JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 38, XXX-XXX (2022)
DOI: 10.6688/JISE.20220X_38(X).00XX

Multi-Agent TD3 with Prioritized Experience Replay for Multi-Robot Flocking Control

CHING-CHIH TSAI[1], ALI ROSPAWAN[1,2], HONG-SHENG WU[1] AND WEI-KENG CHIEN[1]
[1]Department of Electrical Engineering,
National Chung Hsing University, Taichung 40227, Taiwan
[2]Department of Electronics Engineering,
Politeknik Manufaktur Negeri Bangka Belitung, Sungailiat 33211, Indonesia
E-mail: cctsai@nchu.edu.tw

This paper proposes a practically validated multi-robot flocking control framework that integrates an improved Rapidly-exploring Random Tree Star (RRT*) algorithm with a Multi-Agent Twin Delayed Deep Deterministic (MATD3) policy gradient algorithm enhanced by Prioritized Experience Replay (PER) to effectively steer and coordinate multiple omnidirectional Autonomous Mobile Robots (AMRs) in cohesive group movement. Rather than comprehensive global path planning, the improved RRT* with path optimization generates strategic waypoints from Simultaneous Localization and Mapping (SLAM) information, while the PER-MATD3 employs decentralized execution with a centralized learning policy to navigate each robot locally, ensuring robust obstacle avoidance while maintaining dynamic flocking behavior. Two theoretical guarantees are provided through convergence and stability analysis, proving that under appropriate conditions, the PER-MATD3 algorithm converges to locally optimal policies satisfying three key flocking objectives: collision avoidance, communication maintenance, and cohesive movement. Comparative simulations with a swarm of omnidirectional AMRs with extended experimental validation further confirm that the proposed approach efficiently coordinates multiple robots to achieve cooperative flocking behavior and maintain cohesive group movement through complex environments.

Keywords: Deep reinforcement learning, flocking control, improved RRT*, multi-robot, omnidirectional robots.

1. INTRODUCTION

Multi-robot missions have diverse applications, including search and rescue [1], [2], [3], logistics transportation [4], [5], agricultural automation [6], and industrial applications [7], [8], [9]. Such missions require efficient collaboration while addressing key challenges such as inter-robot collision avoidance, navigation, and optimal formation maintenance [10], [11]. Omnidirectional mobile robots are well suited to complex tasks and crowded environments due to their ability to move in any direction without reorientation. A notable application by Airbus [12], deployed two heavy-duty transport vehicles for A380 aircraft assembly, where heavy loads had to be moved in confined spaces. However, due to the uncertainty of the environment, autonomous path planning and navigation remain challenging [13].

Deep Reinforcement Learning (DRL) has demonstrated exceptional generality, stability, and robustness in various applications [14], [15]. A notable instance includes the Deep Q Network (DQN) [16] trained to play video games from image pixels and MuZero [17] mastered chess, shogi, Go, and Atari without any rule-based guidance or human knowledge. DRL robustness extended to robot control, where a time-efficient human-robot collaboration system enhanced productivity and safety in a packaging context [18].

Value-based approaches, including DQN, double DQN [16], and dueling DQN [19], learn optimal value functions but struggle with continuous control problems, whereas

policy-based methods handle continuous action spaces at the cost of high variance in gradient estimation. Actor-critic methods combine the two. The deep deterministic policy gradient (DDPG) algorithm [20], [21] learns a deterministic state-action mapping, and the twin-delayed DDPG (TD3) policy gradient [22], [23] adds a second Q-function and delayed policy update to reduce overestimation bias and stabilize learning.

Multi-agent reinforcement learning (MARL) extends these methods to multi-robot settings [24], [25], [26]. For example, OpenAI [27] trained agents to play hide-and-seek, in which agents learned strategies to interact with other agents and achieve goals. MADDPG [28] is a multi-agent extension of DDPG [20], built on the concept of centralized training that learns the optimal policy by considering the behaviors of all agents, and decentralized execution that allows each agent to act based on its own observations [29]. The study in [30] combined MADDPG with prioritized experience replay (PER) [31], resulting in improved training speed in a single-agent DRL, to solve the multi-robot flocking control.

Despite its promise in various domains, including smart service robots [32], smart manufacturing [33], and semiconductor manufacturing [34], DRL alone faces difficulties in complex environments [35], [36], such as long-range navigation routes or U-shaped traps that yield sparse rewards. In such cases, agents may fall into local minima, become stuck, or take too long to reach the goal.

In terms of global path planning, graph-based methods such as Dijkstra and A* [37] find optimal paths but require substantial computational resources and time, making them unsuitable for real-time operations in large map environments. Sampling-based methods such as PRMs and RRT [38] rely on randomly sampled candidate points and offer fast computation, but the generated paths may not be optimal. RRT* [39] addressed this issue through iterative optimization, and an improved RRT* enhanced path quality by removing residual and redundant nodes along the path [40].

Inspired by prior studies [11], [20], [22], [28], [30], [31], [39], [40], [41], this paper proposes a hierarchical approach that combines the strengths of improved RRT* and DRL for multi-robot flocking control navigation. The improved RRT* generates global paths and strategic waypoints as high-level guidance, while DRL generates velocity commands for local navigation, enabling each AMR to navigate toward the next waypoint and avoid uncertain obstacles simultaneously, mitigating the sparse reward issue. The novelty, originality, contribution, and advantages of this article are summarized as follows;

1. A hybrid and practically validated multi-robot flocking control framework that integrates improved RRT* waypoint planning with a PER-enhanced multi-agent TD3 algorithm, balancing global path guidance with efficient local control.
2. Improved convergence stability and execution efficiency through path optimization in RRT* and clipped double Q-learning with delayed policy updates in MATD3.
3. Validation in simulations and physical robots, including scenarios with static and dynamic obstacles, demonstrating superior collision avoidance, communication maintenance, and cooperative navigation.

The rest of this paper is organized as follows. Section II presents the multi-robot flocking problem, the proposed hierarchical framework, and the improved RRT* algorithm. Section III introduces MATD3 and PER. Section IV presents the proposed PER-MATD3-based local flocking controller. Section V analyzes the convergence and stability. Section

VI describes the training setup and simulation results, while Section VII presents the experimental results. Finally, Section VIII concludes the paper.

2. PROBLEM STATEMENT AND DESCRIPTION OF THE FLOCKING CONTROL SYSTEM

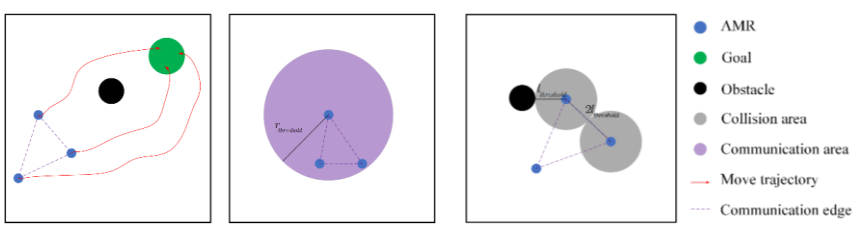
The configuration of the three omnidirectional AMRs is depicted in Fig. 1, where two are equipped with Mecanum wheels and one with omnidirectional wheels. Each AMR carries a high-level computing unit for navigation, a LiDAR, and a wireless communication module, and a low-level motion controller, driven by an OpenCR board, executes the velocity commands from the high-level unit and computes odometry from encoder data using the kinematic models. Once GMapping SLAM [42], [43] is activated, the assigned navigation task is executed via the proposed framework.



Fig. 1. Pictures of the experimental omnidirectional AMRs.

2.1. Related work

Recent studies such as the CTDE-based DRL [31] achieved effective formation control and cohesion [11], [44], [45], but lacked global path planning, limiting its use in long-range. Guo et al. [8] proposed obstacle avoidance in cluttered environments through a distributed control strategy, yet without global waypoint guidance. Liu et al. [7] combined optimization-based flocking with MPC gait synchronization, though such methods may incur high computational costs in dynamic settings. Mao et al. [25] and Liu et al. [24] applied MARL to traffic signal control and heterogeneous UAV task allocation, demonstrating the scalability of attention-based and soft actor-critic methods for multi-agent collaboration. In contrast, our framework leverages the improved RRT* to generate global waypoints and uses PER-MATD3 for local navigation; a hierarchical design intended to reduce reward sparsity in obstacle-rich environments.



(a) (b) (c)

Fig. 2. Description of the multi-AMR flocking control scenario.

2.2. Problem description

The multi-agent flocking navigation task is illustrated in Fig. 2, designed to navigate the AMRs from initial positions to predetermined goals through a challenging environment (Fig. 2a), under flocking constraints. First, the AMRs must maintain communication and move cohesively (Fig. 2b). Second, avoid collisions (Fig. 2c), both between AMRs, which requires maintaining a minimum separation distance, and with environmental obstacles. Each AMR carries a LiDAR for SLAM and obstacle detection, and the Robot Operating System (ROS) handles data exchange [46]. The task is complete when all AMRs reach their goals without collision while maintaining the flocking constraints.

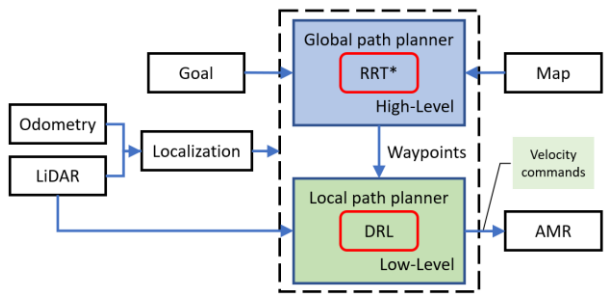


Fig. 3. Hierarchical system architecture of the proposed collision-free flocking control.

2.3. Describing the hierarchical flocking control system

The design procedure is detailed in Fig. 3. First, the improved RRT* algorithm uses the SLAM-generated map to plan a global path from the starting point to the navigation goal, with waypoints placed along the route. Second, each waypoint serves as a local goal. DRL acts as the local planner, generating velocity commands that avoid obstacles and follow the global path when no obstacles are present. The local planner is trained offline to learn a short-range, point-to-point navigation policy from odometry, LiDAR, and the relative distance to the local goal, without map information. The trained policy is then executed online to steer the AMRs. Third, once a local goal is reached, the next waypoint becomes the new local goal. Finally, steps two and three repeat until the navigation goal is reached. Overall, the adopted control strategy is thus hierarchical; the improved RRT* provides waypoints as sequential local goals, the trained decentralized actor maps local observations into velocity commands, and the low-level controller converts these commands into wheel motions via the inverse kinematics in (1)-(3).

2.4. Kinematic modeling

Extending the configuration in Fig. 1, Figs. 4-5 depict the geometry and posture definitions of omnidirectional AMRs. The forward and inverse kinematic models of both AMRs with the orientation θ are described as follows [10], [47];

$$\boldsymbol{V_0} = \boldsymbol{J_0}^{+}(\theta)\boldsymbol{V_W}, \text{ and } \boldsymbol{V_W} = \boldsymbol{J_0}(\theta)\boldsymbol{V_0} \tag{1}$$

which $\boldsymbol{V_W} \in R^{4\times1}$ denotes the wheel velocity vector and $\boldsymbol{V_0} = [v_x \quad v_y \quad \omega_z]^T \in R^{3\times1}$ represents the velocity vector in the world frame. For the Mecanum-wheeled AMR, let $\theta_1 = \theta + (\pi/4)$ and then obtain $\boldsymbol{J}^{+}(\theta)\boldsymbol{J}(\theta) = \boldsymbol{I}_3$ where

$$\boldsymbol{J}(\theta)=\begin{bmatrix}\sqrt{2}\sin(\theta_1) & -\sqrt{2}\cos(\theta_1) & -(l+L)\\ \sqrt{2}\cos(\theta_1) & \sqrt{2}\sin(\theta_1) & (l+L)\\ \sqrt{2}\cos(\theta_1) & \sqrt{2}\sin(\theta_1) & -(l+L)\\ \sqrt{2}\sin(\theta_1) & -\sqrt{2}\cos(\theta_1) & (l+L)\end{bmatrix};\ \boldsymbol{J}^{+}(\theta)=\frac{1}{4}\begin{bmatrix}\sqrt{2}\sin(\theta_1) & \sqrt{2}\cos(\theta_1) & \sqrt{2}\cos(\theta_1) & \sqrt{2}\sin(\theta_1)\\ -\sqrt{2}\cos(\theta_1) & \sqrt{2}\sin(\theta_1) & \sqrt{2}\sin(\theta_1) & -\sqrt{2}\cos(\theta_1)\\ -\frac{1}{L+l} & \frac{1}{L+l} & -\frac{1}{L+l} & \frac{1}{L+l}\end{bmatrix} \quad (2)$$

where L and l denote the half-length and half-width distances from the robot center to the wheel axes. For the omnidirectional AMR, we also have $\boldsymbol{J}^{+}(\theta)\boldsymbol{J}(\theta)=\boldsymbol{I}_3$ where

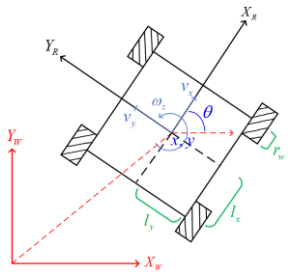


Fig. 4. Simplified geometry of the Mecanum-wheeled AMR.

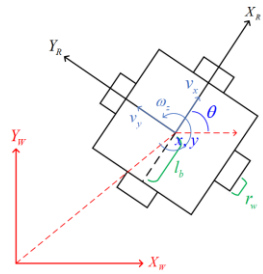


Fig. 5. Simplified geometry of the omni-wheeled AMR.

Algorithm 1: RRT*

$V \leftarrow \{x_{start}\}; E \leftarrow \varnothing; T \leftarrow (V, E)$
for $i = 1, \cdots, N$ **do**
  $x_{rand} \leftarrow$ RandomSample()
  $x_{near} \leftarrow$ getNearestNode(x_{rand}, T)
  $x_{new} \leftarrow$ steer(x_{near}, x_{rand})
  if noCollision(x_{near}, x_{new}) **then**
    $X_{neibors}$ = findNeighbors($x_{new}, T, radius$)
    $x_{best} = x_{near}$
    $c_{\min}$ = cost(x_{near}) + dist(x_{near}, x_{new})
    for x_{neibor} **in** $X_{neibors}$
      if noCollision(x_{neibor}, x_{new}) **then**
        c = cost(x_{neibor})+dist(x_{neibor}, x_{new})
        if $c < c_{\min}$ **then**
          $x_{best} = x_{neibor}$, $c_{\min} = c$
    $V \leftarrow \{x_{new}\}; E \leftarrow \{e_{x_{best}, x_{new}, c_{\min}}\}$
  for x_{neibor} **in** $X_{neibors}$
    if noCollision(x_{new}, x_{neibor}) **then**
      c_{new} = cost(x_{new}) + dist(x_{new}, x_{neibor})
      if $c_{new} <$ cost(x_{neibor}) **then**
        rewire ($x_{neibor}, x_{new}, c_{new}, T$)
retune T

Algorithm 2: Path Optimization

Input : $T(V, E)$
Output : $T'(V', E')$
N = *Length*(T)
$V' \leftarrow \{x_0\}$
while $i < N$-1 **do**
  for $j = N$-1,…, i **do**
    if noCollosion(x_i, x_j) **then**
      $V' \leftarrow \{x_j\}; E' \leftarrow \{e_{x_i, x_j}\}$
      $i = j$
      break
retune T'

$$\boldsymbol{J}(\boldsymbol{\theta})=\begin{bmatrix}-\sin(\theta) & \cos(\theta) & l_b\\ \cos(\theta) & \sin(\theta) & -l_b\\ -\sin(\theta) & \cos(\theta) & -l_b\\ \cos(\theta) & \sin(\theta) & l_b\end{bmatrix};\ \boldsymbol{J}^{+}(\boldsymbol{\theta})=\begin{bmatrix}-\frac{\sin(\theta)}{2} & \frac{\cos(\theta)}{2} & -\frac{\sin(\theta)}{2} & \frac{\cos(\theta)}{2}\\ \frac{\cos(\theta)}{2} & \frac{\sin(\theta)}{2} & \frac{\cos(\theta)}{2} & \frac{\sin(\theta)}{2}\\ \frac{1}{4l_b} & -\frac{1}{4l_b} & -\frac{1}{4l_b} & \frac{1}{4l_b}\end{bmatrix} \tag{3}$$

where l_b is the distance from the robot's center to each omnidirectional wheel.

2.5. Improved rapidly-exploring random tree* (RRT*)

The basic RRT algorithm (Algorithm 1) finds a feasible path without considering the path quality, such as length or smoothness. RRT* improves it by re-evaluating nearby connections after inserting each new node to minimize the total cost-to-come, promoting convergence toward an optimal solution as time increases [48]. In this work, we extend RRT* with a post-processing path optimization step that removes redundant and residual nodes that do not contribute to path deviation. This simplification results in a shorter, more efficient trajectory while maintaining feasibility, with the step summarized in Algorithm 2. This improved RRT*, inspired by [40], [49], produces more strategic waypoints for global navigation and improves integration with the learning-based local controller.

3. MULTI-AGENT DEEP REINFORCEMENT LEARNING

3.1. Multi-agent deep reinforcement learning

The multi-agent task is modeled as a partially observable Markov decision process (POMDP) with N agents [10], [11]. At time step t, each agent $i \in \{1,\dots,N\}$ receives a partial observation $o_{i,t} \in \mathcal{O}_i$ of the underlying state $s_t \in \mathcal{S}$, forming the joint observation $\boldsymbol{o_t} = (o_{1,t},\dots,o_{N,t})$. Each agent executes an action $a_{i,t} \in \mathcal{A}_i$ under its policy $\pi_i: \mathcal{O}_i \to \mathcal{A}_i$, which results in a joint action $\boldsymbol{a_t} = (a_{1,t},\dots,a_{N,t})$ and joint policy $\boldsymbol{\pi} = (\pi_1,\dots,\pi_N)$. It then receives an individual reward $r_{i.t}: \mathcal{S} \times \mathcal{A}_i \to \mathbb{R}$ and a new observation $o_{i.t+1}$, with the joint reward $\boldsymbol{r}_t = (r_{1,t},\dots,r_{N,t})$. Each agent maximizes its accumulated discounted return $R_{i,t} = \sum_{j=t}^{T} \gamma^{j-t} r_{i,j}$, where $\gamma \in [0,1]$ is the discount factor.

3.2. Multi-agent deep deterministic policy gradient (MADDPG)

In MADDPG, each transition step $< \boldsymbol{o_t}, \boldsymbol{a_t}, \boldsymbol{r_t}, \boldsymbol{o_{t+1}} >$ is stored in the replay buffer B, from which mini-batches of size m are uniformly sampled. The actor ϕ_i of agent i is updated by the deterministic policy gradient

$$\nabla_{\phi_i} J(\phi_i) = \frac{1}{m}\sum_{j}^{m} \mathbb{E}_{\boldsymbol{o_j},\boldsymbol{a_j}\sim B}\left[\nabla_{a_i} Q_i^{\boldsymbol{\pi}}(\boldsymbol{o_j},\boldsymbol{a_j};\theta_i)\Big|_{a_i=\pi_i(o_{i,j};\phi_i)} \nabla_{\phi_i}\pi_i(o_{i,j};\phi_i)\right] \tag{4}$$

where $Q_i^{\boldsymbol{\pi}}(\boldsymbol{o_i},\boldsymbol{a_i};\theta_i)$ denotes the Q-value under policy $\boldsymbol{\pi}$. The critic θ_i is updated by temporal difference (TD) learning with a frozen target network, giving the loss

$$L(\theta_i) = \frac{1}{m}\sum_{j} \mathbb{E}_{\boldsymbol{o_j},\boldsymbol{r_j}\sim B,\boldsymbol{a_j}\sim\boldsymbol{\pi}}\left[\left(y_{i,j} - Q(\boldsymbol{o_j},\boldsymbol{a_j};\theta_i)\right)^2\right] \tag{5}$$

$$y_{i,j} = r_{i,j} + \gamma Q\left(\boldsymbol{o_{j+1}},\boldsymbol{a_{j+1}};\theta_i'\right) \tag{6}$$

where $\boldsymbol{a_{j+1}} = (\pi_1(o_{1,j+1};\phi_1'),\dots,\pi_N(o_{N,j+1};\phi_N'))$ is produced by the target actors, and θ_i' and ϕ_i' are the target critic and target actor parameters, updated by exponential smoothing with the update rate $\tau \in (0,1)$

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'; \qquad \phi' \leftarrow \tau\phi + (1-\tau)\phi' \tag{7}$$

Therefore, MADDPG trains the actor to maximize this Q-value while the critic learns to estimate it accurately.

3.3. Multi-agent twin delayed deep deterministic (MATD3) policy gradient

MATD3 policy gradient introduces three mechanisms to address DDPG's challenges, with details illustrated in Fig. 6. First, clipped double Q-learning utilizes two critic networks, replacing the objective in (6) turns out to be;

$$y_{i,j} = r_{i,j} + \gamma \min\left(q^1, q^2\right) \tag{8}$$

$$q^1 = Q\left(\boldsymbol{o}_{j+1}, \boldsymbol{a}_{j+1}; \theta_i^{1'}\right), \quad q^2 = Q\left(\boldsymbol{o}_{j+1}, \boldsymbol{a}_{j+1}; \theta_i^{2'}\right) \tag{9}$$

The target value is taken as the minimum between these two estimates. Second, the actor is updated only after the critic has been updated for d steps, reducing the variance of the value estimate before each policy update. Third, target policy smoothing adds clipped Gaussian noise to the target action as a regularization that smooths the Q-value:

$$y_{i,j} = r_{i,j} + \gamma Q\left(\boldsymbol{o}_{j+1}, \boldsymbol{a}_{j+1} + \boldsymbol{\varepsilon}; \theta_i'\right) \tag{10}$$

where $\boldsymbol{\varepsilon} \sim \text{clip}\left(\mathcal{N}(0,\sigma), -c, c\right)$.

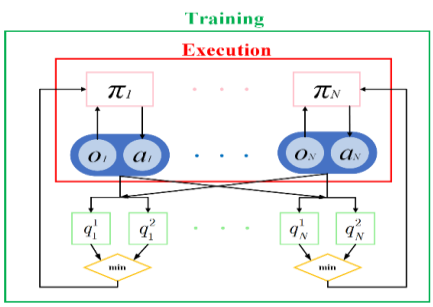


Fig. 6. Decentralized execution and centralized training approach.

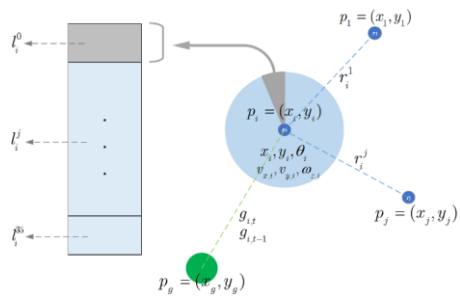


Fig. 7. Graphical illustration of the observation space.

3.4. Prioritized experience replay (PER)

Uniform sampling treats all transitions as equally informative. PER instead ranks them by TD error so that transitions with larger errors are replayed more often. The probability of agent i drawing sample j is

$$P_{i,j} = \frac{(p_{i,j})^{\alpha}}{\sum_{M}(p_{i,m})^{\alpha}} \tag{11}$$

where M is the number of samples in the replay buffer, and $\alpha \in [0,1]$ adjusts the balance between uniform and greedy sampling. The priority of the sample j is expressed as;

$$p_{i,j} = \max\left(\left|\delta_{i,j}^1\right|, \left|\delta_{i,j}^2\right|\right) + \varepsilon \tag{12}$$

where $\varepsilon > 0$ is a small constant preventing transitions from never being revisited once their error reaches zero and

$$\left|\delta_{i,j}^{1}\right| = \left|y_{i,j} - Q\left(\boldsymbol{o}_j, \boldsymbol{a}_j; \theta_i^1\right)\right|; \quad \left|\delta_{i,j}^{2}\right| = \left|y_{i,j} - Q\left(\boldsymbol{o}_j, \boldsymbol{a}_j; \theta_i^2\right)\right| \tag{13}$$

represent absolute TD errors for the two Q-value approximations. Since prioritization biases the sampling distribution, the applied importance-sampling weights become

$$w_{i,j} = \left(\frac{1}{M} \cdot \frac{1}{P_{i,j}}\right)^{\beta} \tag{14}$$

where β incrementally approaches unity to counterbalance the PER effect, giving all experiences a chance to be selected. The PER hyperparameters follow the standard settings in [31] ($\alpha = 0.6$, β annealed from 0.4 to 1.0, $\varepsilon = 10^{-6}$). The weights are normalized by

$$w_{i,j} = \frac{(m \cdot P_{i,j})^{-\beta}}{(m \cdot P_{i,k})^{-\beta}} = \frac{(P_{i,j})^{-\beta}}{(P_{i,k})^{-\beta}} \tag{15}$$

where k is the index with the largest sample weight. These weights are integrated into the Q-learning update by using $w_{i,j}\delta_{i,j}$, so the loss function becomes:

$$L\left(\theta_i\right) = \frac{1}{m}\sum_{j} \mathbb{E}_{\boldsymbol{o}_j, \boldsymbol{r}_j \sim B, \boldsymbol{a}_j \sim \boldsymbol{\pi}}\left[w_{i,j} \cdot \left(y_{i,j} - Q\left(\boldsymbol{o}_j, \boldsymbol{a}_j; \theta_i\right)\right)^2\right] \tag{16}$$

4. FLOCKING CONTROLLER DESIGN USING DEEP REINFORCEMENT LEARNING

4.1. Observation and action space

For agent i, it receives the partially observable observations $o_i \in \mathbb{R}^{52\times 1}$ from the environment, as illustrated in Fig. 7. It includes LiDAR data, AMR pose, information about the goal, locations of other AMRs, and the relative distances between these AMRs. In practice, the LiDAR provides 360 readings at one-degree resolution. To reduce redundancy and computational cost, these readings are divided into 36 sectors, with the minimum value from each sector normalized to [0,1], yielding $l_i^j, j = 0, \ldots, 35$. Additionally, the AMR pose includes its position $p_i = (x_i, y_i)$, orientation θ_i, linear velocities $v_{x,i}, v_{y,i}$ and angular velocity $\omega_{z,i}$. The goal information consists of the goal's position $p_g = (x_g, y_g)$, the current distances $g_{i,t}$ and previous distances $g_{i,t-1}$ to the goal. Lastly, the observations include the locations of other AMRs $p_j, j \neq i$, and the relative distance $r_i^j = \|p_i - p_j\|_2$. In the action space, each agent's actor generates its own velocity commands $a_{i,t} \in \mathbb{R}^{3\times 1}$, which includes linear velocities $v_{x,i}, v_{y,i}$ ranging from -0.25 m/s to 0.25 m/s and angular velocity $\omega_{z,i}$ ranging from -1 rad/s to 1 rad/s.

4.2. Chosen reward functions

Each agent has the same reward function. For agent i, the reward is calculated by

$$r_i = \alpha_{norm}\left(r_{goal} + r_{lidar} + r_{comm}\right) \tag{17}$$

where r_i is the sum of rewards, α_{norm} acts as the reward scaling factor to rescale r_i to an appropriate number. This indirectly adjusts the Q-values to a suitable range, making it easier for the critic network to fit the function. The coefficients α_{norm}, β_g, and β_l were fixed a priori so that the per-step reward remains on the order of $[-1, 1]$, and were not tuned. Within (17), r_{goal} is the goal-oriented reward, motivating the agent to reach the goal point. Conversely, r_{lidar} serves as a penalty for any collision, discouraging such behavior.

Finally, r_{comm} denotes the communication penalty, which penalizes any AMR that falls out of the communication range. Thus, r_{goal} is calculated by

$$r_{goal} = \begin{cases} r_{arrival} & \text{if } g_{i,t} < g_{threshold} \\ r_{forward} & \text{otherwise} \end{cases} \tag{18}$$

where $r_{arrival}$ is the arrival reward when the distance between the goal and the AMR $g_{i,t}$ is less than the threshold $g_{threshold}$. Otherwise, set $r_{forward} = \beta_g \Delta d_g = \beta_g (g_{i,t} - g_{i,t-1})$ in order to encourage the agent to move toward the goal, and β_g is a tuning parameter. Moreover, r_{lidar} is defined as follows

$$r_{lidar} = \begin{cases} r_{collision} & \text{if } \min(l_i^j) < l_{threshold} \text{ or } \min(r_i^j) < 2l_{threshold} \\ r_{avoid} & \text{otherwise} \end{cases} \tag{19}$$

In this definition, r_{col} represents the negative reward when the minimum LiDAR distance $min(l_i^j)$, falls below the threshold distance $l_{threshold}$, indicating the AMR collides with any uncertain obstacles. Similarly, a collision between AMRs is implied if the minimum inter-AMR distance, $min(r_i^j)$, drops below twice the $l_{threshold}$. In contrast, when there is no collision, the agent receives a reward r_{avoid}, which is computed as

$$r_{avoid} = \text{clip}\left(\frac{-1}{\beta_l \min(l_i^j) + \varepsilon} + \frac{1}{\beta_l}, -1, 0 \right) \tag{20}$$

where β_l is a tuning variable and ε is a small constant to avert division by zero.

Furthermore, the penalty is clipped between -1 and 0 to restrict it to a suitable range. In addition, r_{comm} is a penalty term that discourages AMRs from being out of communication range. It is computed as follows;

$$r_{comm} = \begin{cases} r_{offline} & \text{if } \max(r_i^j) > r_{threshold} \\ r_{keep} & \text{otherwise} \end{cases} \tag{21}$$

Here, $r_{offline}$ means a penalty that is incurred when the maximum distance between AMRs $max(r_i^j)$, exceeds the communication range threshold $r_{threshold}$. In situations where the AMR remains within the communication range, it is awarded a reward referred to as r_{keep}, encouraging the maintenance of connectivity between AMRs. The thresholds are set by the physical configuration: $l_{threshold}$ is the robot's radius plus safety margin, $g_{threshold}$ matches the SLAM tolerance, and $r_{threshold}$ below the wireless reliable range.

4.3. Neural network architectures

In its architecture, each agent employs identical settings. The number of layers and neurons follow TD3 [22] and MADDPG [28], and were selected to balance representational capacity with the real-time inference of the onboard computers. Following Fig. 8, the actor network accepts partial observations as its inputs, comprising three fully connected hidden layers, each featuring 256 neurons and ReLU activation functions. This configuration provides sufficient nonlinearity to map the 52-dimensional observation to velocity commands while remaining compact enough for online execution. The outputs (velocity commands) are passed through a hyperbolic tangent function. In addition, layer normalization is applied to each layer to accelerate training. Subsequently, the velocity

commands and partial observations are concatenated as inputs to the critic network. As shown in Fig. 9, the critic network structure includes three fully connected hidden layers. The first contains 512 neurons to handle the higher input load and the remaining hidden layers maintain the standard 256 neurons.

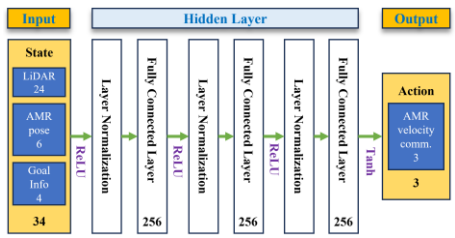


Fig. 8. Actor network architecture.

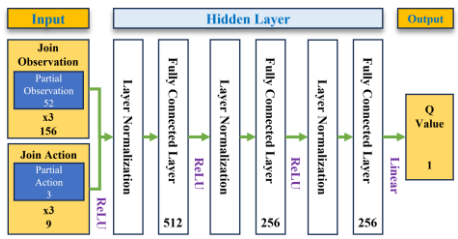


Fig. 9. Critic network architecture.

5. CONVERGENCE AND STABILITY ANALYSIS

To ensure robustness, we present two theorems that provide theoretical guarantees for the system's behavior and convergence properties adapted from the MADRL framework in Section III and the flocking control theory in Section IV.

Theorem 1 (Convergence Analysis): Consider a multi-agent system where each agent $i \in 1, \ldots, N$ updates its policy π_i and the critic network Q_i via the PER-MATD3 algorithm, and suppose that the following conditions hold:

1. The reward r_i defined in (17) is bounded.
2. The observation $\mathcal{O}_i$ and action space $\mathcal{A}_i$ are compact subsets of Euclidean spaces.
3. ADAM optimizer parameters are set within appropriate ranges for convergence.
4. The replay buffer B is sufficiently large to represent the transition distribution.
5. The sampling distribution is corrected using importance sampling weights (15).

Then, the critic network $Q_i(\boldsymbol{o_j}, \boldsymbol{a_j}; \theta_i)$ converges almost surely to a neighborhood of the optimal Q-function Q_i^*, and the actor network $\pi_i(o_{i,j}; \phi_i)$ converges to a locally optimal deterministic policy.

Proof: The critic network for each agent minimizes the TD error loss (16) with the target value computed via clipped double Q-learning (8). The weights (15) correct the bias initiated by the probability (11) and priority (12), then the parameter update becomes:

$$\theta_i \leftarrow \theta_i - \eta \nabla_{\theta_i} L(\theta_i)$$

which constitutes a stochastic gradient descent with corrected variance and satisfies the Robbins-Monro conditions under appropriate learning rates.

Thus, Q_i converges almost surely to a small neighborhood of the true Q_i^*, where the neighborhood size depends on the replay buffer approximation error and the capacity of the neural network. However, since Q_i converges and $\nabla_{a_i} Q_i$ is Lipschitz in a_i, the actor is updated via the deterministic policy gradient (4) and converges toward a local maximum of the expected return. The delayed policy updates and target action noise (10) further stabilize this convergence. Therefore, the convergence is established and confirmed [50].

Theorem 2 (Stability Analysis): Let N AMRs be controlled by the decentralized PER-MATD3 policies using the reward function (17), and the following conditions hold:

1. Sensors have bounded noise.
2. Communication ranges are sufficient for the operational environment.

3. Feasible paths to goals exist without deadlock.

Then, the learned policies π_i achieve almost surely:

1. No collisions between AMRs or obstacles.
2. Maintenance of communication links among AMRs.
3. Convergent flocking motion toward goals.

Proof: The reward function decomposes into three components that directly correspond to the three flocking objectives:

1. **Goal-oriented Reward:** From (18), if agent *i*'s distance to goal $g_{i,t}$ satisfies

$$g_{i,t} < g_{threshold} \Rightarrow r_{goal} = r_{arrival} > 0 \tag{22}$$

Otherwise

$$r_{goal} = r_{forward} = \beta_g \left(g_{i,t-1} - g_{i,t} \right) \tag{23}$$

then this positive shaping reward can move the agent closer to the goal.

2. **Collision Avoidance:** From (19), when the minimum LiDAR reading $\min\left(l_i^j\right)$ falls below the safety $l_{threshold}$, or the minimum inter-AMR distance $\min\left(r_i^j\right)$ falls below $2l_{threshold}$, a negative reward is

$$r = \begin{cases} r_{collision} & \text{If a collision is detected} \\ r_{avoid} & \text{otherwise} \end{cases}$$

where r_{avoid} (20) is a function that increasingly penalizes closer proximity to obstacles.

3. **Communication Maintenance:** From (21), if the maximum distance between agents exceeds the communication limit $r_{threshold}$, policies are trained to favor the connected movement.

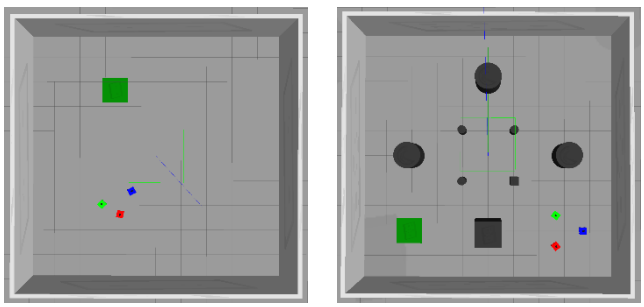
Since the reward function penalizes collisions and disconnections while incentivizing progress toward goals, the learned behavior optimizes $E\left[\sum_{t=0}^{\infty} \gamma^t r_{i,t}\right]$ toward minimizing collision risks and communication loss, while promoting goal reaching. By the convergence of Theorem 1, and given that the reward function directly aligns with the three flocking objectives, the resulting behavior maintains stable flocking satisfying all three objectives almost surely. Furthermore, PER prioritizes critical experiences involving potential collisions or communication failures, accelerating convergence to policies that effectively avoid these failure modes.

6. COMPARATIVE SIMULATIONS AND DISCUSSION

6.1. Training Results and Discussion

During training, the Gazebo physics simulation engine was utilized. The first training environment has a size of 10m × 10m with no obstacles as shown in Fig. 10(a), and many static obstacles in Fig. 10(b). The goal is marked by a dark-green box, while the three AMRs are represented in red, green, and blue for clarity. Both the initial poses of the AMRs and the goal point are randomly generated in this environment. The initial poses of the AMRs and the goal are randomly generated in each episode. The reward curves in Fig. 11 show a representative training run, consistent with related works. All hyperparameters are listed in Table 1, with identical settings for all agents. We selected deterministic policy algorithms (DDPG, TD3, and MATD3) for their advantages in continuous control and consistently low-variance actions, which are suitable for real-world multi-robot systems. Although stochastic methods such as MA-SAC [51] can improve exploration, they

typically require additional tuning and may introduce coordination instability. We first compare MADDPG, PER-MADDPG, MATD3, and PER-MATD3 in an obstacle-free environment. For clearer comparison, the rewards of all agents are summed per step and exponentially smoothed.



(a) (b)

Fig. 10. Gazebo simulation environment settings. (a) Obstacle-free. (b) Static obstacles.

Table 1. Hyperparameter settings

Hyperparameter	
Critic Learning Rate	0.0003
Actor Learning Rate	0.0003
Optimizer	Adam
Target Update Rate	0.005
Policy Update Delay	2
Batch Size	256
Discount Factor	0.99
Exploration Policy	$\mathcal{N}(0,0.1)$
Policy Smoothing	$\mathcal{N}(0,0.2)$

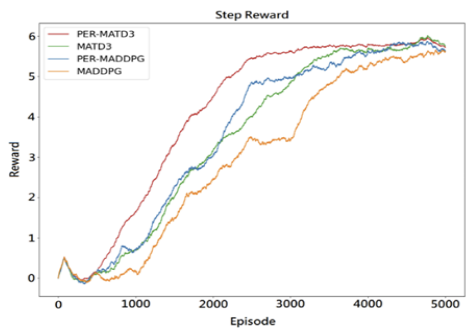


Fig. 11. Learning curves comparison in the obstacle-free environment.

Fig. 11 can be explained as follows. First, MADDPG exhibits greater fluctuations in its reward curve, while PER-MADDPG performs better than MATD3 and PER-MATD3, indicating that the proposed mechanisms help stabilize training. Second, PER-MADDPG and PER-MATD3 outperform their non-PER counterparts, demonstrating that PER improves learning efficiency. Regarding convergence speed, PER-MATD3 outperforms MADDPG and PER-MADDPG, indicating that combining MATD3 with PER enhances cooperative control performance in obstacle-free environments. This also demonstrates that TD3's faster convergence advantage extends to multi-agent cooperative control. Finally, although MADDPG and PER-MADDPG achieve stable learning after sufficient training, their higher reward variance results in less consistent performance than MATD3 and PER-MATD3.

Next, we compare MATD3 and PER-MATD3 in two different scenarios. The first scenario involves static obstacles (Fig. 12(b)). The second scenario includes mixed obstacles composed of both static and dynamic obstacles (Fig. 13). As shown in Fig. 14, the learning curves indicate that the PER mechanism not only accelerates convergence speed but also marginally contributes to the stability of the training process.

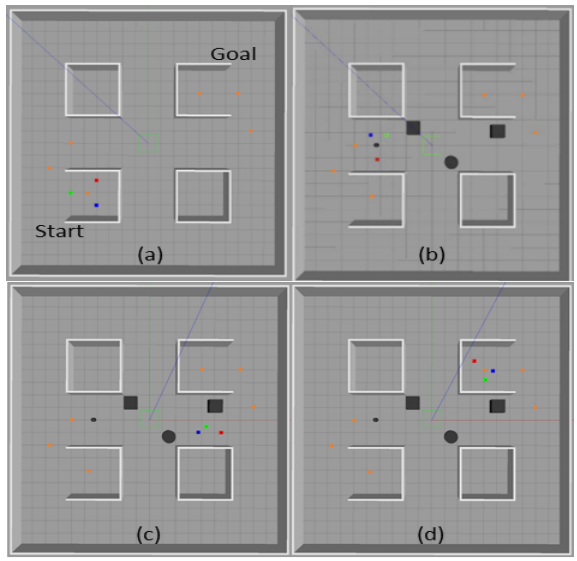


Fig. 12. Simulation results of integrating the improved RRT* and PER-MATD3. (a) - (d) Descriptions of the process of flocking control.

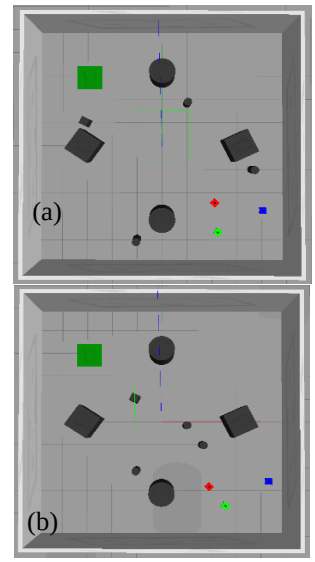
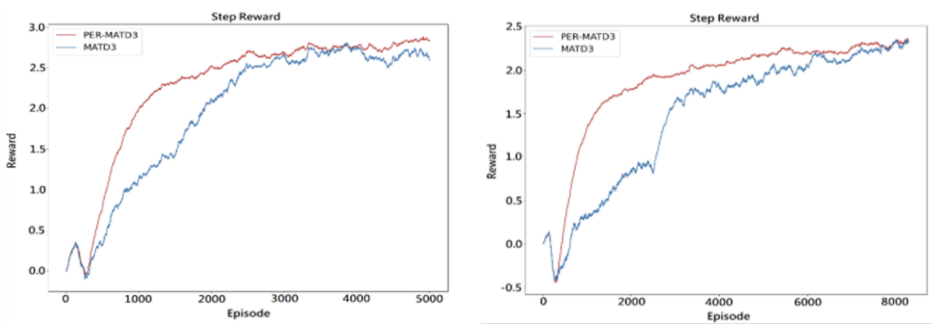


Fig. 13. Moving obstacles in a mixed environment: (a) Initial positions, (b) During simulation.



(a) (b)

Fig. 14. Learning curve of MATD3 and PER-MATD3.
(a) Static environment. (b) Mixed obstacles environment.

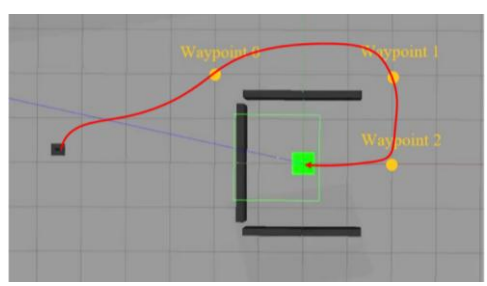


Fig. 15. Illustration scenario of integrating improved RRT* and TD3 algorithms in a U-shaped trap environment.

Table 2. Success rate comparison of TD3 with and without improved RRT* in a U-shaped trap environment

Algorithm	TD3	Improved RRT* + TD3
Success	9	100
Timeouts	91	0
Total	100	100
Success Rate	9%	100%

6.2. Simulation of Improved RRT* and PER-MATD3 Integration

This subsection presents a detailed Gazebo simulation involving two U-shaped structures that challenge purely DRL methods due to sparse rewards. The task involves navigating three AMRs from one U-shaped structure to another. Initially, the Gmapping SLAM is used, and then waypoints are generated using the improved RRT*. As shown by the orange points in Fig. 15, the PER-MATD3 successfully follows the waypoints while avoiding obstacles and maintaining cohesive flocking formation throughout the sequence.

To further demonstrate the impact of integrating the improved RRT*, we conducted a scenario-based comparison between TD3 alone and improved RRT* + TD3 in a challenging U-shaped trap environment. As shown in Fig. 15, improved RRT* was used to generate three intermediate waypoints between the initial pose and the goal, which could not be effectively reached using TD3 alone due to sparse rewards and local minima. Using these waypoints, the TD3 controller successfully navigated the robot through the trap.

The results, summarized in Table 2, indicate that TD3 without RRT* achieved only a 9% success rate, with 91 out of 100 trials ending in timeout. In contrast, the integrated approach achieved a 100% success rate. This ablation-style study confirms that the improved RRT* significantly enhances navigation reliability in complex environments and plays a critical role in the effectiveness of our proposed framework.

7. EXPERIMENTAL RESULTS AND DISCUSSION

The experimental study devoted into two stages. In the first experiment, AMRs aimed to reach their destinations through static obstacles while in the second experiment through static and mixed obstacles. The illustration in Fig. 16 details the connections and communication methodologies between each component and demonstrates the application of ROS for data transmission. Moreover, the low-level motion controller using OpenCR subscribes to this topic and uses inverse kinematics to decompose it into the speed commands of the four motors. Afterward, the speed commands are sent to drive the motors, thus completing the robot's motion control [10], [44].

7.1. Experiment Setup

The experimental site was set between the corridor and the stairs outside the laboratory. A destination location behind the iron door (Fig. 17 (a)) was selected as the navigation target. As shown in Fig. 17 (b), a pre-constructed map derived through SLAM was utilized to establish waypoints for TD3 using the improved RRT* approach. The AMR was equipped with a single-board computer, power system, electric power system, and sensing system.

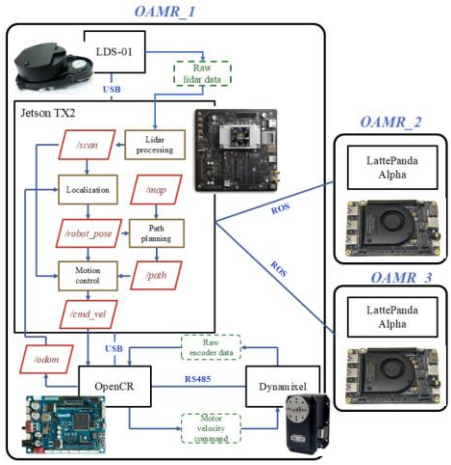


Fig. 16. Multi-robot flocking control structure using three OAMRs: OAMR-1 employs an Nvidia Jetson TX2 as the main controller, while OAMR-2 and OAMR-3 use single-board computers.



(a)



(b)

Fig. 17. Experiment Setup. (a) Real-world experimental site. (b) SLAM map with waypoints.

7.2. Real-World Experiment with Static Obstacles

The first experimental processes are shown in Figs. 18(a)-(h), where the boxes are represented as unexpected static obstacles. Figs. 18(b)-(d) show how the flocking AMRs split to avoid the uncertain obstacles, while Fig. 18(c) describes how they merged back to maintain flocking. Figs. 18(f)-(i) demonstrate how the three AMRs passed through a narrow door one by one. Finally, Fig. 18(j) confirms the successful control of flocking from the start to the target location. As shown in Fig. 19, the continuous trajectories of the three collective AMRs were respectively displayed in red, green, and blue lines.

7.3. Real-World Experiment with Mixed Obstacles

The second experiment used the same configuration as the first experiment (Section 7.2) with different static obstacles placed in various positions and one dynamic obstacle. Fig. 20 shows the experimental sequence, where two cardboard boxes were used as static obstacles (Fig. 20(a)), while random moving person served as a dynamic obstacle (Figs. 20(b)-(e)). Figs. 20(b)-(c) exhibit that the third AMR initially took the right route to avoid the first static obstacle but encountered a dynamic obstacle along the way. In response, the third AMR moved to the left side while avoiding both dynamic and static obstacles (Figs. 20(d)-(f)). Finally, Fig. 20(j) confirms that the flocking AMRs reached the destination. Similar to the first experiment, Fig. 21 depicts the continuous moving trajectories of the three AMRs in red, green, and blue lines, respectively. As demonstrated by a series of still figures, the AMRs successfully avoided static and dynamic obstacles while navigating toward their destinations, demonstrating the performance and versatility of the proposed method in complex, dynamic flocking environments.

DOI: 10.6688/JISE.20220X_38(X).00XX

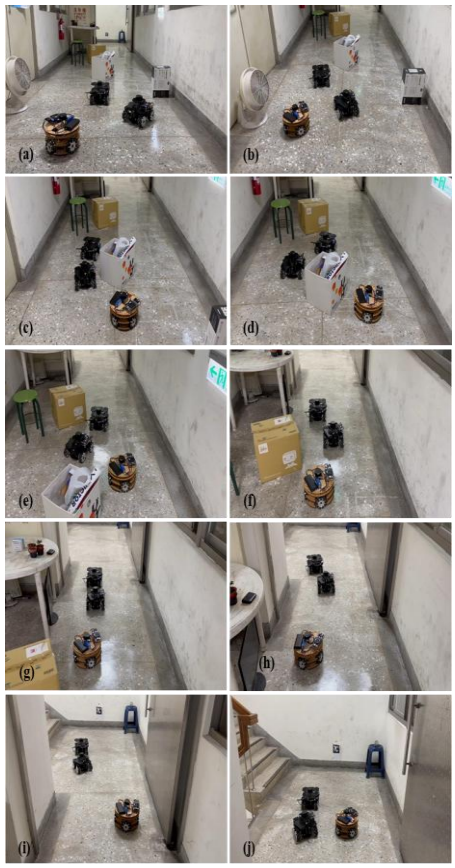


Fig. 18. Real-world experiment with a scenario of static obstacles.
(a) - (j) Still pictures during the autonomous navigation.

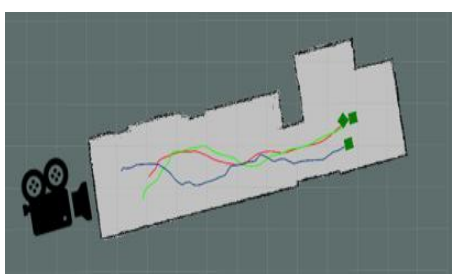


Fig. 19. Experimental trajectories of three AMRs with static obstacles.



Fig. 20. Real-world experiment with a scenario of mixed obstacles.
(a) - (h) Still pictures during the autonomous navigation.

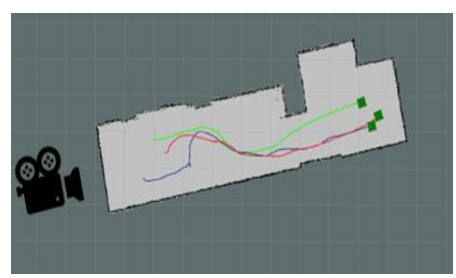


Fig. 21. Experimental trajectories of three AMRs with mixed obstacles

Furthermore, although the experiments were conducted in controlled environments with three AMRs and limited obstacle types, they provide a practical validation of the proposed framework's applicability in real deployment settings. The experimental site included constrained and semi-open spaces with static and dynamic obstacles, representing common indoor navigation challenges. Regarding scalability, the proposed framework is decentralized during execution, in which each AMR runs only its lightweight actor network, inter-robot communication is limited to pose exchange, and the sectored LiDAR representation maintains a fixed-dimensional input, so per-robot computation does not increase with swarm size. The main bottleneck is centralized training, where the critic input grows linearly with the number of agents; moreover, policies require retraining for different team sizes because the observations include other AMRs' states.

For more complex environments, the hierarchical design limits the impact of increasing task difficulty as the improved RRT* decomposes complex global maps into short waypoint-to-waypoint segments, keeping the local navigation task at a similar difficulty regardless of overall environment complexity. The additional computational cost is mainly borne by the high-level planner, whose complexity increases with map size and clutter but remains outside the real-time control loop.

8. CONCLUSIONS

This paper has proposed a practically validated control framework for cooperative flocking of multiple omnidirectional mobile robots. Although no novel learning algorithm is introduced, our approach contributes a unique integration of an improved RRT* algorithm, which has leveraged SLAM-generated maps to produce strategic global waypoints, with a MATD3 algorithm enhanced by PER. To the best of our knowledge, this integration for multi-robot has not been previously reported in the literature. This integration has enabled centralized training with decentralized execution, allowing individual robots to navigate locally while maintaining group coordination. The framework has trained robots using the PER-MATD3 algorithm to follow designated waypoints while simultaneously avoiding obstacles and preserving flocking behavior, thereby resulting in cohesive group movement even in complex environments. Two theoretical foundations, through convergence and stability analysis, have established mathematical guarantees that the PER-MATD3 algorithm converges to locally optimal policies that successfully achieve the three key flocking objectives: collision avoidance, communication maintenance, and cohesive movement toward goals. Comparative simulations have illustrated that our method has superior performance relative to existing approaches. Furthermore, through comparative experimental results, the proposed integrated approach has been shown to be capable of steering the multi-robots to accomplish satisfactory flocking behavior from any outset points to desired destinations. While the simulation and experimental validation were conducted using three omnidirectional AMRs, this setup still provides a meaningful and representative scenario of multi-robot coordination. An interesting topic for further research would be an integration of an extended version of a fuzzy broad learning system [50], [52], [53] for better collision-free trajectory tracking and scaling up the framework to larger robot swarms and applying it in real-world scenarios such as warehouse automation, search and rescue missions, or intelligent transportation systems. Finally, addressing memory-based architectures or uncertainty-aware sensing presents another valuable direction for improving robustness under occlusion and sensor noise.

REFERENCES

[1] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 33, no. 3, pp. 367–385, Jun. 2003, doi: 10.1109/TSMCB.2003.811794.

[2] A. Davids, "Urban search and rescue robots: from tragedy to technology," *IEEE Intell. Syst.*, vol. 17, no. 2, pp. 81–83, Mar. 2002, doi: 10.1109/MIS.2002.999224.

[3] M. Aggravi, G. Sirignano, P. R. Giordano, and C. Pacchierotti, "Decentralized Control of a Heterogeneous Human–Robot Team for Exploration and Patrolling," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3109–3125, Oct. 2022, doi: 10.1109/TASE.2021.3106386.

[4] D. Zhang, L. G. Pee, and L. Cui, "Artificial intelligence in E-commerce fulfillment: A case study of resource orchestration at Alibaba's Smart Warehouse," *Int. J. Inf. Manag.*, vol. 57, p. 102304, Apr. 2021, doi: 10.1016/j.ijinfomgt.2020.102304.

[5] A. Farinelli, N. Boscolo, E. Zanotto, and E. Pagello, "Advanced approaches for multi-robot coordination in logistic scenarios," *Robot. Auton. Syst.*, vol. 90, pp. 34–44, Apr. 2017.

[6] M. Govindaraju, D. Fontanelli, S. S. Kumar, and A. S. Pillai, "Optimized Offline-Coverage Path Planning Algorithm for Multi-Robot for Weeding in Paddy Fields," *IEEE Access*, vol. 11, pp. 109868–109884, 2023, doi: 10.1109/ACCESS.2023.3322230.

[7] K. Liu, L. Dong, X. Tan, W. Zhang, and L. Zhu, "Optimization-based flocking control and MPC-based gait synchronization control for multiple quadruped robots," *IEEE Robot. Autom. Lett.*, vol. 9, no. 2, pp. 1929–1936, Feb. 2024, doi: 10.1109/LRA.2024.3350372.

[8] J. Guo, J. Qi, M. Wang, C. Wu, and G. Yang, "Collision-Free Distributed Control for Multiple Quadrotors in Cluttered Environments with Static and Dynamic Obstacles," *IEEE Robot. Autom. Lett.*, vol. 8, no. 3, pp. 1501–1508, Mar. 2023, doi: 10.1109/LRA.2023.3240376.

[9] H. Zhao, H. Liu, Y.-W. Leung, and X. Chu, "Self-Adaptive Collective Motion of Swarm Robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1533–1545, Oct. 2018.

[10] C. C. Tsai, H. Y. Chen, C. C. Chan, C. C. Hung, and G. M. Chen, "Intelligent actor-critic learning control for collison-free trajectory tracking of mecanum-wheeled mobile robots," *Int. J. Fuzzy Syst.*, vol. 26, no. 4, pp. 1133–1142, 2024, doi: 10.1007/s40815-023-01656-1.

[11] C. C. Tsai, H. C. Huang, H. Y. Chen, C. C. Hung, and S. T. Chen, "Intelligent collision-free formation control of ball-riding robots using output recurrent broad learning in industrial cyber-physical systems," *IEEE Trans. Ind. Cyber-Phys. Syst.*, vol. 2, pp. 459–470, 2024.

[12] "KUKA omniMove moves heavy parts in confined spaces during construction of the A380 at Airbus," KUKA AG. Accessed: Aug. 04, 2024. [Online]. Available: https://www.kuka.com/en-de/industries/solutions-database/2016/07/solution-robotics-airbus

[13] X. An, C. Wu, Y. Lin, M. Lin, T. Yoshinaga, and Y. Ji, "Multi-Robot Systems and Cooperative Object Transport: Communications, Platforms, and Challenges," *IEEE Open J. Comput. Soc.*, vol. 4, pp. 23–36, 2023, doi: 10.1109/OJCS.2023.3238324.

[14] Y. Jia et al., "A deep reinforcement learning approach using asymmetric self-play for robust multirobot flocking," *IEEE Trans. Ind. Inform.*, vol. 21, no. 4, pp. 3266–3275, Apr. 2025.

[15] L. H. Teixeira and A. Huszak, "Reinforcement learning in path lifetime routing algorithm for VANETs," *J. Inf. Sci. Eng.*, vol. 39, no. 1, pp. 129–147, 2023.

[16] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[17] J. Schrittwieser et al., "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, Dec. 2020, doi: 10.1038/s41586-020-03051-4.

[18] A. Ghadirzadeh, X. Chen, W. Yin, Z. Yi, M. Björkman, and D. Kragic, "Human-centered

collaborative robots with deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 566–571, Apr. 2021, doi: 10.1109/LRA.2020.3047730.

[19] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," in *Proc. 33rd Int. Conf. Machine Learning*, PMLR, Jun. 2016, pp. 1995–2003. Accessed: Apr. 10, 2025.

[20] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," Jul. 05, 2019, *arXiv*: arXiv:1509.02971. doi: 10.48550/arXiv.1509.02971.

[21] J. Xiang, Q. Li, X. Dong, and Z. Ren, "Continuous Control with Deep Reinforcement Learning for Mobile Robot Navigation," in *2019 Chinese Autom. Congr.*, Nov. 2019, pp. 1501–1506.

[22] S. Fujimoto, H. Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," in *Proc. 35th Int. Conf. Machine Learning*, PMLR, Jul. 2018, pp. 1587–1596. Accessed: Apr. 10, 2025. [Online]. https://proceedings.mlr.press/v80/fujimoto18a.html

[23] A. Beikmohammadi and S. Magnússon, "Accelerating actor-critic-based algorithms via pseudo-labels derived from prior knowledge," *Inf. Sci.*, vol. 661, p. 120182, Mar. 2024.

[24] D. Liu, L. Dou, R. Zhang, X. Zhang, and Q. Zong, "Multi-Agent Reinforcement Learning-Based Coordinated Dynamic Task Allocation for Heterogenous UAVs," *IEEE Trans. Veh. Technol.*, vol. 72, no. 4, pp. 4372–4383, Apr. 2023, doi: 10.1109/TVT.2022.3228198.

[25] F. Mao, Z. Li, Y. Lin, and L. Li, "Mastering Arterial Traffic Signal Control With Multi-Agent Attention-Based Soft Actor-Critic Model," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 3129–3144, Mar. 2023, doi: 10.1109/TITS.2022.3229477.

[26] S. Domitran and M. B. Babac, "The Use of Deep Reinforcement Learning for Flying a Drone," *J. Inf. Sci. Eng.*, vol. 37, no. 5, pp. 1165–1176, 2021, doi: 10.6688/JISE.202109_37(5).0012.

[27] B. Baker et al., "Emergent Tool Use From Multi-Agent Autocurricula," Feb. 10, 2020, arXiv: *arXiv*:1909.07528. doi: 10.48550/arXiv.1909.07528.

[28] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," Mar. 14, 2020, *arXiv*: arXiv:1706.02275.

[29] C. C. Tsai, H.-S. Wu, L. H. Chen, A. Rospawan, C. H. Li, and S. C. Chen, "Autonomous steering system using Fuzzy TD3 and GRU-Attention for omnidirectional AMRs in dynamic environments," *Int. J. Fuzzy Syst.*, Feb. 2026, doi: 10.1007/s40815-025-02209-4.

[30] P. Zhu, W. Dai, W. Yao, J. Ma, Z. Zeng, and H. Lu, "Multi-Robot Flocking Control Based on Deep Reinforcement Learning," *IEEE Access*, vol. 8, pp. 150397–150406, 2020.

[31] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," Feb. 25, 2016, *arXiv*: arXiv:1511.05952. doi: 10.48550/arXiv.1511.05952.

[32] H. Liu, Y. Deng, D. Guo, B. Fang, F. Sun, and W. Yang, "An Interactive Perception Method for Warehouse Automation in Smart Cities," *IEEE Trans. Ind. Inform.*, vol. 17, no. 2, pp. 830–838, Feb. 2021, doi: 10.1109/TII.2020.2969680.

[33] C. C. Lin, D.-J. Deng, Y. L. Chih, and H. T. Chiu, "Smart manufacturing scheduling with edge computing using multiclass deep Q network," *IEEE Trans. Ind. Inform.*, vol. 15, no. 7, pp. 4276–4284, Jul. 2019, doi: 10.1109/TII.2019.2908210.

[34] Z. Ma and T. Pan, "Adaptive weight tuning of EWMA controller via model-free deep reinforcement learning," *IEEE Trans. Semicond. Manuf.*, vol. 36, no. 1, pp. 91–99, Feb. 2023.

[35] M. Cai, E. Aasi, C. Belta, and C.-I. Vasile, "Overcoming exploration: deep reinforcement learning for continuous control in cluttered environments from temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 8, no. 4, pp. 2158–2165, Apr. 2023.

[36] A. H. Tan, F. P. Bejarano, Y. Zhu, R. Ren, and G. Nejat, "Deep reinforcement learning for decentralized multi-robot exploration with macro actions," *IEEE Robot. Autom. Lett.*, vol. 8, no.

1, pp. 272–279, Jan. 2023, doi: 10.1109/LRA.2022.3224667.
[37] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.
[38] S. LaValle, "Rapidly-exploring random trees: a new tool for path planning," *Annu. Res. Rep.*, 1998, Accessed: Apr. 11, 2023.
[39] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011, doi: 10.1177/0278364911406761.
[40] J. Ding, Y. Zhou, X. Huang, K. Song, S. Lu, and L. Wang, "An improved RRT* algorithm for robot path planning based on path expansion heuristic sampling," *J. Comput. Sci.*, vol. 67, p. 101937, Mar. 2023, doi: 10.1016/j.jocs.2022.101937.
[41] C. C. Tsai, G. H. Yang, S. C. Chen, A. Rospawan, and H. Y. Zheng, "Global cooperative localization method using fuzzy DDEKF and broad learning system for swarmed omnidirectional mobile robots with uncertainties," *Int. J. Fuzzy Syst.*, Feb. 2026.
[42] J. A. Placed et al., "A survey on active simultaneous localization and mapping: state of the art and new frontiers," *Trans Rob*, vol. 39, no. 3, pp. 1686–1705, Jun. 2023.
[43] C. C. Tsai, C. H. Li, A. Rospawan, and Z. H. Wu, "Adaptive fixed-time sliding-mode trajectory following control of nonholonomic AMRs with adaptive sliding-mode observers," *Asian J. Control*, doi: 10.1002/asjc.70088.
[44] C. C. Tsai, H. Y. Chen, S. C. Chen, F. C. Tai, and G. M. Chen, "Adaptive reinforcement learning formation control using ORFBLS for omnidirectional mobile multi-robots," *Int. J. Fuzzy Syst.*, vol. 25, no. 5, pp. 1756–1769, Jul. 2023, doi: 10.1007/s40815-023-01491-4.
[45] C. C. Tsai, A. Rospawan, C. H. Li, B. W. Chen, and H. S. Wu, "Hybrid navigation of autonomous mobile robots with two swerve wheels in complex and confined environments," *IEEE Access*, vol. 14, pp. 105456–105467, 2026.
[46] D. Cai, R. Li, Z. Hu, J. Lu, S. Li, and Y. Zhao, "A comprehensive overview of core modules in visual SLAM framework," *Neurocomputing*, vol. 590, p. 127760, Jul. 2024.
[47] H. Taheri, B. Qiao, and N. Ghaeminezhad, "Kinematic model of a four mecanum wheeled mobile robot," *Int. J. Comput. Appl.*, vol. 113, no. 3, pp. 6–9, Mar. 2015.
[48] J. Yu, C. Chen, A. Arab, J. Yi, X. Pei, and X. Guo, "RDT-RRT: Real-time double-tree rapidly-exploring random tree path planning for autonomous vehicles," *Expert Syst. Appl.*, vol. 240, p. 122510, Apr. 2024, doi: 10.1016/j.eswa.2023.122510.
[49] T. Xu, "Recent advances in Rapidly-exploring random tree: A review," *Heliyon*, vol. 10, no. 11, Jun. 2024, doi: 10.1016/j.heliyon.2024.e32451.
[50] A. Rospawan, C. C. Tsai, and C. C. Hung, "Two-layer intelligent learning control using output recurrent fuzzy neural long short-term memory broad learning system with RMSprop," *IEEE Access*, vol. 13, pp. 34334–34349, Feb. 2025, doi: 10.1109/ACCESS.2025.3543867.
[51] Z. He, L. Dong, C. Song, and C. Sun, "Multiagent soft actor-critic based hybrid motion planner for mobile robots," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10980–10992, Dec. 2023, doi: 10.1109/TNNLS.2022.3172168.
[52] A. Rospawan, C. C. Tsai, and C. C. Hung, "Intelligent MIMO ORFBLS-based setpoint tracking control with its application to temperature control of an industrial extrusion barrel," *Int. J. Fuzzy Syst.*, vol. 27, no. 3, pp. 774–790, Jun. 2024, doi: 10.1007/s40815-024-01804-1.
[53] A. Rospawan, C. C. Tsai, and C. C. Hung, "Output recurrent fuzzy broad learning systems for adaptive MIMO PID control: theory, simulations, and application," *IEEE Access*, vol. 12, pp. 19388–19404, 2024, doi: 10.1109/ACCESS.2024.3359293.