-making, o_l is preprocessed into a feature vector Λ_l:

$$\Lambda_l = \left[\min \left(l_{\max}, \frac{1}{n_\alpha} \sum_{\alpha=1}^{n_\alpha} l_{\alpha,\beta} \right) \right]_{\beta=1}^{n_\beta}, \tag{3}$$

where the N_l measurements are partitioned into n_β sectors, each containing $n_\alpha = N_l/n_\beta$ data points. Here, $l_{\alpha,\beta}$ is the distance measured at the α-th angle in the β-th sector, and the averaging reduces noise while preserving spatial information. The resulting Λ_l serves as a primary component of the observation o_t, enabling the robot to assess its surroundings.

IV. CRAR Navigation Approach

Mapless navigation aims to enable robots to navigate safely and efficiently in unknown environments using only local sensor observations. However, during the initial phases of reinforcement learning (RL), robots often experience frequent collisions due to underdeveloped policies, resulting in unsafe and inefficient training. To address this, we propose the Coherent Risk-Aware Regulation (CRAR) framework, integrating Risk Prediction, Measurement, and Correction module to enhance safety and efficiency. The framework is illustrated in Fig. 1.

A. Risk Prediction

Given the emphasis on lightweight, flexible robots, CRAR relies exclusively on 2D LiDAR for environmental perception. A core component of CRAR is the Risk Prediction module, processing sequential LiDAR data to anticipate future hazards, enabling proactive decision-making, as described in Fig. 2. The input comprises three consecutive LiDAR frames, representing observations at timesteps $t-2$, $t-1$, and t. These frames are processed through two parallel channels:

1) **Obstacle Feature Channel**: The LiDAR sequence is fed into two 1D convolutional layers with filter sizes of 8 and 16, kernel sizes of 5, and strides of 2. The resulting features pass through an attention module to emphasize critical regions, followed by pooling and flattening, before entering a fully connected (FC) layer.
2) **Robot State Channel**: Six discrete features are extracted from the current LiDAR frame and robot state, including linear velocity, angular velocity, distance and angle to the target, and distance and angle to the nearest obstacle. These are processed through two FC layers with 32 and 64 units, respectively.

The outputs from both channels are concatenated and fused via an additional FC layer, yielding a predicted risk values

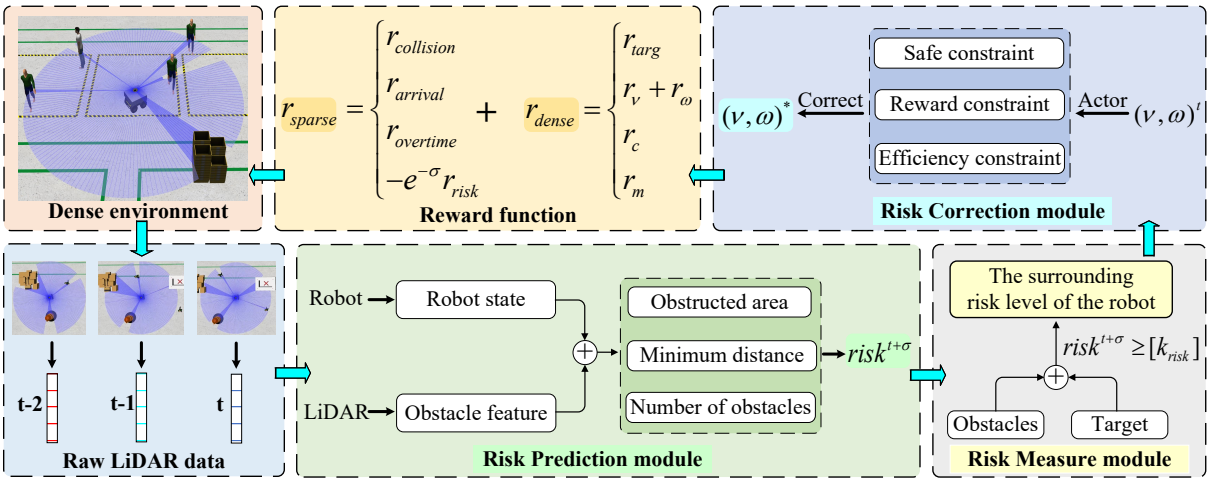


Fig. 1. Framework of the Coherent Risk-Aware Regulation (CRAR) approach for mapless navigation.

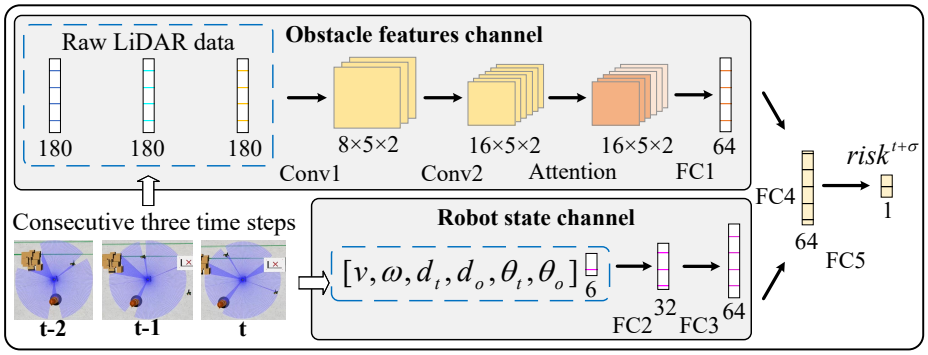


Fig. 2. Structure of the risk prediction network in CRAR.

$risk(s^t, s^{t-1}, s^{t-2})$ for a future timestep $t+\sigma$. The network is trained to minimize the mean squared error between predicted and real risk values:

$$L_{risk} = \frac{1}{N}\sum_{i\in N}\left(risk_i^{real}(s^{t+\sigma}) - risk_i(s^t, s^{t-1}, s^{t-2})\right)^2, \tag{4}$$

where N is the batch size, and $s^{t+\sigma}$ is the state σ steps ahead. As training progresses, the network's predictive accuracy improves, allowing it to forecast risk over longer horizons (larger σ). This extended foresight equips the robot with ample time to respond to potential threats.

The real risk $risk^{real}(s^{t+\sigma})$ is collected online during the interaction between the DRL model and the training environment. Accurately speaking, it is computed as a weighted combination of three metrics: μ_{area} is the area obstructed by obstacles within the LiDAR range, reflecting spatial congestion; n_o represents the risk caused by the presence of more obstacles; and l_{leng} quantifies proximity to obstacles via the shortest LiDAR ray. The three metrics dynamically reflect the safety of the robot's surrounding environment.

$$risk^{real}(s^{t+\sigma}) = \mu_{area}^{t+\sigma} + l_{leng}^{t+\sigma} + n_o^{t+\sigma}, \tag{5}$$

where n_o increases as the number of surrounding obstacles n_{obst} increases. It is defined as follows:

$$n_o = k_{num}\cdot(n_{obst} - 1). \tag{6}$$

Additionally, l_{leng} increases as the length of the shortest LiDAR ray $\min(l)$ decreases:

$$l_{leng} = \begin{cases} k_l \cdot 1/\min(l), & \min(l) < l_{crit} \\ 0, & l_{crit} \le \min(l) \le l_{\max} \end{cases}, \tag{7}$$

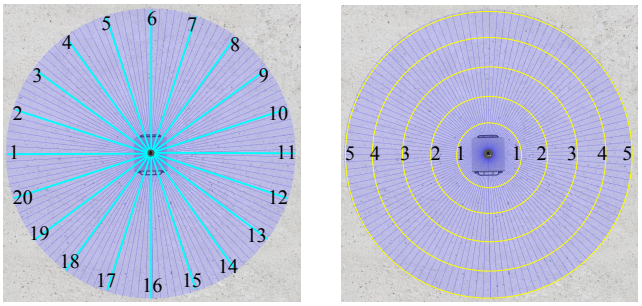
where l_{crit} denotes the critical LiDAR ray length threshold for risk. If $\min(l) \in [l_{crit}, l_{\max}]$, the LiDAR length does not pose a risk to the robot for the time being. And l_{crit} is defined as $l_{crit} = k_{crit}\cdot l_{\max}, k_{crit} \in [0,1]$.

The parameters k_{num} and k_l are dynamically adjusted weights designed to address different hazardous situations. When n_{obst} increases, k_{num} increases; when $min(l)$ becomes too short, k_l increases. This dynamic adjustment reflects the risk level of the environment. When the predicted risk exceeds a threshold, the Risk Measurement module is triggered, enabling the robot to proactively identify safer navigation regions.

B. Risk Measurement

After developing the risk prediction network, the robot assesses action-related risk during navigation. As training progresses, it learns safer, more effective policies. Early on, frequent collisions slow experience accumulation in the replay buffer, delaying training. To address this, we introduce the Risk Measurement module. It evaluates risk distribution within the LiDAR range to identify low-risk areas, thereby enhancing experience collection and accelerating policy optimization. Its design is shown in Fig. 3. The LiDAR's circular field, depicted in blue, is discretized into a grid for risk assessment. A cyan radial line of length $l_{\max}$ divides the area into $part$ equal angular sectors, numbered to indicate their positions, while yellow concentric rings partition the area into $layer$ radial layers, equally spaced and similarly numbered. This creates a grid of $layer \times part$ regions, with each grid in a given

layer having equal area. In this study, we set $layer = 5$ and $part = 20$, yielding 100 grid cells.



(a) sector division (b) layer division

Fig. 3. Schematic of the Risk Measurement module.

We assign a score to each of the grid cell based on their risk levels. The purpose is to evaluate the score of each grid cell, enabling the robot to identify regions with relatively lower risk. In this paper, each grid cell is assigned a risk scores:

$$score(i,j) = \Psi_{lidar,i,j} + \Psi_{angle,i,j} - \Psi_{area,i,j}, \tag{8}$$

where $\Psi_{lidar,i,j}$ reflects LiDAR ray lengths within grid (i,j), $\Psi_{area,i,j}$ quantifies the occluded area, and $\Psi_{angle,i,j}$ accounts for alignment with the target. As a result, we guide the robot to move along safer and more efficient paths by assigning scores to different regions.

The LiDAR-based term $\Psi_{lidar,i,j}$ balances proximity and overall obstacle presence:

$$\begin{aligned}\Psi_{lidar,i,j} = k_{min} \cdot \min_{h\in[1,n_l]}(l_{i,j,h}) \\ + (1-k_{min}) \cdot \left[\sum_{h=1}^{n_l} l_{i,j,h} - \min_{h\in[1,n_l]}(l_{i,j,h})\right],\end{aligned} \tag{9}$$

where $l_{i,j,h}$ is the length of the h-th LiDAR ray in grid (i,j), $n_l = N_l/part$ is the number of rays per sector (with N_l as the total LiDAR beams), and $k_{min} \in [0,1]$ weights the minimum ray length, with higher values emphasizing the risk of obstacles approaching. Note that $\Psi_{lidar,i,j}$ increases with shorter ray lengths, indicating higher risk.

The term $\Psi_{area,i,j}$ approximates the occluded area within grid (i,j) using geometric methods based on LiDAR intersections, though exact computation depends on obstacle shapes (details deferred to implementation). A larger occluded area elevates the risk scores.

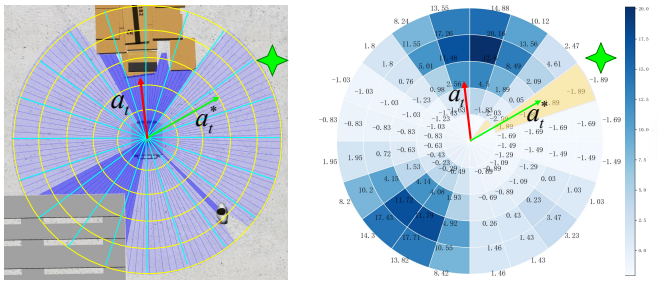
The angular term $\Psi_{angle,i,j}$ aligns navigation with the target:

$$\Psi_{angle,i,j} = |\theta_t| / \pi, \tag{10}$$

where θ_t is the angle between the robot's orientation toward grid (i,j) and the target direction, normalized by π. Smaller values indicate grids closer to the target's bearing, reducing perceived risk.

After measuring the risk across the entire LiDAR region, we obtain the risk scores for each grid within the area. These scores provide a clear visualization of the risk distribution within the robot's LiDAR range, as shown in Fig. 4.

Fig. 4(a) shows a local top view of the robot during navigation, where some obstacles and pedestrians are present around the robot. At this state, the obstacles and pedestrians



(a) (b)

Fig. 4. Risk Correction module guiding navigation: (a) local top view with obstacles; (b) risk scores heatmap and corrected path with selected grids in yellow.

are detected by the LiDAR, introducing a certain level of risk to the robot's navigation safety. Therefore, we have designed the Risk Measurement module. Through this module, we identify the location of low-risk areas that are less prone to collisions. Fig. 4(b) illustrates the corresponding risk heatmap, where darker regions (higher $score(i,j)$) represent potential hazards and a higher likelihood of collision, while lighter regions denote safer areas.

By mapping risk scores, the module identifies low-risk regions even in dense, dynamic environments. The subsequent Risk Correction module leverages these scores to adjust actions, enhancing safety and reward accumulation, as detailed in the next subsection.

C. Risk Correction

The Risk Measurement module provides a spatial risk map within the LiDAR range, enabling the identification of safer navigation paths. To accelerate strategy updates, when the robot's current action $a_t = (\nu_t, \omega_t)$ directs it toward a high-risk region, the Risk Correction module intervenes to yield a corrected action a_t^*. This adjustment aims to minimize navigation risk while aligning the robot's trajectory with rewarding, low-risk directions. In other words, under the premise of ensuring the safe navigation of the robot, the efficiency of task completion is accelerated. We formulate this as an optimization problem:

$$column^* = \arg \min_{(1\le j_1, j_2, \ldots, j_5 \le 20)} \left(\sum_{i=1}^{5} \text{score}(i, j_i)\right). \tag{11}$$

$$s.t.\ \theta(j_{layer}, j_{layer+1}) \le [\theta]. \tag{12}$$

where the solution to this optimization problem is to select a grid from each layer within the $layer \times part$ LiDAR area, based on the risk level of each grid. The requirement is to minimize the sum of the selected grid scores across all layers, thereby reducing the risk of navigation to the lowest possible level.

By continuously sorting, the optimal grid combination $column^*$ can be found. Then, the selected grid from each layer is connected to form a forward path that minimizes risk and maximizes reward for the current state. The set of selected grid column indices is denoted as:

$$column = [j_{layer=1}, j_{layer=2}, j_{layer=3}, j_{layer=4}, j_{layer=5}],$$

where variable j_{layer} represents the number of columns in the grid at the $layer$-th layer, with $j_{layer} \in [1, 20]$. The robot will navigate through the direction of each selected grid in each layer sequentially.

For example, if the grid chosen in the first layer is in the first column, then $j_{layer=1} = 1$, and the robot should turn left by $(j_r - j_{layer=1}) \cdot (360/part)$ degrees. Similarly, if the second layer is selected a grid in the third column, then $j_{layer=2} = 3$, and the robot should turn left by $(j_{layer=1} - j_{layer=2}) \cdot (360/part)$ degrees. As can be seen from Fig. 3(a), the robot's initial position is in column $j_r = 6$, and the angular difference between any two consecutive columns is $360/part$ degrees.

To decide whether to activate the correction module, we first determine if the robot has learned a reasonable obstacle avoidance strategy π^* using the following method:

$$\pi = \begin{cases} \pi^*, & k_\eta > [k_\eta] \\ \pi', & \text{otherwise} \end{cases}, \tag{13}$$

where π' indicates that the robot is still optimizing its policy and temporarily requires the Risk Correction module to assist with obstacle avoidance. The parameter k_η is used to measure the learning degree of the robot's strategy, and it is updated by:

$$k_\eta = \begin{cases} k_\eta + 1, & |\omega^* - \omega| \le k_\omega \text{ and } |\nu^* - \nu| \le k_\nu \\ 0, & \text{otherwise} \end{cases}, \tag{14}$$

where k_ω and k_ν represent a small error range. If the speed changes are minimal between consecutive time steps, it means that the robot knows how to avoid obstacles.

When $\pi = \pi'$, by correcting the current action $a_t = (\nu, \omega)$, the robot will perform a safer, more reasonable, and more efficient action $a_t^* = (\nu^*, \omega^*)$. As a result, the buffer can store more valuable experiences and accelerating learning. The definition of the correction method is as follows:

$$\omega^* = \begin{cases} -\overline{\theta} \cdot (j_{layer} - j_r), & layer = 1 \\ -\overline{\theta} \cdot (j_{layer} - j_{layer-1}), & layer > 1 \end{cases}, \tag{15}$$

$$\nu^* = \begin{cases} \nu^{t-1}(1 - |\overline{\omega}|) + k_s, & |\omega^*| \ge \left|\omega^{t-1}\right| \\ \nu^{t-1}(1 + |\overline{\omega}| + k_p, & \text{otherwise} \end{cases}. \tag{16}$$

In Equation 15, $\overline{\theta} = (360/part) \cdot (\pi/180)$, this parameter is used to calculate the angular velocity required for rotation between two adjacent grids at the current moment. Additionally, the outermost "-" is related to the robot's rotation direction. The expression $(j_{layer} - j_{layer-1})$ is used to calculate the number of columns that differ from the selected grid between two adjacent layers.

In Equation 16, the parameter $\overline{\omega}$ is defined as:

$$\overline{\omega} = \left(\omega^* - \omega^{t-1}\right) / \omega^{t-1}, \tag{17}$$

where ω^{t-1} represents the angular velocity output from the previous time step. Based on the adjusted angular velocity ω^*, we will correspondingly adjust the linear velocity according to the current change in angular velocity $\overline{\omega}$. The parameter k_s is used to prevent stalling caused by an abrupt stop during deceleration, with its value typically being small. Contrastly, the parameter k_p prevents insufficient linear velocity during acceleration, and is generally set to a larger value. After the action adjustment, if the adjusted value exceeds a reasonable range, action clipping is performed:

$$\nu^* = \begin{cases} 2, & \nu^* \ge \nu_{\max} \\ 0, & \nu^* < 0 \end{cases}, \tag{18}$$

$$\omega^* = \begin{cases} 2, & \omega^* \ge \omega_{\max} \\ -2, & \omega^* < -\omega_{\max} \end{cases}. \tag{19}$$

Moreover, to ensure smooth output for the robot's actions, we introduce an angular constraint in Equation 12, where $\theta(j_{layer}, j_{layer+1})$ represents the angle between the grids of two adjacent layers. If the $[\theta]$ is too large or too small, it may lead to unnecessary movements or difficulty in turning.

Based on the robot navigation example from the previous section, the reasonable path for the current state is illustrated in the Fig. 4(b). We can see that the selected grid positions for each layer are marked in yellow. The five grid positions are [(0, 7), (1, 8), (2, 8), (3, 8), (4, 8)], so the reasonable path, represented by the column indices of the grids, is $column = [7, 8, 8, 8, 8]$.

The pseudocode for the Risk Correction module is shown in Algorithm 1. When $k_\eta > [k_\eta]$, it indicates that the model has already learned to output appropriate actions to complete the navigation task. Therefore, even if the predicted value $risk^{t+\sigma}$ exceeds the set threshold, the module does not need to be activated for correction. Similarly, in both the simulation tests and real-world deployment, the robot makes decisions solely based on the previously learned policy.

Additionally, in order to enable the robot to respond quickly to dynamic changes, the Risk Prediction module adjusts the predicted moments dynamically. In the training process, for instance, the prediction network first learns to predict the risk for the state s_{t+1}. When the prediction accuracy ε^{t+1} reaches the required threshold, it will then predict the risk for the state s_{t+2}, and so on, enabling multi-step prediction. Specifically, when the network cannot accurately predict the risk for the state $s_{t+\sigma}$, it will decide whether to activate the correction module based on the current state's risk.

Algorithm 1: Process of the Risk Correction Module.

Input: Parameter k_η, action a_t under state s_t, $score(i, j)$ of the grid, the accuracy ε of the prediction network, and the predicted value $risk^{t+\sigma}$ under state $s_{t+\sigma}$.
Output: The action a_t^* after module correction.

```
for σ = 3 to 0
    if k_η ≤ [k_η]
        if (ε^{t+σ} ≤ [ε^{t+σ}] and risk^{t+σ} ≥ [k_risk]) or
          risk^t ≥ [k_risk]
            Activate the Risk Correction module;
            Determine a_t^* = (ω^*, ν^*) by solving equation using
              score(i, j);
            if |ω^* − ω_t| ≤ [k_ω] and |ν^* − ν_t| ≤ [k_ν]
                k_η = k_η + 1;
```

D. DRL Models

1) Observation Space: The robot's observation space $o = \{o_l, o_t, o_v\}$ consists of LiDAR-data o_l and two discrete states of the robot, o_t and o_v. For the discrete states, o_t includes ρ and ϕ, where ρ represents the Euclidean distance between the robot and the target, and is defined as follows:

$$\rho = \sqrt{(x_c - x_t)^2 + (y_c - y_t)^2}, \tag{20}$$

where (x_c, y_c) and (x_t, y_t) represent the robot's current position and target position. ϕ denotes the angle between the heading direction of the robot and the target location:

$$\phi = \text{sign}(\overrightarrow{v_r} \times \overrightarrow{v_t}) \cdot \arccos\left(\overrightarrow{v_r} \cdot \overrightarrow{v_t} / (|\overrightarrow{v_r}|\,|\overrightarrow{v_t}|)\right), \tag{21}$$

where $\phi \in [-\pi, \pi]$, and the sign function is used to determine the robot's rotation direction ("-" for clockwise, "+" for counterclockwise), $\overrightarrow{v_r}$ and $\overrightarrow{v_t}$ are the vectors representing the robot's heading direction and its target position, respectively. The other state $o_v = \{\nu, \omega\}$ indicates the actions taken by the robot, including linear velocity ν and angular velocity ω.

To accelerate the training process, we normalize each element in the observation space. Specifically, the values related to distance, o_l and ρ, are normalized by dividing by $l_{\max}$ and $\rho_{\max}$, respectively, as follows:

$$\begin{cases} l_i' = l_i / l_{\max} \\ \rho' = \rho / \rho_{\max} \end{cases}, \ i \in \{1, 2, \cdots, N\}, \tag{22}$$

where $\rho_{\max}$ is the diagonal length of the environment. The angle-related value ϕ are normalized by dividing π:

$$\phi^{'} = \phi / \pi. \tag{23}$$

The velocity-related values $\{\nu, \omega\}$ are normalized by dividing their corresponding maxima as follows,

$$\begin{cases} \nu' = \nu / \nu_{\max} \\ \omega' = \omega / \omega_{\max} \end{cases}. \tag{24}$$

Similarly, the risk value is normalized by dividing by the maximum value:

$$risk^{'} = \begin{cases} risk^{real} / risk_{max}, & \varepsilon \geq [\varepsilon] \\ risk^{pre} / risk_{max}, & \text{otherwise} \end{cases}. \tag{25}$$

Finally, the state at time step t can be described as:

$$s_t = [l_1', l_2', \cdots, l_N', \rho', \phi', \nu_t', \omega_t', risk^{'}]. \tag{26}$$

2) Action Space: The robot's continuous action space $\mathcal{A}$ consists of two components: the linear velocity $\nu \in [0, 2.0]$m/s and the angular velocity $\omega \in [-2.0, 2.0]$rad/s. Based on its current state, the robot takes an appropriate action from the action space.

3) Reward Function: To ensure safe, quick, and smooth navigation, we design a reward function to guide the robot's movement. The total reward is divided into sparse rewards r_{sparse} and dense rewards r_{dense}:

$$R = r_{sparse} + r_{dense}. \tag{27}$$

r_{sparse} take into account the termination conditions of the robot during navigation. If the robot's distance to an obstacle or the target is smaller than a defined safety threshold d_{end}, the robot will receive a high penalty $r_{collision}$ or reward $r_{arrival}$. Additionally, if the predicted risk exceed a threshold $[k_{risk}]$ for consecutive t_{cons} time steps, the robot will be penalized with a risk aggravation penalty $-e^{-\sigma} \cdot r_{risk}$. Furthermore, when the navigation task exceeds the predefined time duration $[t]$, a penalty $r_{overtime}$ is also applied.

$$r_{sparse} = \begin{cases} r_{collision}, & d_o \leq d_{end} \\ r_{arrival}, & d_t \leq d_{end} \\ r_{overtime}, & [t] \leq t_t \\ -e^{-\sigma} \cdot r_{risk}, & [k_{risk}] \leq risk^{t+\sigma} \end{cases}, \tag{28}$$

where d_t and d_o represent the distance between the robot and the target, and the robot and the obstacle, respectively. t_t denote the time taken to reach the target.

However, r_{sparse} merely reflects the basic navigation objective and fails to effectively incentivize environmental exploration by the robot. Therefore, we have designed r_{dense} that provide feedback at each time step before the navigation task is completed, encouraging the robot to move toward the target. This not only accelerates the training process but also makes the trajectory smoother:

$$r_{dense} = r_t + r_c + r_v + r_m, \tag{29}$$

where r_t denotes a goal-oriented reward and is defined as:

$$r_t = r_t^d + r_t^\theta + r^\theta, \tag{30}$$

$$\begin{cases} r_t^d = k_d \cdot (d_t^{t-1} - d_t) / \max(d) \\ r_t^\theta = k_a \cdot (\theta_t^{t-1} - \theta_t) / \max(\theta) \\ r^\theta = -k_o \cdot |\theta_t| / \pi \end{cases}, \tag{31}$$

where d_t^{t-1} and θ_t^{t-1} represent the distance and angular deviation from the target at the moment of $t-1$, respectively. $\max(d)$ and $\max(\theta)$ are the maximum distance and angle that the robot can change in one step.

r_c is the obstacle-related reward. The robot is penalized when its distance to an obstacle is below a predefined threshold l_{crit}. To avoid collisions, the robot must adjust its direction in time when approaching obstacles and move away from them. r_c is defined as:

$$r_c = \begin{cases} k_b \cdot (d_o - d_o^{t-1}) / \max(d), & d_o \leq d_o^{t-1} \leq l_{crit} \\ 0, & \text{otherwise} \end{cases}. \tag{32}$$

r_v is the velocity-related reward. To complete the navigation task quickly and smoothly, the robot should maintain a high linear velocity and a low angular velocity. r_v is defined as:

$$r_v = r_\nu^v + r_\omega^v. \tag{33}$$

$$\begin{cases} r_\nu^v = k_r \cdot (\nu / \nu_{\max}) \\ r_\omega^v = -k_r \cdot |\omega / \omega_{\max}| \end{cases}, \tag{34}$$

Lastly, at each time step, the robot will receive a slight penalty r_m to speed up the learning process and reduce the occurrence of timeouts.

These designs of r_{sparse} and r_{dense} enable the robot to improve navigation efficiency while ensuring safety, achieving effective and smooth goal-directed behavior.

V. Experiments and Results

A. Experiment Setup

1) Hardware: Experiments are conducted using the Gazebo simulator integrated with the Robot Operating System (ROS), leveraging Python for implementation. Training and testing occur on a system with a 13th Gen Intel(R) Core(TM) i7-13620H CPU and an NVIDIA GeForce RTX 4060 Laptop GPU. The simulated robot is equipped with a 180-line 2D LiDAR providing a 360-degree field of view. Key robot parameters are listed in Table I.

TABLE I
ROBOT MODEL PARAMETERS.

Parameter	Value	Parameter	Value
$\nu_{\max}$	2 m/s	$\max(\theta)$	0.2 rad/s
$\omega_{\max}$	2 rad/s	$\max(d)$	0.2 m
l_{risk}	0.6 m	N	512
l_{max}	3.5 m	N_l	180
l_{min}	0.12 m	n_α	6

2) Scenarios: Three environments, illustrated in Fig. 5, are developed to evaluate CRAR's generalization and robustness. Panels (a), (b), and (c) represent the training environment, test environment 1, and test environment 2, respectively, with panels (d), (e), and (f) providing their top-down views. The training environment, a 40 × 60 m space, replicates a dense factory setting. It features 23 static obstacles—including shelves, pallets, and stationary workers—and 9 dynamic pedestrians. Pedestrian trajectories are pre-defined and intersect with the robot's common paths, allowing pedestrians to walk repeatedly along the set routes to reflect real-world dynamics. Static obstacles, diverse in shape and randomly positioned, include elements like large yellow shelves that permit the robot to pass underneath.

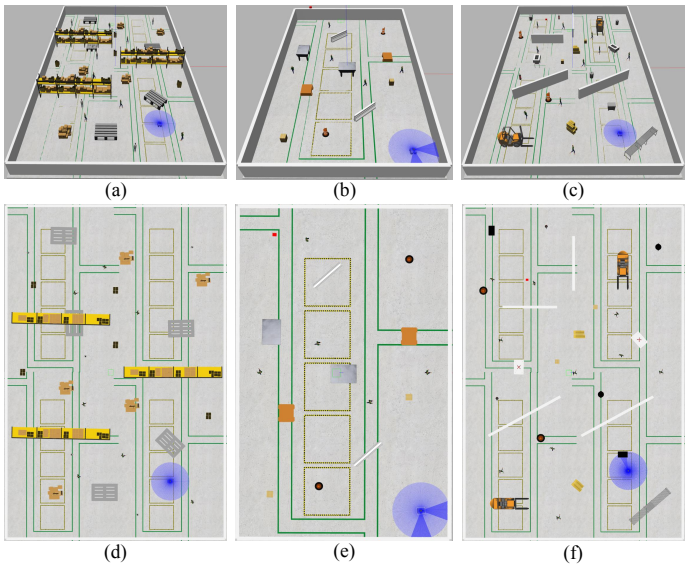


Fig. 5. Simulation environments: (a, d) training environment; (b, e)test environment 1; (c, f) test environment 2. Top row shows 3D views; bottom row shows top-down views.

Two test environments are designed to assess the robot's performance, featuring obstacles distinct from those in the training environment and unfamiliar to the robot. Test environment 1 (20 × 30 m) includes 5 pedestrians and 10 obstacles, with 5 of the obstacles capable of sudden movement, testing the robot's ability to handle unexpected situations. The speeds of both pedestrians and moving obstacles range from $[0, 2\sqrt{2}]$ m/s, while the robot's maximum speed is 2 m/s. This disparity suggests potential collisions as pedestrians may outpace the robot. Conversely, Test environment 2 (40 × 60 m) introduces 10 dynamic pedestrians and 20 obstacles (5 of which can also move suddenly). In this environment, dense obstacles, impassable walls, and a higher concentration of entities near the target elevate navigation complexity. Additionally, the pedestrian and obstacle speeds are adjusted to $[0, 2]$ m/s, mirroring human social behavior in confined spaces.

The robot's starting position and target are randomly generated within specified ranges. A trial concludes if the robot approaches within 1 m of the target or an obstacle, or exceeds the navigation time limit. To simplify decision-making, the robot is rendered "invisible" to pedestrians, who do not actively avoid it.

3) Overhead analysis: This section analyzes the computational overhead and response latency of CRAR to demonstrate its response speed. The analysis aims to elucidate the fundamental trade-off between its predictive intelligence and real-time efficiency when compared to purely reactive methods. Table II details the average execution time and proportion of each key step per control cycle, providing a data foundation for the subsequent analysis.

TABLE II
OVERHEAD CALCULATIONS FOR TRAINING.

(a) The training overhead of the CRAR algorithm.

CRAR	RC	RP	RMC	AE	DT	PNT
Time/ms	0.30	1.10	2.20	119.20	40.30	56.30
Rate/%	0.14	0.46	1.00	54.35	18.38	25.67

(b) The training overhead of the TD3 algorithm.

TD3	RC	RP	RMC	AE	DT	PNT
Time/ms	\	\	\	108.51	33.60	\
Rate/%	\	\	\	75.98	23.53	\

The key steps are listed in the table as follows:

- **RC:** Computes the instantaneous risk based on the current LiDAR scan.
- **RP:** Predicts future risk via a neural network.
- **RMC:** The core of risk regulation, assessing risk distribution and guiding the robot towards lower-risk areas.
- **AE:** Applies the final control command, i.e., the model's output or a corrected action.
- **DT & PNT:** Used for updating the policy model and the risk prediction network. This process belongs to background training and is not on the critical real-time control path.

Based on the Table II data, we evaluate the system's real-time performance. During operation, the average duration of a single "perception-decision-action" cycle (i.e., RC/RP + RMC + AE) ranges from 121.7 to 122.5 ms. Consequently, the embedded platform can stably maintain a control frequency of approximately 8 Hz.

To validate the sufficiency of this frequency in dynamic environments, we consider the worst-case scenario. In our test scenarios, the maximum speed of pedestrians and obstacles is 2.8 m/s. Thus, the maximum positional uncertainty caused by the system's periodic latency is: 0.1225 × 2.8 = 0.34 m. Since this value is less than 10% of the LiDAR's maximum detection range, it indicates that the robot is fully capable of handling abruptly appearing obstacles in the scene.

Finally, we compare CRAR with reactive methods that strive to minimize computation for instantaneous response. CRAR invests limited computational resources to obtain proactive

intelligence, as RC/RP + RMC = 2.5 to 3.3 ms, accounting for less than 2% of the total step duration. Therefore, this modest overhead is exchanged for a significant improvement in navigation safety and trajectory smoothness. This trade-off adequately meets the dual requirements of system performance and efficiency for practical deployment.

4) Metrics: To assess the proposed method, we compare it against established methods—ORCA [7], DDPG [32], TD3 [33], SESN [34] and O-TD3 [35]—in two test environments of varying densities, evaluating map-free navigation and obstacle avoidance capabilities. Each method is tested over 100 episodes in identical navigation tasks, with performance averaged across robot behavior. Note: these baselines are trained without the prediction network or risk reward, distinguishing them from our approach CRAR. The following metrics are employed:

- **SR:** Ratio of successful target reaches to total episodes.
- **CR:** Ratio of collisions with pedestrians or static obstacles to total episodes.
- **TR:** Ratio of instances exceeding the maximum navigation time to total episodes.
- ρ**T:** Average time (in seconds) to reach the target.
- ρ**L:** Average path length (in meters) traveled to the target.
- ρ**D:** Average minimum distance (in meters) to pedestrians or obstacles during navigation.
- ρ**C:** Average curvature (in meters) of trajectories, where lower values indicate smoother paths, reflecting adaptability and reduced collision risk from sharp turns.

These metrics collectively evaluate the robot's navigation efficiency, safety, and adaptability in dense settings.

B. Quantitative Performance

1) Comparison Training: In the same simulation environment, we trained different algorithms for 1,000 episodes using identical hyperparameters, and the results from multiple runs are presented in Fig. 6. Since the ORCA algorithm does not require training, it was excluded from comparison. First, the Fig. 6 shows that CRAR achieves high rewards even in the early stages of training, benefiting from the Risk Correction module, guiding the robot to avoid obstacles and collect high-quality experiences, thereby accelerating the training process. Then, by around the 100-th episode, the robot has gradually acquired navigation skills, at which point the Risk Correction module is deactivated. Next, CRAR not only converges approximately 100 episodes earlier than the other algorithms but also attains higher average rewards. Finally, the shaded regions represent the standard deviation of the average rewards. Thus, the narrower shading of CRAR indicates greater robustness. While DDPG and TD3 achieve similar average rewards, DDPG exhibits less stable performance. Compared to O-TD3, SESN optimizes its policy at a faster rate.

2) Comparison Results: Table III (a) and (b) compare our CRAR method with ORCA, DDPG, TD3, O-TD3 and SESN in test environments 1 and 2. Overall, CRAR outperforms the baselines in both settings, achieving higher success rates, shorter navigation times, and smoother trajectories. Although its path length is not the shortest, CRAR prioritizes safety by selecting lower-risk routes, enabling proactive risk prediction and mitigation as shown in Fig. 7.

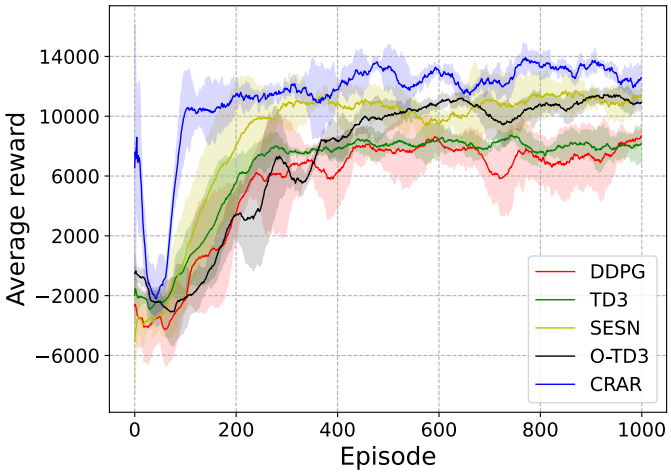


Fig. 6. Comparison of the episode rewards of different strategies in training.

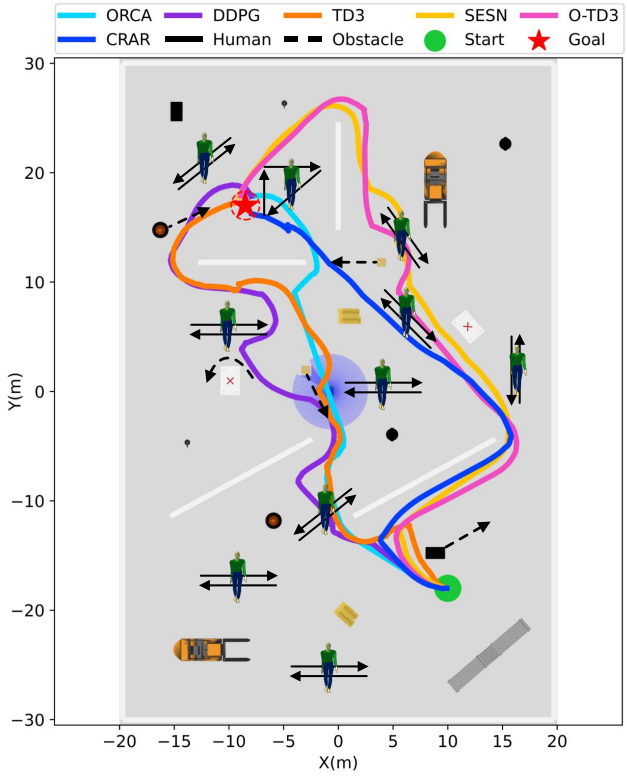


Fig. 7. Trajectories of different policies in scenes with 10 humans and 20 obstacles.

Specifically, ORCA favors the shortest central path, minimizing path length but compromising success rate and increasing timeouts due to dense obstacles. As obstacle density rises, ORCA struggles to balance brevity and safety. In contrast, our CRAR navigates via the less obstructed right side, thereby enhancing success rates and efficiency by reducing risk. Notably, TD3 and DDPG produce less smooth paths; in real-world scenarios, their frequent lateral oscillations could confuse pedestrians and elevate accident risk. In test environment 2, SESN requires excessive navigation time, while O-TD3 encounters timeouts, indicating that both methods are relatively limited in adapting to dense and complex environments.

Figure 8 highlights the statistical differences in motion characteristics (average path curvature) and social attributes (average safety distance). CRAR demonstrates the most favorable and stable average path curvature, indicated by the compact box, reflecting smoother trajectories. Regarding safety distance, although DDPG and O-TD3 maintain larger distances than CRAR, they exhibit greater variability (longer boxes), indicating less predictable trajectories that could potentially pose a threat to pedestrians.

TABLE III
QUANTITATIVE NAVIGATION RESULTS WITH DIFFERENT POLICIES.

(a) Test Environment 1 (5 pedestrians and 10 obstacles)

	SR↑	CR↓	TR↓	ρT↓	ρL↓	ρD↑	ρC↓
ORCA	0.63	0.37	**0**	58.28	46.02	1.11	0.83
DDPG	0.65	0.35	**0**	52.06	47.63	0.88	0.80
TD3	0.77	0.23	**0**	56.60	45.25	**1.37**	1.78
SESN	0.84	0.16	**0**	53.03	**43.64**	0.90	0.64
O-TD3	0.90	0.10	**0**	58.94	46.98	1.22	0.92
CRAR	**0.95**	**0.05**	**0**	**50.32**	47.30	1.23	**0.60**

(b) Test Environment 2 (10 pedestrians and 20 obstacles)

	SR↑	CR↓	TR↓	ρT↓	ρL↓	ρD↑	ρC↓
ORCA	0.39	0.55	0.06	72.71	**49.42**	0.67	0.93
DDPG	0.56	0.44	**0**	86.22	63.80	1.15	1.24
TD3	0.69	0.31	**0**	80.48	71.31	0.98	0.81
SESN	0.68	0.32	**0**	94.02	67.76	0.69	0.56
O-TD3	0.71	0.28	0.01	75.57	62.23	**1.20**	0.53
CRAR	**0.79**	**0.21**	**0**	**64.18**	57.09	0.64	**0.51**

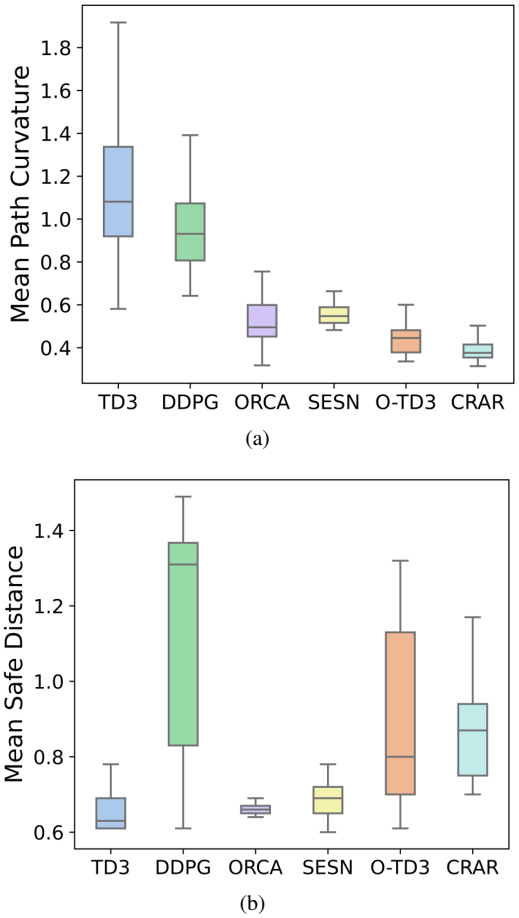


Fig. 8. Box plots: (a) path curvature; (b) safety distance across policies.

3) Ablation Experiments: To confirm the effectiveness of the proposed method, ablation experiments evaluate the contribution of added modules to navigation performance, with results presented in Table IV. Four variant models are tested for comparison: CRAR-RC (omitting risk values from the state), CRAR-RS (omitting the Risk Correction module), CRAR-SC (where the risk in the state is the instantaneous risk rather than predicted risk), and CRAR-D (constructed upon the DDPG algorithm instead of TD3).

The data in the table leads to the following analysis: 1) CRAR-RC achieves shorter path lengths at the expense of success rate, indicating that without an understanding of risk, the model tends to adopt riskier behaviors. 2) CRAR-RS sometimes achieves better ρC and ρT than CRAR model, likely due to differences in their objective functions. The core of risk correction is safety-first avoidance, prioritizing task completion through preventive adjustments such as deceleration or course changes. These adjustments can, however, lead to slight declines in ρC and ρT. 3) The notable decline in both success rate and safety distance observed with CRAR-SC confirms that the foresight provided by the prediction module is crucial for safety. 4) The performance gains of CRAR-D over DDPG demonstrate the value of the risk-aware regulation mechanism. Its comparable performance to CRAR confirms the effectiveness and generalizability of the proposed DRL model itself.

TABLE IV
ABLATION STUDY: ANALYSIS OF ROBOT NAVIGATION RESULTS IN DIFFERENT TEST ENVIRONMENTS.

(a) Test Environment 1 (5 pedestrians and 10 obstacles)

	SR↑	CR↓	TR↓	ρT↓	ρL↓	ρD↑	ρC↓
DDPG	0.63	0.37	**0**	50.91	**41.34**	1.04	1.04
TD3	0.61	0.39	**0**	52.38	41.81	0.99	0.62
CRAR-RC	0.69	0.31	**0**	55.40	46.07	1.09	0.79
CRAR-RS	0.72	0.28	**0**	54.16	45.24	1.15	0.53
CRAR-SC	0.85	0.15	**0**	56.54	55.89	0.87	0.56
CRAR-D	0.89	0.11	**0**	50.78	46.71	1.03	**0.52**
CRAR	**0.95**	**0.05**	**0**	**50.32**	47.30	**1.23**	0.60

(b) Test Environment 2 (10 pedestrians and 20 obstacles)

	SR↑	CR↓	TR↓	ρT↓	ρL↓	ρD↑	ρC↓
DDPG	0.55	0.34	0.11	95.10	70.24	0.86	0.86
TD3	0.57	0.42	0.01	74.89	**55.42**	0.75	1.16
CRAR-RC	0.61	0.39	**0**	76.24	56.63	0.80	0.66
CRAR-RS	0.75	0.25	**0**	**64.96**	58.03	**0.92**	**0.45**
CRAR-SC	0.77	0.23	**0**	71.60	58.01	0.66	0.72
CRAR-D	0.78	0.22	**0**	72.88	69.27	0.88	0.48
CRAR	**0.86**	**0.14**	**0**	71.96	62.69	0.78	**0.45**

4) Experiments with Varying Crowd Densities: To evaluate navigation performance across different pedestrian densities, we compare CRAR with ORCA, DDPG, and TD3. Dynamic pedestrian counts are set at 0, 5, and 10, with static obstacles fixed at 20. Results are detailed in Table V. CRAR consistently achieves the highest success rate with minimal variation across densities, while ORCA, DDPG, and TD3 exhibit sharp performance declines as complexity rises. ORCA's success rate drops significantly from 0 to 5 pedestrians, and DDPG and TD3 show similar declines from 5 to 10, indicating their suitability for sparser settings. In denser, map-unknown conditions, these

methods struggle with increased collision risk. Conversely, CRAR leverages environmental risk monitoring to avoid crowded zones, sustaining higher success rates. Additionally, CRAR records shorter navigation times, smoother trajectories, and greater safety distances, demonstrating robust adaptability to varying pedestrian densities with minimal performance impact.

TABLE V
RESULTS ACROSS CROWD DENSITIES.

(a) 0 pedestrians and 20 obstacles

	SR↑	CR↓	TR↓	ρT↓	ρL↓	ρD↑	ρC↓
ORCA	0.86	0.07	0.07	**54.65**	**47.65**	0.62	0.46
DDPG	0.85	0.15	**0**	58.27	56.65	1.22	1.86
TD3	0.96	0.04	**0**	62.37	60.89	0.78	0.47
SESN	0.98	0.02	**0**	83.72	69.56	0.85	0.79
O-TD3	0.96	0.04	**0**	71.75	63.82	**1.40**	0.47
CRAR	**0.99**	**0.01**	**0**	58.24	57.33	1.03	**0.41**

(b) 5 pedestrians and 20 obstacles

	SR↑	CR↓	TR↓	ρT↓	ρL↓	ρD↑	ρC↓
ORCA	0.58	0.39	0.03	69.62	**49.48**	0.65	0.67
DDPG	0.79	0.15	0.06	87.25	66.46	0.95	0.94
TD3	0.80	0.20	**0**	71.51	57.01	0.95	0.72
SESN	0.83	0.17	**0**	99.93	68.92	0.68	0.51
O-TD3	**0.90**	0.08	0.02	72.81	64.38	**1.07**	0.50
CRAR	**0.90**	**0.10**	**0**	**69.33**	65.68	0.91	**0.47**

(c) 10 pedestrians and 20 obstacles

	SR↑	CR↓	TR↓	ρT↓	ρL↓	ρD↑	ρC↓
ORCA	0.32	0.64	0.04	71.67	**49.17**	0.67	0.71
DDPG	0.54	0.46	**0**	93.32	61.78	1.14	1.33
TD3	0.70	0.30	**0**	81.02	71.33	1.10	0.82
SESN	0.65	0.35	**0**	92.50	68.41	0.69	0.55
O-TD3	0.74	0.21	0.05	81.21	62.10	**1.16**	0.48
CRAR	**0.81**	**0.19**	**0**	**69.87**	62.89	0.82	**0.44**

C. Real-World Experiments

1) Experimental Configuration: In addition to the simulation experiments, we also conduct real-world tests to evaluate the feasibility and robustness of the proposed approach. The experiments are carried out using a SCOUT2.0 four-wheel differential drive robot equipped with an RS-Helios-16P 3D LiDAR. To ensure consistency with the simulation setup, only 2D LiDAR data from the plane corresponding to the LiDAR installation height are utilized. All robot action commands are computed externally by a computer. To ensure safe and stable navigation in the real world, the robot's linear and angular velocities are constrained to one-quarter of the maximum simulation speed. No additional Sim2Real adaptation techniques are applied, as the differences between the simulation and real-world environments are minimal for the tasks considered in our experiments.

The real-world testing environment, depicted in Fig. 9, measures 21×27 meters and includes multiple static obstacles. Additionally, pedestrians are introduced to interfere with the robot's navigation. To evaluate the robot's decision-making under uncertainty, we include two suspected traversable trap obstacles, simulating the traversable elements present in the simulation. However, in practice, the height of the trap obstacles marked in Fig. 9 (allowing robots under 50 cm to pass through) does not allow passage for the robot used in this experiment (which has a height of 100 cm). This setup aims to test whether the robot would mistakenly attempt shortcuts and collide with these deceptive obstacles. Further testing involving confirmed traversable obstacles is shown in Fig. 10.

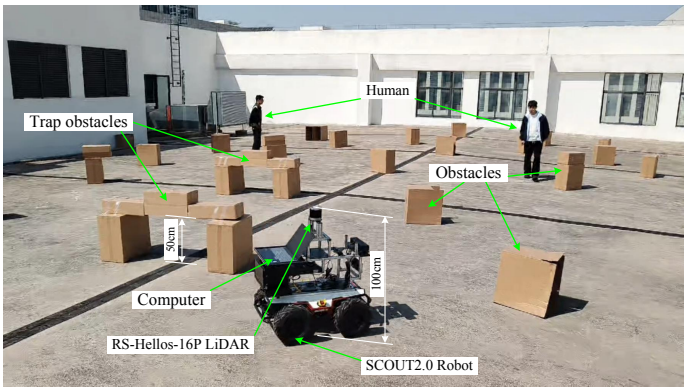


Fig. 9. Setup of the real-world experimental scenario.

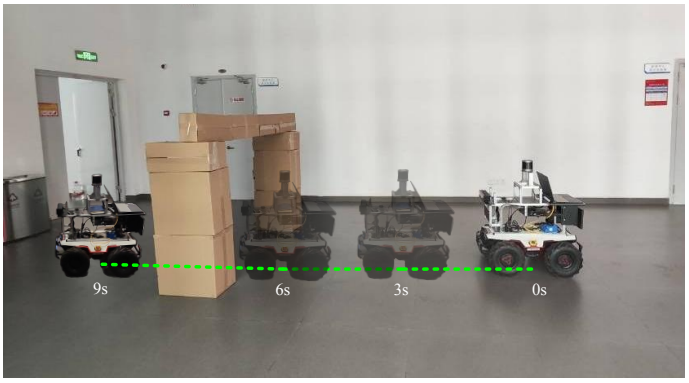


Fig. 10. Testing of traversable obstacles in the real-world environment.

2) Performance Analysis: The visual results of the experiment are presented in Fig. 11, illustrating the robot's and pedestrians' movement trajectories at various time intervals. The robot starts from the right side of the map, following the green dashed line toward the target area on the left, marked in green. Pedestrians 1 and 2 remain stationary throughout the experiment, while pedestrians 3, 4, and 5 begin moving at 15, 25, and 30 seconds, respectively, thereby introducing dynamic interference. Colored arrows indicate the direction of each pedestrian's movement. As shown in Fig. 11, the robot successfully perceives and interprets its surroundings, makes rational navigation decisions, and maintains a smooth trajectory. It consistently keeps a safe distance from both static and dynamic obstacles, demonstrating effective and reliable mapless navigation.

VI. CONCLUSION

This paper presents the Coherent Risk-Aware Regulation (CRAR) framework, a deep reinforcement learning (DRL)-based method for mapless robotic navigation in dense, dynamic environments. CRAR integrates risk prediction, measurement, and correction to reduce early-training collisions,

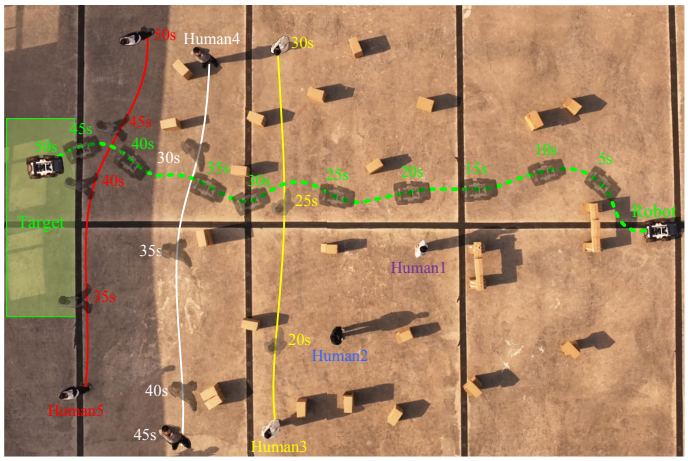


Fig. 11. Navigation test in the real-world environment with 5 pedestrians and 22 obstacles.

accelerate policy convergence, and enhance obstacle avoidance. The Risk Prediction module uses sequential LiDAR data to foresee hazards, enabling proactive decisions, while the Risk Measurement and Correction modules jointly steer the robot toward safer, more efficient paths. Experiments in diverse scenarios demonstrate that CRAR method outperforms the baseline, achieving higher success rates (e.g., 95% in test environment 1), shorter navigation times (e.g., 50.32 seconds), and smoother trajectories (e.g., ρC = 0.60). CRAR also maintains larger safety distances and adapts well to rising crowd densities, highlighting its practical value.

However, limitations persist. First, the risk definition, based on sequential LiDAR geometry, is inherently limited in capturing semantic information. It cannot infer fine-grained intent (e.g., a pedestrian's imminent turn), assess terrain properties (e.g., slippery floors), or recognize semantically hazardous areas (e.g., construction sites). Second, systematic boundaries exist under severe sensor degradation/noise or when encountering highly adversarial pedestrians moving far beyond the trained speed distribution. Finally, a trade-off persists between the prediction module's accuracy and training efficiency, and the persistent activation of the Correction module post-convergence maybe hinder decision.

Accordingly, future work will focus on evolving CRAR into a more general and scalable risk-aware navigation framework. Key directions include: 1) enriching risk perception by integrating multi-modal sensors (e.g., vision) to enable semantic understanding and fine-grained intent prediction; 2) enhancing system robustness through multi-sensor fusion and adversarial training to handle sensor failure and challenging interactions; and 3) developing adaptive control mechanisms, for instance, by dynamically adjusting the intervention threshold of the correction module, to better balance safety, efficiency, and policy autonomy.

REFERENCES

[1] Y. Bai, B. Zhang, N. Xu, J. Zhou, J. Shi, and Z. Diao, "Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review," *Computers and Electronics in Agriculture*, vol. 205, p. 107584, 2023.

[2] V. A. Hajimahmud, A. Khang, V. Hahanov, E. Litvinova, S. Chumachenko, and A. V. Alyar, "Autonomous robots for a smart city: Closer to augmented humanity," in *AI-Centric Smart City Ecosystems*. CRC Press, 2022, pp. 111–122.

[3] E. Poornima, B. Muthu, R. Agrawal, S. P. Kumar, M. Dhingra, R. R. Asaad, and A. K. Jumani, "Fog robotics-based intelligence transportation system using line-of-sight intelligent transportation," *Multimedia Tools and Applications*, pp. 1–29, 2023.

[4] P. E. Dupont, B. J. Nelson, M. Goldfarb, B. Hannaford, A. Menciassi, M. K. O'Malley, N. Simaan, P. Valdastri, and G.-Z. Yang, "A decade retrospective of medical robotics research from 2010 to 2020," *Science Robotics*, vol. 6, no. 60, p. eabi8017, 2021.

[5] Z. Han, P. Chen, B. Zhou, and G. Yu, "Real-time navigation of unmanned ground vehicles in complex terrains with enhanced perception and memory-guided strategies," *IEEE Transactions on Vehicular Technology*, 2025.

[6] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1928–1935.

[7] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*. Springer, 2011, pp. 3–19.

[8] A. J. Sathyamoorthy, U. Patel, T. Guan, and D. Manocha, "Frozone: Freezing-free, pedestrian-friendly navigation in human crowds," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4352–4359, 2020.

[9] J. Cai, A. Du, X. Liang, and S. Li, "Prediction-based path planning for safe and efficient human–robot collaboration in construction via deep reinforcement learning," *Journal of Computing in Civil Engineering*, vol. 37, no. 1, p. 04022046, 2023.

[10] F. Leon and M. Gavrilescu, "A review of tracking and trajectory prediction methods for autonomous driving," *Mathematics*, vol. 9, no. 6, p. 660, 2021.

[11] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[12] Y. Zhang, W. Feng, Z. Yang, Z. Zhou, Z. Zhu, and W. Wang, "Visual navigation of mobile robots in complex environments based on distributed deep reinforcement learning," in *2022 6th Asian Conference on Artificial Intelligence Technology (ACAIT)*. IEEE, 2022, pp. 1–5.

[13] L. Sun, J. Zhai, and W. Qin, "Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 109 544–109 554, 2019.

[14] X. Lu, H. Woo, A. Faragasso, A. Yamashita, and H. Asama, "Robot navigation in crowds via deep reinforcement learning with modeling of obstacle uni-action," *Advanced Robotics*, vol. 37, no. 4, pp. 257–269, 2023.

[15] P. Chen, J. Pei, W. Lu, and M. Li, "A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance," *Neurocomputing*, vol. 497, pp. 64–75, 2022.

[16] X. V. Wang and L. Wang, "A literature survey of the robotic technologies during the covid-19 pandemic," *Journal of Manufacturing Systems*, vol. 60, pp. 823–836, 2021.

[17] C.-C. Lee, S. Qin, and Y. Li, "Does industrial robot application promote green technology innovation in the manufacturing industry?" *Technological Forecasting and Social Change*, vol. 183, p. 121893, 2022.

[18] H. Hewawasam, M. Y. Ibrahim, and G. K. Appuhamillage, "Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments," *IEEE Open Journal of the Industrial Electronics Society*, vol. 3, pp. 353–365, 2022.

[19] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 11 345–11 352.

[20] A. J. Sathyamoorthy, K. Weerakoon, T. Guan, M. Russell, D. Conover, J. Pusey, and D. Manocha, "Vern: Vegetation-aware robot navigation in dense unstructured outdoor environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 11 233–11 240.

[21] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, p. 4282, 1995.

[22] P. Ratsamee, Y. Mae, K. Ohara, T. Takubo, and T. Arai, "Human–robot collision avoidance using a modified social force model with body pose and face orientation," *International Journal of Humanoid Robotics*, vol. 10, no. 01, p. 1350008, 2013.

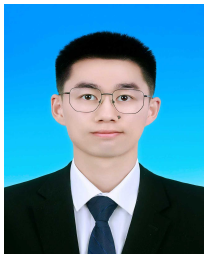
[23] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "Stgat: Modeling spatial-temporal interactions for human trajectory prediction," in *Proceedings*

of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6272–6281.
[24] A. Filotheou, A. Nikou, and D. V. Dimarogonas, "Robust decentralised navigation of multi-agent systems with collision avoidance and connectivity maintenance using model predictive controllers," *International Journal of Control*, vol. 93, no. 6, pp. 1470–1484, 2020.
[25] B. Zheng, S. Verma, J. Zhou, I. W. Tsang, and F. Chen, "Imitation learning: Progress, taxonomies and challenges," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
[26] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, p. 103500, 2021.
[27] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll, "A review of safe reinforcement learning: Methods, theories and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
[28] Q. Zhang, S. Leng, X. Ma, Q. Liu, X. Wang, B. Liang, Y. Liu, and J. Yang, "Cvar-constrained policy optimization for safe reinforcement learning," *IEEE transactions on neural networks and learning systems*, vol. 36, no. 1, pp. 830–841, 2024.
[29] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *Journal of Machine Learning Research*, vol. 18, no. 167, pp. 1–51, 2018.
[30] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.
[31] D. Dobriborsci, R. Zashchitin, M. Kakanov, W. Aumer, and P. Osinenko, "Predictive reinforcement learning: map-less navigation method for mobile robot," *Journal of Intelligent Manufacturing*, pp. 1–16, 2023.
[32] X. He, Y. Kuang, N. Song, and F. Liu, "Intelligent navigation of indoor robot based on improved ddpg algorithm," *Mathematical Problems in Engineering*, vol. 2023, no. 1, p. 6544029, 2023.
[33] B. Huang, J. Xie, and J. Yan, "Inspection robot navigation based on improved td3 algorithm," *Sensors*, vol. 24, no. 8, p. 2525, 2024.
[34] S. Tian, C. Wei, S. Jian, and Z. Ji, "Preference-based deep reinforcement learning with automatic curriculum learning for map-free ugv navigation in factory-like environments," *Engineering Science and Technology, an International Journal*, vol. 70, p. 102147, 2025.
[35] H. A. Neamah and O. A. M. Mayorga, "Optimized td3 algorithm for robust autonomous navigation in crowded and dynamic human-interaction environments," *Results in Engineering*, vol. 24, p. 102874, 2024.

Shaojie Jian received the B.E. degree in Mechanical Engineering from the School of Mechanical and Electronic Engineering, Pingxiang University, Jiangxi, China, in 2023. He is currently pursuing the M.E. degree with the College of Mechanical and Electronic Engineering, Hohai University, Changzhou, China. His main research interests include reinforcement learning, robot navigation, and avoidance collision.

Prof. Dr. Changyun Wei received the B.E. degree from Hohai University, Changzhou, China, in 2008, the M.Sc. degree from Hohai University, Changzhou, China, in 2010, and the Ph.D. degree in Artificial Intelligence from the Delft University of Technology, the Netherlands, in 2015. He is a full Professor at the College of Mechanical and Electrical Engineering, Hohai University, China. His research interests include robotics, multi-agent systems, agent technology, intelligent systems, and computer vision.

Shunyu Tian received the B.E. degree in Mechanical Engineering from the School of Mechanical and Energy Engineering, Tongji University, Shanghai, China, in 2021, and the M.E. degree in Mechanical Engineering from the College of Mechanical and Electronic Engineering, Hohai University, Changzhou, China, in 2025. He is currently pursuing the Ph.D. degree with the College of Mechanical and Electronic Engineering, Hohai University. His main research interests include robot navigation, unmanned surface vessel control, and reinforcement learning.



Xiangyu Wu received the B.E. degree in Mechanical Engineering from the College of Mechanical and Electronic Engineering, Hohai University, Changzhou, China, in 2023. He is currently pursuing the M.E. degree with the College of Mechanical and Electronic Engineering, Hohai University, Changzhou, China. His main research interests include reinforcement learning, unmanned surface vehicle, and navigation.

Dr. Ze Ji received the B.E. degree from Jilin University, Changchun, China, in 2001, the M.Sc. degree from the University of Birmingham, Birmingham, U.K., in 2003, and the Ph.D. degree from Cardiff University, Cardiff, U.K., in 2007. He is a Reader at the School of Engineering, Cardiff University. Before this, he worked on autonomous robotics in the industry (Dyson and Lenovo). His research interests include autonomous robot navigation, robot manipulation, robot learning, computer vision, simultaneous localization and mapping (SLAM), acoustic localization, and tactile sensing.