

## Article

# A Belief-Driven Hybrid Reinforcement Learning Framework for Decentralized Multi-Robot Navigation Under Partial Observability

Vineetha Malathi <sup>1</sup>, Pramod Sreedharan <sup>1,\*</sup>, Rthuraj Puthiyaveedu Rajesh <sup>1</sup>, Vyshnavi Anil Kumar <sup>1</sup>,  
Anil Lal Sadasivan <sup>2</sup>, Ganesha Udupa <sup>1</sup> and Liam Pastorelli <sup>3</sup>

<sup>1</sup> Department of Mechanical Engineering, Amrita Vishwa Vidyapeetham, Amritapuri 690525, Kerala, India; vineetha013@gmail.com (V.M.); rthuraj@gmail.com (R.P.R.); vyshnaviak13@gmail.com (V.A.K.); ganesh@am.amrita.edu (G.U.)

<sup>2</sup> Department of Mechanical Engineering, National Institute of Technology Sikkim, Ravangala 737139, Sikkim, India; anillal65@gmail.com

<sup>3</sup> Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milan, Italy; liam.pastorelli@mail.polimi.it

\* Correspondence: pramods@am.amrita.edu

## Abstract

Decentralized multi-robot navigation is difficult when robots must act from local observations without centralized coordination or explicit inter-robot communication. A belief-driven hybrid reinforcement learning framework is evaluated for planar multi-robot navigation under partial observability. Each robot builds a compact local state from its position, waypoint target, sector-based proximity readings, and a decaying occupancy belief that summarizes recent obstacle evidence. A Deep Deterministic Policy Gradient (DDPG) actor produces continuous velocity proposals, and a lightweight geometric safety-blending layer combines this command with goal-seeking and reactive avoidance vectors before execution. The simulation was revised to use e-puck-compatible heading-limited forward motion rather than side-slip motion. The framework is intentionally solver-free at runtime and does not introduce online constrained optimization or new communication mechanisms. The evaluation reports a controlled five-seed study using seeds 101–105 and a 10-seed stress suite covering scalability, symmetric crossing, corridor, and dense dynamic-obstacle cases. In the controlled nominal evaluation, full three-robot completion occurred in all five runs, with 100.0% mean success, 142.2 mean steps, and no recorded collision timestep. In the hybrid stress suite, nominal, four-robot swap, five-robot crossing, symmetric-deadlock, and corridor cases achieved full success in all 10 seeds. Dense dynamic obstacles were the main failure case, with 5/10 full-success runs, 5 robot timeouts, and 10.1 mean collision events per run. These results support the feasibility of the hybrid structure in moderate tested conditions while showing that dense moving obstacles remain a practical limitation. Formal safety guarantees, matched benchmark comparisons, physical robot validation, and wider randomization remain areas requiring future work.



Academic Editor: David Portugal

Received: 18 June 2026

Revised: 18 July 2026

Accepted: 21 July 2026

Published: 28 July 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

**Keywords:** decentralized multi-robot navigation; reinforcement learning; DDPG; belief state; partial observability; safety blending

## 1. Introduction

In the tested workspace, three e-puck-style robots share a bounded planar area with fixed cubes, debris blocks, and moving spherical obstacles. Each robot receives only local

observations and must still reach its assigned goal. The control problem is therefore not just a shortest-path problem. It also includes local reaction to nearby robots, obstacle evidence that may disappear from view, and motion commands that must remain meaningful for heading-limited forward movement.

Classical planning and control methods can provide strong behavior when accurate models and reliable state information are available. However, global maps, precise localization, and complete obstacle knowledge are not always available in decentralized multi-robot settings. Learning-based navigation methods, including deep reinforcement learning (DRL), reduce dependence on hand-coded environment models by learning policies directly from interaction data [1–3]. However, learned policies can be unsafe in the presence of partial observability, particularly in cases where local observations do not provide sufficient information for discriminating between safe and dangerous actions.

In this paper, we study an architecture for a mixture of learned policies and lightweight safety blending. The key point here is straightforward: generate goal-directed movement using reinforcement learning but do not perform the learned action in the case of close-by objects or other robots. The actual velocity is created from the blend of the learned command with geometric components representing goal and avoidance directions. The observation framework, belief update method, policy learning elements, safety blending algorithm, and performance evaluation methods employed in the proposed controller are presented below.

### *Contributions*

The contributions of the paper can be outlined as follows:

- A decentralized navigation scheme is devised for three robots utilizing only local sensory observations in the absence of online communication and central control.
- Compact state representation is adopted incorporating position, goal orientation, and sector-based proximity features, along with 4D decay-based occupancy representation.
- A DDPG continuous policy is integrated along with a solver-less geometric safety-blending action modifier based on proximity and spacing information.
- A controlled 5-seed evaluation and a 10-seed stress suite are reported, including success, timeout, collision-timestep, collision-event, trajectory, reward, scalability, and corner-case outputs.

## **2. Related Work**

### *2.1. Learning-Based Robot Navigation*

Deep reinforcement learning has been widely studied for robot navigation because it can map observations directly to actions and can optimize long-horizon behavior through trial-and-error interaction [1,2]. Actor–critic approaches are particularly applicable to continuous robot control due to the fact that they result in smooth movements compared to discrete movements. DDPG is an example of a deterministic actor–critic approach for continuous control [3].

Learning-based navigation systems are appealing when there are uncertainties in the environment, but their success is highly dependent on the quality of observations, rewards, and types of scenarios used during training. New research in reinforcement learning involving unmanned and robotic systems still places high importance on safety, disturbance rejection, and control reliability [4,5]. Such problems become especially relevant in multi-robot navigation, where every robot can see only a limited part of the environment.

## 2.2. Decentralized Multi-Robot Navigation

In decentralized multi-robot navigation, the robots select their actions in isolation, using locally available information. This enhances scalability, but it creates an issue of non-stationarity for the environment from the robots' perspective. Prior work on decentralized deep reinforcement learning has shown that local interaction cues can support implicit coordination [6]. However, dense interaction can still lead to collisions, deadlocks, or inefficient trajectories.

The velocity obstacle and reciprocal avoidance concepts provide structured collision avoidance cues for multi-agent motion. Reinforcement learned reciprocal velocity obstacle methods combine such structure with learned policies to improve distributed navigation behavior [7]. Communication-enabled approaches can improve dense coordination, but they also introduce communication assumptions that are not always available in simple decentralized deployments [8].

Recent multi-agent control studies also address formation tracking, obstacle avoidance, disturbance rejection, intermittent communication, and learning-based uncertainty estimation [9–11]. These studies target broader cooperative-control settings such as USV swarms, heterogeneous multi-agent systems, and fault-tolerant platoon-like tracking. In comparison, this work focuses on decentralized waypoint navigation from compact local observations using a DDPG-assisted hybrid controller with explicit geometric safety blending. Table 1 summarizes this positioning.

**Table 1.** Qualitative positioning against recent related multi-agent control studies.

| Study             | Main Focus                                                                                                           | Position Relative to This Work                                                                                                                                                                                                            |
|-------------------|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Chang et al. [9]  | USV formation control with APF obstacle avoidance, intermittent communication, and disturbance observation.          | Related through cooperative obstacle avoidance, but the objective is formation tracking for networked USVs. This work studies decentralized waypoint navigation for mobile robots without online communication.                           |
| Liu and Dong [10] | Fixed-time cooperative fault-tolerant control for heterogeneous agents with uncertainties, faults, and disturbances. | Related through learning-based multi-agent control, but the learning component estimates uncertainties and faults. This work uses learning for local navigation action generation.                                                        |
| Liu and Dong [11] | Learning-based estimation and fault-tolerant control for uncertain heterogeneous multi-agent systems.                | Related through fault-tolerant cooperative control, but it addresses synchronization and compensation. This work evaluates belief-conditioned navigation and reactive safety blending.                                                    |
| This work         | Decentralized waypoint navigation with static/dynamic obstacles and partial observability.                           | Uses compact belief-conditioned observations, a DDPG continuous-action actor, and solver-free geometric safety blending for local navigation. It does not address fault diagnosis, formation maintenance, or disturbance observer design. |

## 2.3. Safe Reinforcement Learning and Hybrid Control

Safe reinforcement learning methods often enforce constraints using safety filters, control barrier functions (CBFs), quadratic programming (QP), or model predictive control

(MPC) shields [12,13]. These mechanisms can reduce unsafe actions and may provide stronger guarantees when their assumptions hold. Their limitation is that they often require online optimization, model rollout, or constraint solving at each control step.

Hybrid methods combine learning-based decisions with classical control or reactive safety rules. Such methods usually do not provide the same formal guarantees as CBF- or MPC-based approaches, but they can reduce implementation and runtime complexity. The present work follows this hybrid direction. The proposed system evaluates whether a compact belief representation and a lightweight safety-blending rule can produce functional decentralized navigation in the tested simulation setting.

### 3. Proposed Framework

#### 3.1. Problem Definition

Consider  $N$  mobile robots moving in a bounded planar workspace  $\mathcal{W} \subset \mathbb{R}^2$ . Robot  $i \in \{1, \dots, N\}$  has position  $x_t^i \in \mathbb{R}^2$  at time  $t$  and goal position  $g^i \in \mathbb{R}^2$ . Each robot receives only local observations and does not communicate with other robots during execution. The objective is to generate a continuous control command,

$$u_t^i \in \mathbb{R}^2 \quad (1)$$

that moves robot  $i$  toward  $g^i$ , avoids obstacles and nearby robots, and maintains smooth motion.

Because each robot observes only local information, the navigation problem is treated as a decentralized partially observable Markov decision process (Dec-POMDP). The Dec-POMDP is not solved exactly. Instead, each robot builds a compact local state vector that acts as an approximate belief-conditioned observation.

The Dec-POMDP is written as

$$\mathcal{M} = \langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}^i\}, \{\mathcal{O}^i\}, T, Z, R, \gamma \rangle, \quad (2)$$

where  $\mathcal{I} = \{1, \dots, N\}$ ,  $\mathcal{S}$  contains robot poses, headings, waypoint indices, obstacle states, and map values,  $\mathcal{A}^i = [-0.2, 0.2]^2$ ,  $\mathcal{O}^i \subset \mathbb{R}^{12}$ ,  $T$  is the heading-limited pose transition with obstacle motion,  $Z$  maps the system state to local observations,  $R$  is the training reward, and  $\gamma = 0.99$ .

#### 3.2. State Representation

The state vector for robot  $i$  is defined as

$$s_t^i = [x_t^i, d_t^i, \ell_t^i, b_t^i] \in \mathbb{R}^{12}, \quad (3)$$

where  $x_t^i \in \mathbb{R}^2$  is the robot position,  $d_t^i \in \mathbb{R}^2$  is the relative goal vector or goal-direction encoding,  $\ell_t^i \in \mathbb{R}^4$  contains sector-based proximity measurements, and  $b_t^i \in \mathbb{R}^4$  contains belief features extracted from a decaying local occupancy map. The dimensionality follows directly from

$$\dim(s_t^i) = 2 + 2 + 4 + 4 = 12. \quad (4)$$

#### 3.3. Belief Encoding

To reduce the effect of partial observability, each robot maintains a compact local occupancy summary  $G_t^i \in \mathbb{R}^4$ :

$$G_t^i = [g_{NE}^i, g_{NW}^i, g_{SW}^i, g_{SE}^i]. \quad (5)$$

The workspace grid is  $10 \times 10$  with 1 m cells. Cell values decay as  $M_t^i(c) = 0.9M_{t-1}^i(c)$ , and detected cells are updated as  $M_t^i(c) = \min(1, M_t^i(c) + 0.5)$ . Each component of  $G_t^i$  is the mean occupancy in one quadrant around robot  $i$ . The earlier  $\Delta G_t^i$  notation is not used in the actor input.

### 3.4. Continuous Policy Learning

The learned policy uses DDPG. The actor network  $\mu_\theta$  maps the local state to a continuous velocity proposal:

$$a_t^i = \mu_\theta(s_t^i). \quad (6)$$

The critic  $Q_\phi(s, a)$  estimates the action-value function during training. For a transition  $(s_t, a_t, r_t, s_{t+1})$ , the critic target is

$$y_t = r_t + \gamma Q_{\phi'}(s_{t+1}, \mu_{\theta'}(s_{t+1})), \quad (7)$$

and the critic loss is

$$L(\phi) = \frac{1}{M} \sum_{m=1}^M (Q_\phi(s_m, a_m) - y_m)^2, \quad (8)$$

where  $M$  is the minibatch size,  $\gamma$  is the discount factor, and primed symbols denote target networks.

### 3.5. Safety Blending

The actor command is modified before execution. Let  $e_g^i$  denote the normalized goal direction and  $e_a^i$  denote a reactive avoidance vector computed from nearby obstacles and robots. The executed command is

$$u_t^i = \alpha_t^i a_t^i + \beta_t^i e_g^i + \eta_t^i e_a^i, \quad (9)$$

where  $\alpha_t^i, \beta_t^i, \eta_t^i \geq 0$  are blending weights. The values used in the evaluation are empirical piecewise constants:

$$(\alpha, \beta, \eta) = \begin{cases} (0.01, 0.01, 0.98), & \text{critical zone,} \\ (0.15, 0.60, 0.25), & d_{\text{goal}} < 0.5 \text{ m,} \\ (0.25, 0.30, 0.45), & \text{otherwise.} \end{cases} \quad (10)$$

The critical zone is  $d < r_{\text{obs}} + 0.35$  m for obstacles and  $d < 0.5$  m for other robots. Thus,  $\eta_t^i$  switches by mode rather than increasing continuously. Distance dependence is mainly encoded in the avoidance vector magnitude. The tangential avoidance gain is  $k_{\text{tan}} = 0.55$ . This rule is reactive and does not solve an online QP, MPC, or CBF optimization problem.

For the e-puck-compatible evaluation, the blended planar direction is interpreted as a desired heading rather than as a direct lateral displacement. The robot yaw is updated by a bounded angular increment, and the forward displacement is reduced when the yaw error is large. This prevents the simulated robot from moving sideways in one step, which would not be physically meaningful for a differential-drive e-puck.

No polynomial trajectory generator is used. The robots are assigned intermediate waypoints between start and goal positions, but the executed path is produced online by the actor command, heading-limited forward motion, and the safety-blending rule. Therefore, the curved trajectories should be interpreted as the result of local reactive navigation under partial observability, not as optimized polynomial paths.

### 3.6. Reward Structure

The reward balances progress, goal completion, smoothness, obstacle proximity, inter-robot spacing, cross-track deviation, and team progress. The constants used in the evaluation are listed in Table 2; they were empirically selected and then fixed for all reported runs.

**Table 2.** Implemented reward components and constants.

| Term                          | Implemented Value                       |
|-------------------------------|-----------------------------------------|
| Distance reward               | $\max(0, 10 - d_t)$                     |
| Progress reward               | $50(d_{t-1} - d_t)$                     |
| Goal bonus                    | +5 if $d_t < 0.5$ m                     |
| Smoothness penalty            | $2\ a_t - a_{t-1}\ $                    |
| Cross-track penalty           | $2e_{\text{cross}}$                     |
| Team spacing penalty          | $3(0.8 - d_{ij})$ , if $d_{ij} < 0.8$ m |
| Obstacle collision penalty    | 100 inside the collision threshold      |
| Inter-robot collision penalty | 100 if robot distance is below 0.5 m    |

### 3.7. Execution Procedure

The online execution sequence is given in Algorithm 1. At each control step, every active robot forms its local belief-conditioned state, evaluates the DDPG actor, blends the actor proposal with goal and avoidance directions, and then applies the bounded command. The episode ends when all robots reach their goals or when the maximum horizon is reached.

#### Algorithm 1 Belief-Driven Hybrid Decentralized Navigation

**Require:** Robot states, goal positions, trained actor  $\mu_\theta$ , maximum horizon  $T$

```

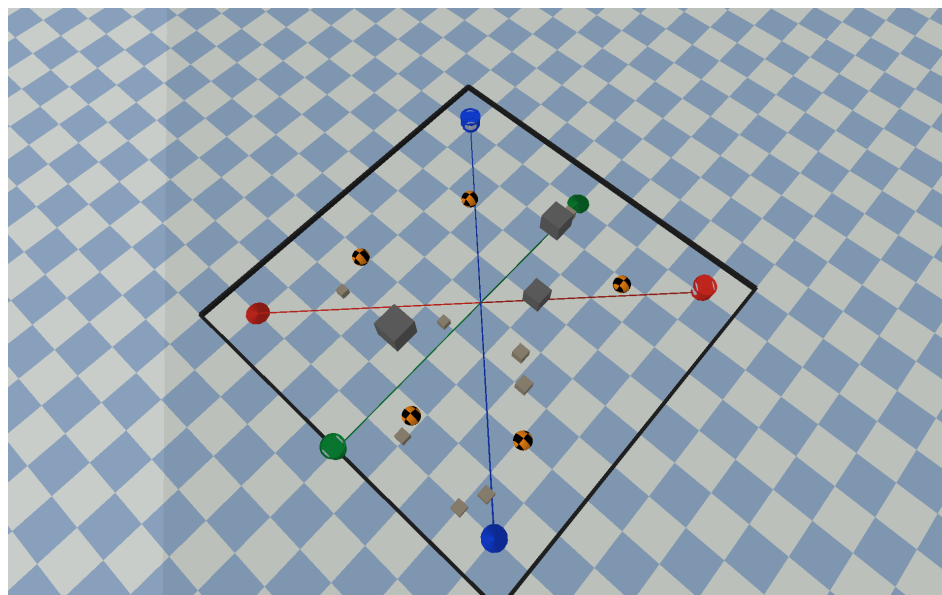
1: for  $t = 1$  to  $T$  do
2:   for each active robot  $i$  do
3:     Acquire local proximity observations
4:     Update  $G_t^i = \lambda G_{t-1}^i + \Delta G_t^i$ 
5:     Build  $s_t^i = [x_t^i, d_t^i, \ell_t^i, b_t^i]$ 
6:     Compute actor command  $a_t^i = \mu_\theta(s_t^i)$ 
7:     Compute goal direction  $e_g^i$  and avoidance direction  $e_a^i$ 
8:     Blend command using Equation (9) and enforce action bounds
9:     Execute  $u_t^i$  for robot  $i$ 
10:  end for
11:  if all robots have reached their goals then
12:    break
13:  end if
14: end for

```

## 4. Experimental Setup

The evaluation uses a simulated workspace with three e-puck mobile robots, as shown in Figure 1. The scene includes three fixed cube obstacles, eight randomly placed debris blocks, and five moving spherical obstacles. Each robot has its own goal and runs the same decentralized controller from local observations. The recorded metrics include total steps, robot success, timeout count, collision count, path length, and final reward.

The controlled evaluation uses five fixed random seeds, 101–105. Each run uses the same trained DDPG policy with deterministic evaluation and stops a robot after 800 robot steps if it has not reached its goal. Direct comparisons with external benchmark baselines require matched protocols and are therefore left for future work.



**Figure 1.** Simulation workspace used for the controlled runs: three e-puck mobile robots, fixed cubes, debris blocks, and moving spheres.

The available robot description provides simulation geometry, mass, and inertia terms for the e-puck links. Hardware specifications such as payload capacity, motor torque, electrical power, and drive-motor thermal limits are outside the claimed evaluation scope. Table 3 lists the main simulation parameters, and Table 4 gives the start, goal, and waypoint values used in the controlled runs.

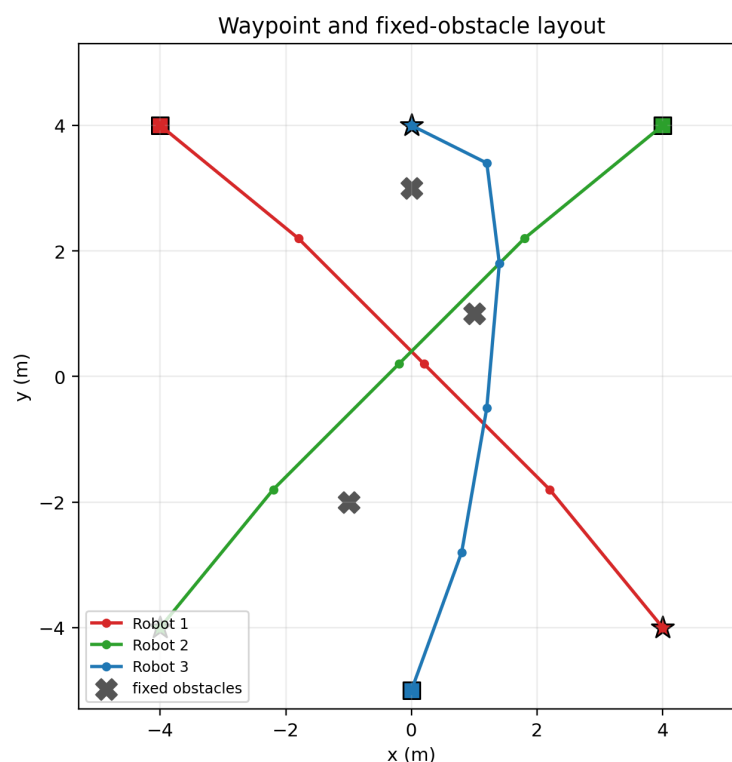
**Table 3.** Simulation and robot parameters used in the controlled evaluation.

| Parameter                  | Value Used in Simulation                                       |
|----------------------------|----------------------------------------------------------------|
| Simulator                  | PyBullet                                                       |
| Robot model                | e-puck mobile robot model                                      |
| Number of robots           | 3                                                              |
| Workspace bounds           | $x, y \in [-5, 5]$ m                                           |
| Start positions            | $(-4, 4), (4, 4), (0, -5)$ m                                   |
| Goal positions             | $(4, -4), (-4, -4), (0, 4)$ m                                  |
| Nominal position increment | 0.1 m per active robot step                                    |
| Action bounds              | $[-0.2, 0.2]$ per component                                    |
| Maximum yaw update         | 0.35 rad per active robot step                                 |
| Minimum forward scale      | 0.15 before heading-error gating                               |
| Local sensing range        | 2.0 m                                                          |
| Waypoint structure         | Lane waypoints for robots 1–2, clearance waypoints for robot 3 |
| Per-robot timeout          | 800 robot steps                                                |
| Obstacles                  | 3 fixed cubes, 8 random debris blocks, 5 moving spheres        |
| Controlled seeds           | 101, 102, 103, 104, 105                                        |

**Table 4.** Start, goal, and waypoint structure used in the e-puck-compatible evaluation.

| Robot | Start (m) | Goal (m)   | Intermediate Waypoints (m)                         |
|-------|-----------|------------|----------------------------------------------------|
| 1     | $(-4, 4)$ | $(4, -4)$  | $(-1.8, 2.2), (0.2, 0.2), (2.2, -1.8)$             |
| 2     | $(4, 4)$  | $(-4, -4)$ | $(1.8, 2.2), (-0.2, 0.2), (-2.2, -1.8)$            |
| 3     | $(0, -5)$ | $(0, 4)$   | $(0.8, -2.8), (1.2, -0.5), (1.4, 1.8), (1.2, 3.4)$ |

Figure 2 shows the corresponding waypoint layout and fixed-obstacle positions.



**Figure 2.** Waypoint and fixed-obstacle setup for the controlled e-puck-compatible evaluation. Squares represent starting locations, stars represent goal locations, and crosses represent fixed cube obstacles.

## 5. Results

### 5.1. Controlled Evaluation

Table 5 shows the controlled five-seed e-puck compatible evaluation results. In all the five rounds, the full execution is achieved without exceeding per-robot timeout. There was no recorded collision during the evaluation round. Table 6 gives the aggregate values over the same seed set.

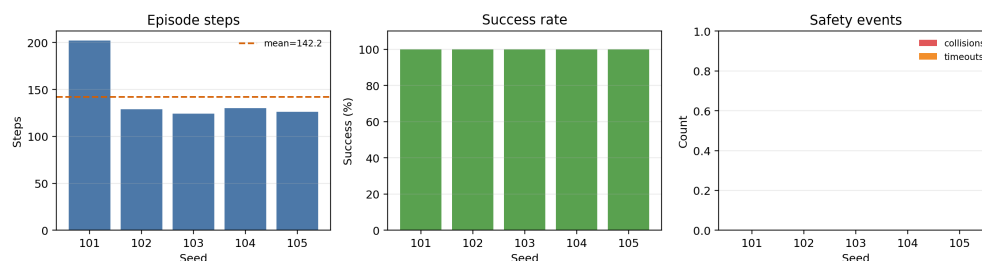
**Table 5.** Controlled e-puck-compatible evaluation over five fixed seeds.

| Seed | Total Steps | Successful Robots | Timed-Out Robots | Total Collisions |
|------|-------------|-------------------|------------------|------------------|
| 101  | 202         | 3/3               | 0/3              | 0                |
| 102  | 129         | 3/3               | 0/3              | 0                |
| 103  | 124         | 3/3               | 0/3              | 0                |
| 104  | 130         | 3/3               | 0/3              | 0                |
| 105  | 126         | 3/3               | 0/3              | 0                |

**Table 6.** Controlled-evaluation metrics for seeds 101–105 in the e-puck-compatible simulation.

| Metric                           | Value over Seeds 101–105 |
|----------------------------------|--------------------------|
| Three robot complete runs        | 5/5 runs                 |
| Mean success rate                | 100.0%                   |
| Mean successful robots per run   | 3.0/3                    |
| Mean timed-out robots per run    | 0.0/3                    |
| Mean episode length              | 142.2 steps              |
| Minimum / maximum episode length | 124/202 steps            |
| Mean collision count             | 0.0                      |
| Best run by steps                | Seed 103, 124 steps      |

Five runs of the evaluation resulted in 3/3 complete robots per seed. The average number of steps in each episode was 142.2, with no timeouts or collisions detected during evaluation. The run with seed 103 had the fewest number of steps, at 124, while seed 101 had 202 due to the lengthy path of robot 2. Figure 3 plots the same five-seed summary.



**Figure 3.** Seeds 101–105 in the controlled evaluation. Each run met all three out of the three goals, and the collision and timeout traces stayed at zero.

Table 7 reports the final robot-level values for seed 105. Seed 105 ended after 126 simulation steps. The recorded outcome was successful robots = 3/3, timed-out robots = 0/3, and total collisions = 0. Table 8 further reports path-ratio values for the same run.

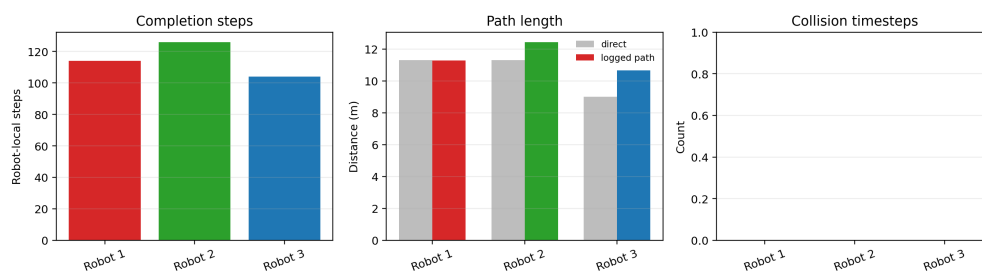
**Table 7.** Seed 105 final controlled run. Robot labels match the trajectory colors used in the following figures.

| Robot       | Steps | Collisions | Path Length (m) | Final Reward |
|-------------|-------|------------|-----------------|--------------|
| Red robot   | 114   | 0          | 11.29           | 1966.0       |
| Green robot | 126   | 0          | 12.44           | 1761.9       |
| Blue robot  | 104   | 0          | 10.66           | 1613.1       |

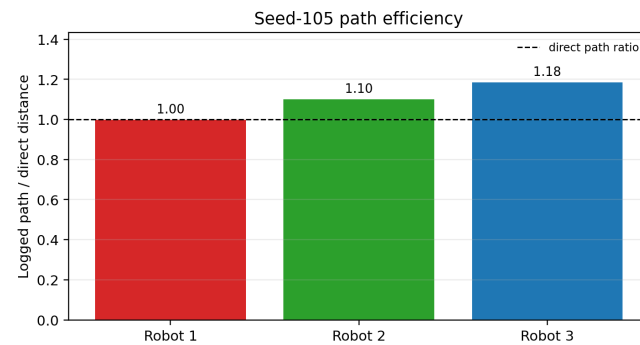
**Table 8.** Robot-level path and collision values for seed 105. Path ratio is the measured path length divided by direct start–goal distance.

| Robot       | Steps | Direct Distance (m) | Path Length (m) | Path Ratio/<br>Collisions |
|-------------|-------|---------------------|-----------------|---------------------------|
| Red robot   | 114   | 11.31               | 11.29           | 1.00 / 0                  |
| Green robot | 126   | 11.31               | 12.44           | 1.10 / 0                  |
| Blue robot  | 104   | 9.00                | 10.66           | 1.18 / 0                  |

For seed 105, the red robot was the one that stayed closest to the straight-line distance between its start and goal position with a path ratio of 1.00. However, the blue robot had a path ratio of 1.18 since its chosen path included clearance via waypoints from the obstacle at (0,3). It still finished in 104 local steps, fewer than the red and green robots. Figures 4 and 5 show these robot-level values graphically.



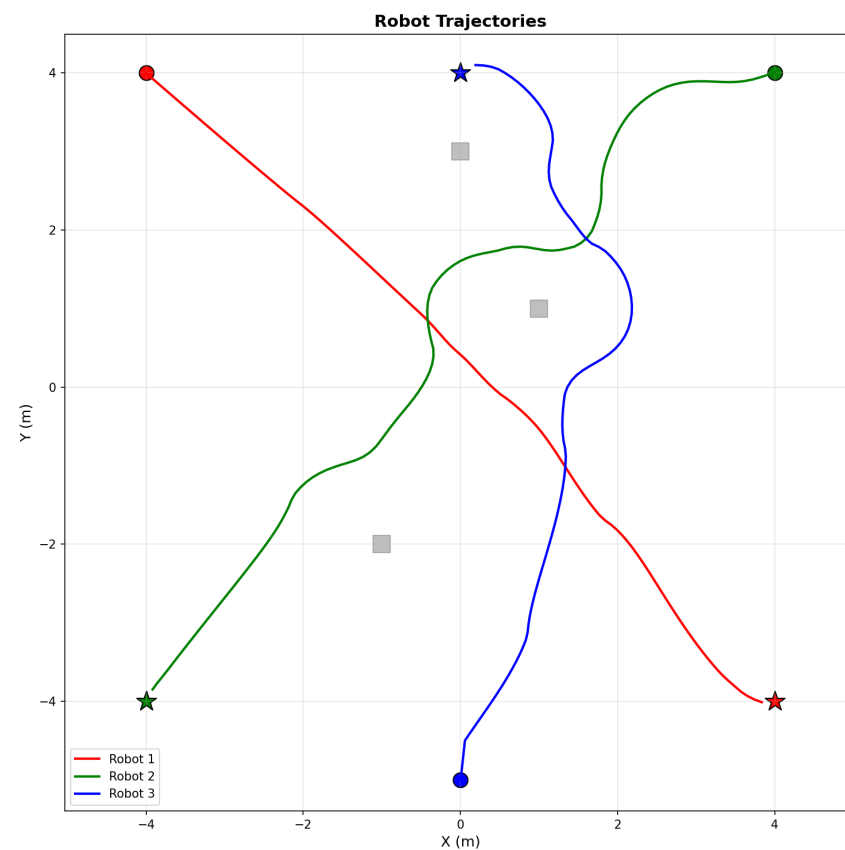
**Figure 4.** Seed 105 robot-level plot with completion steps, measured path length, direct start–goal distance, and collision events.



**Figure 5.** Path-efficiency ratio for seed 105. Path efficiency ratios closer to 1 indicate paths that approach the optimal path between start and goal.

### 5.2. Trajectory, Collision, and Reward Behavior

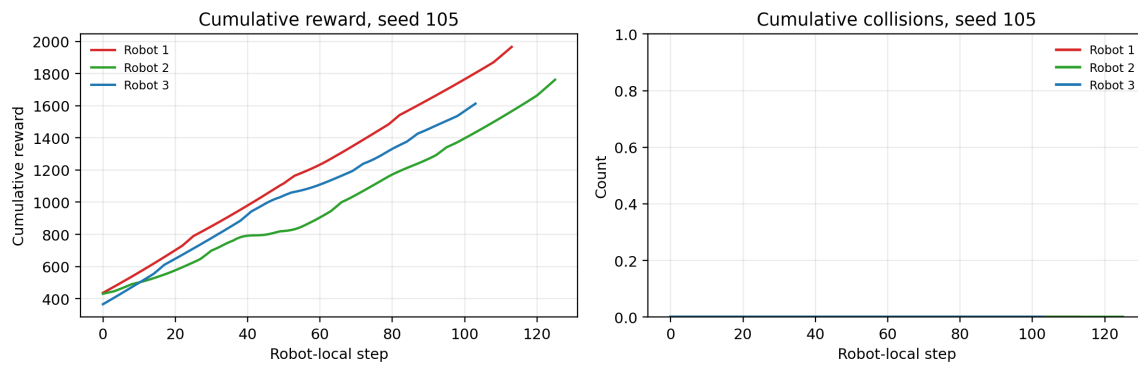
Figure 6 shows the robot trajectories for seed 105. Trajectories adhere to the corridor waypoints and retain local curvatures from the heading-constrained control and reactive collision.



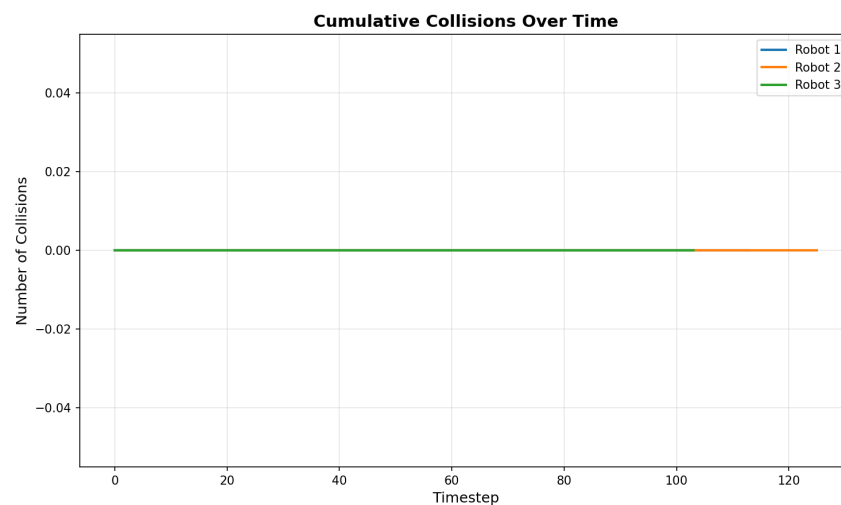
**Figure 6.** Robot trajectories in the representative run for seed 105, compatible with the e-puck simulator.

Figure 7 combines cumulative reward and cumulative collision traces for seed 105. The collision trace remains at zero for all robots.

Figure 8 gives the cumulative collision trace separately for the same trial.

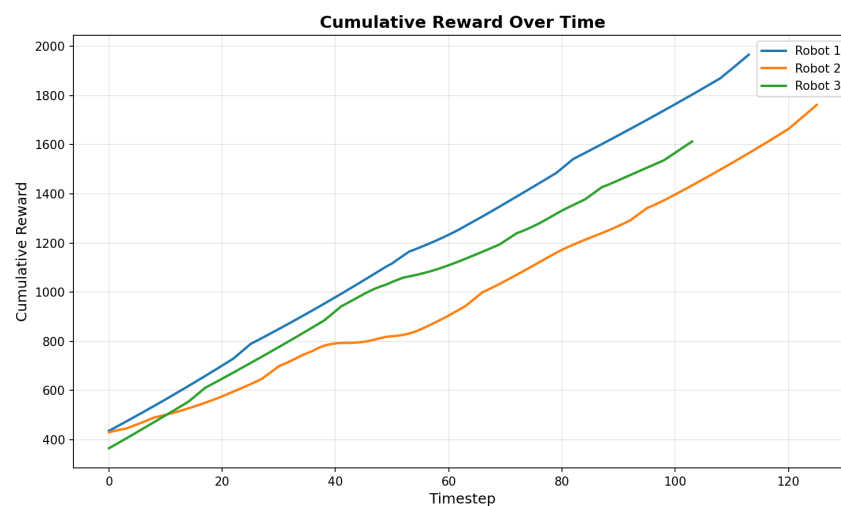


**Figure 7.** Cumulative reward and cumulative collision traces for the representative seed 105 run.



**Figure 8.** Cumulative collisions graph for the exemplary seed-105 trial.

Figure 9 below is a representation of cumulative reward behavior. Reward fluctuations depend on goal achievement, waypoint switching, obstacles, and local control mechanisms



**Figure 9.** Cumulative reward curves for the three robots during the representative seed 105 simulation run.

### 5.3. Controlled Evaluation Summary

Table 9 is a summary of some of the recordings from the controlled experiment. Seed 103 provided the quickest controlled performance in this series, whereas seed 105 is the sample final run for trajectory, collision, and reward graphs.

**Table 9.** Selected records from the controlled five-seed evaluation.

| Record                  | Seed | Steps | Success | Timeout | Collisions |
|-------------------------|------|-------|---------|---------|------------|
| First controlled run    | 101  | 202   | 3/3     | 0/3     | 0          |
| Shortest controlled run | 103  | 124   | 3/3     | 0/3     | 0          |
| Final controlled run    | 105  | 126   | 3/3     | 0/3     | 0          |

The controlled evaluation summary from Figure 10 also describes the same five runs. The biggest difference in that summary is the episode duration, rather than the success and collision pattern. Only seed 101 exceeds 200 steps, while the other four finish within the range of 124 to 130 steps.



**Figure 10.** Controlled evaluation summary for the five fixed-seed e-puck-compatible runs. Color groups distinguish the success, timeout, collision, step-count, path-length, and reward panels.

### 5.4. Comparative Evaluation of Controller Variants

A comparative evaluation was carried out to assess the effect of the final hybrid control strategy relative to a legacy DDPG-dominant controller. The legacy controller used the same actor-based navigation structure but relied on the earlier geometric blending rule. Both controller variants were evaluated with the nominal three-robot start-goal layout and the same five fixed seeds, 101–105.

The controller-variant comparison shows that the final hybrid controller removed the timeout cases observed with the legacy DDPG-dominant controller. The mean episode length also decreased from 621.4 steps to 142.2 steps. This indicates that the final blending and waypoint-recovery logic contributed to reliable completion in the nominal three-robot setting, rather than only preserving the behavior of the learned actor. Table 10 gives the numerical comparison.

**Table 10.** Comparative evaluation of controller variants over five fixed seeds.

| Controller Variant              | Full Success | Mean Success (%) | Mean Steps | Mean Collision Count | Timeouts |
|---------------------------------|--------------|------------------|------------|----------------------|----------|
| Legacy DDPG-dominant controller | 3/5          | 86.7             | 621.4      | 4.0                  | 2        |
| Final hybrid controller         | 5/5          | 100.0            | 142.2      | 0.0                  | 0        |

### 5.5. Stress and Corner-Case Evaluation

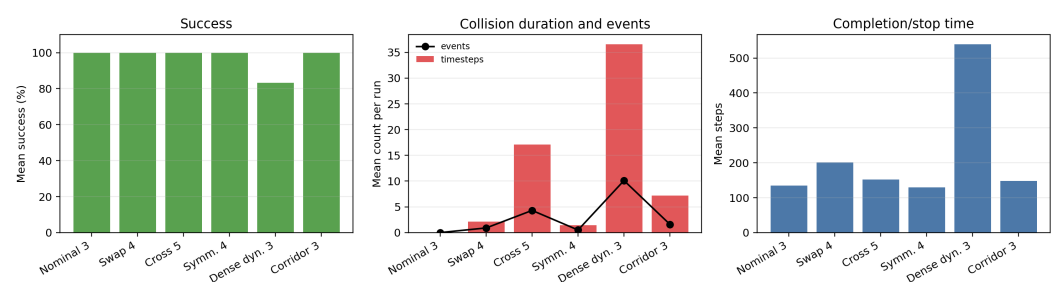
The stress suite then used 10 seeds, 101–110, for each scenario. It changed robot count, crossing symmetry, corridor layout, and dynamic-obstacle density while keeping the same hybrid controller. Collision timesteps count control steps with active contact. Collision events use a cooldown counter, so one continuous contact is not counted again at every timestep.

Table 11 shows that robot count alone was not the dominant failure source in these tests. The four-robot swap and five-robot crossing cases completed in all 10 seeds. However, the five-robot crossing case had a higher collision-event count than the nominal and four-robot cases, indicating that denser robot interactions increase safety-layer activity even when goals are reached. Dense dynamic obstacles produced the clearest failure mode: only 5 of the 10 runs completed all robots, and 5 robot timeouts occurred. Figure 11 visualizes the stress-test summary.

Table 12 extracts the robot-count subset from the stress suite.

**Table 11.** Ten-seed stress and corner-case evaluation of the hybrid controller. CT denotes collision timesteps and CE denotes cooldown-based collision events.

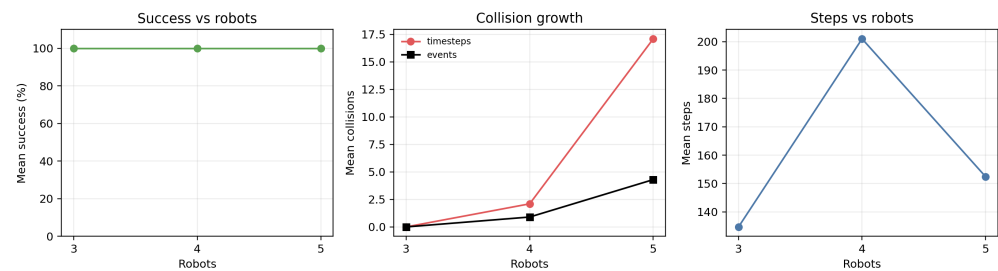
| Scenario                | Robots | Full Success | Mean Success | Mean Steps | Mean CT/CE  | Timeouts |
|-------------------------|--------|--------------|--------------|------------|-------------|----------|
| Nominal                 | 3      | 10/10        | 100.0%       | 134.7      | 0.0 / 0.0   | 0        |
| Four-robot swap         | 4      | 10/10        | 100.0%       | 201.0      | 2.1 / 0.9   | 0        |
| Five-robot crossing     | 5      | 10/10        | 100.0%       | 152.4      | 17.1 / 4.3  | 0        |
| Symmetric deadlock      | 4      | 10/10        | 100.0%       | 130.0      | 1.4 / 0.5   | 0        |
| Corridor crossing       | 3      | 10/10        | 100.0%       | 148.0      | 7.2 / 1.6   | 0        |
| Dense dynamic obstacles | 3      | 5/10         | 83.3%        | 539.9      | 36.6 / 10.1 | 5        |

**Figure 11.** Hybrid controller stress test summary over 10 seeds per scenario. The dense dynamic-obstacle case is the main failure condition.

**Table 12.** Scalability subset extracted from the stress suite. The five-robot crossing case completed all runs, but collision activity increased.

| Scenario            | Robots | Full Success | Mean Steps | Mean CT | Mean CE |
|---------------------|--------|--------------|------------|---------|---------|
| Nominal             | 3      | 10/10        | 134.7      | 0.0     | 0.0     |
| Four-robot swap     | 4      | 10/10        | 201.0      | 2.1     | 0.9     |
| Five-robot crossing | 5      | 10/10        | 152.4      | 17.1    | 4.3     |

Figure 12 shows the corresponding scalability trend.



**Figure 12.** Scalability trend for the three-, four-, and five-robot stress cases. Success remains at 100%, while collision timesteps and collision events increase for the five-robot crossing case.

Table 13 summarizes the main observed strengths and limitations from the stress suite.

**Table 13.** Main observed strengths and limitations from the stress suite.

| Observed Strength                                                                                                 | Observed Limitation                                                                                        |
|-------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| Completed all nominal, four-robot swap, five-robot crossing, symmetric-deadlock, and corridor runs over 10 seeds. | Dense dynamic obstacles caused 5/10 full-success runs, 5 timeouts, and the largest collision-event count.  |
| No online QP, MPC, CBF, or polynomial optimization was required at runtime.                                       | The reactive safety layer does not provide formal invariance or worst-case collision-avoidance guarantees. |
| Heading-limited execution removes earlier side-slip behavior and is more consistent with e-puck-style motion.     | Hardware validation is still required before physical deployment.                                          |
| The method scales to five robots in the crossing test without timeout.                                            | Collision activity rises in denser robot interactions, even when all goals are reached.                    |

## 6. Discussion

The controlled e-puck-compatible results show that the hybrid controller can guide all three robots to their goals in the tested nominal configuration. Seeds 101–105 completed all three goals with no timeout and no collision timestep. The larger stress suite gives a broader view: the controller also completed all runs in the four-robot swap, five-robot crossing, symmetric deadlock, and corridor scenarios over 10 seeds per scenario. However, the dense dynamic-obstacle scenario exposed the main limitation, with 5 full-success runs out of 10 and 5 robot timeouts.

Another factor was the change from lateral pose updates to heading-limited forward motion. Earlier side-slip behavior gave trajectories that were not suitable for e-puck interpretation. The revised controller constrains yaw change and reduces forward motion when the desired direction is far from the current heading. This led to more realistic path execution for a differential-drive robot.

### *Physical Robot Implementation Considerations*

The current evaluation is limited to simulation, and physical robot execution would require additional validation before the controller can be treated as hardware-ready. The main practical issues are as follows:

- The simulation bypasses robot acceleration, braking, and turning dynamics. The critical-zone avoidance rule may therefore provide less clearance on hardware when the robot cannot instantly rotate or stop.
- The DDPG actor uses idealized position, waypoint, LiDAR-sector, and belief features. Localization or feature-construction errors may produce actor inputs outside the training distribution.
- The avoidance weights, distance thresholds, tangential gain, and heading limits are simulation-tuned and require hardware-specific validation because they depend on robot speed and control-update rate.
- Hardware testing must also account for wheel slip, floor friction variation, sensing latency, occlusion, battery voltage changes, and mechanical tolerances. These are common real-world challenges for multi-robot navigation, not limitations specific to the proposed controller.

## 7. Limitations

The limitations of this study are as follows:

- The evaluation uses fixed seed sets and manually specified waypoint corridors; broader randomized testing would further characterize generalization.
- The safety-blending layer is reactive and empirical, so it does not provide formal collision avoidance guarantees.
- Dense dynamic obstacles remain the most difficult tested case, and physical robot validation is outside the current scope.

## 8. Conclusions

A belief-driven hybrid reinforcement learning framework was evaluated for decentralized multi-robot navigation under partial observability. The method combines a compact belief-conditioned state representation, a DDPG-based continuous policy, and a lightweight geometric safety-blending layer. In the controlled five-seed e-puck-compatible evaluation, full three-robot completion occurred in all five runs. The mean episode length was 142.2 steps, and no collision timestep was recorded in the controlled evaluation. The controller-variant comparison showed that the final hybrid controller improved full-success runs from 3/5 to 5/5 and reduced mean episode length from 621.4 to 142.2 steps relative to the legacy DDPG-dominant controller. A larger 10-seed stress suite showed full completion for nominal, four-robot swap, five-robot crossing, symmetric-deadlock, and corridor scenarios, but dense dynamic obstacles reduced full-run success to 5/10 and produced five timeouts.

Future work should include wider multi-seed evaluations, stronger collision-free safety guards, improved handling of dense moving obstacles, hardware-level runtime profiling, and physical robot validation.

**Author Contributions:** Conceptualization, V.M. and P.S.; writing—original draft preparation, V.M.; writing—review and editing, V.M., R.P.R. and V.A.K.; visualization, V.M. and R.P.R.; editing, A.L.S. and G.U.; supervision, L.P.; project administration, P.S. and G.U. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The data are not publicly available due to internal project restrictions.

**Acknowledgments:** The authors gratefully acknowledge Sri Mata Amritanandamayi Devi (Amma), Amrita Vishwa Vidyapeetham, for her inspiration and for providing financial support for the Article Processing Charges (APC) of this publication. During the preparation of this manuscript, the authors used OpenAI (GPT-5) for language editing, formatting assistance, and document review. The authors reviewed and edited the output and take full responsibility for the content of this publication.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Tai, L.; Paolo, G.; Liu, M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 31–36.
2. Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J.J.; Gupta, A.; Fei-Fei, L.; Farhadi, A. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In Proceedings of the IEEE International Conference on Robotics and Automation, Singapore, 29 May–3 June 2017; pp. 3357–3364.
3. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1509.02971.
4. Gao, Z.; Hu, J.; Liu, Y.; Li, H. Event-Driven Prescribed Optimal Disturbance Rejection for Dynamic Positioning of Ships via Reinforcement Learning. *IEEE Trans. Autom. Sci. Eng.* **2026**, *23*, 4615–4626. <https://doi.org/10.1109/TASE.2026.3660669>.
5. Timóteo, R.; Barbosa, J.; Jesus, I.S. Robotic Arm Control Using a Q-Learning Reinforcement Algorithm. *Robotics* **2026**, *15*, 50. <https://doi.org/10.3390/robotics15030050>.
6. Long, P.; Liu, W.; Pan, J. Towards Optimally Decentralized Multi-Robot Navigation via Deep Reinforcement Learning. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, Australia, 21–25 May 2018; pp. 6252–6259.
7. Han, R.; Chen, S.; Wang, S.; Zhang, Z.; Gao, R.; Hao, Q.; Pan, J. Reinforcement Learned Distributed Multi-Robot Navigation with Reciprocal Velocity Obstacle Shaped Rewards. *IEEE Robot. Autom. Lett.* **2022**, *7*, 5896–5903. <https://doi.org/10.1109/lra.2022.3161699>.
8. Arul, S.H.; Bedi, A.S.; Manocha, D. DMCA: Dense Multi-agent Navigation using Attention and Communication. *arXiv* **2022**, arXiv:2209.06415. <https://doi.org/10.48550/arXiv.2209.06415>.
9. Chang, Z.; Zong, G.; Wang, W.; Yue, M.; Zhao, X. Formation Control and Obstacle Avoidance Design for Networked USV Swarm With Exogenous Disturbance Under Intermittent Communication. *IEEE Trans. Netw. Sci. Eng.* **2025**, *12*, 3234–3243. <https://doi.org/10.1109/TNSE.2025.3558548>.
10. Liu, J.; Dong, J. Fixed-Time Cooperative Fault-Tolerant Control of Heterogeneous Multiagent Systems via Learning-Based Uncertainty Estimation. *IEEE Internet Things J.* **2025**, *12*, 42803–42812. <https://doi.org/10.1109/JIOT.2025.3594023>.
11. Liu, J.; Dong, J. A Learning-Based Estimation and Fault-Tolerant Control Scheme for Uncertain Heterogeneous Multi-Agent Systems. *IEEE Trans. Autom. Sci. Eng.* **2026**, *23*, 9617–9626. <https://doi.org/10.1109/TASE.2026.3692974>.
12. Liu, Z.; Chen, Y.; Wu, J.; Liu, M. Learning Observation-Based Certifiable Safe Policy for Decentralized Multi-Robot Navigation. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 23–27 May 2022; pp. 1178–1184.
13. Vinod, A.P.; Safaoui, S.; Chakrabarty, A.; Quirynen, R.; Yoshikawa, N.; Di Cairano, S. Safe Multi-Agent Motion Planning via Filtered Reinforcement Learning. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 23–27 May 2022; pp. 7270–7276. <https://doi.org/10.1109/ICRA46639.2022.9812259>.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.