


## Article

# Multiagent Reinforcement Learning for Active Guidance Control of Railway Vehicles with Independently Rotating Wheels

Juyao Wei , Zhenggang Lu, Zheng Yin and Zhipeng Jing

Institute of Rail Transit, Tongji University, Shanghai 201804, China

\* Correspondence: 1810988@tongji.edu.cn

**Abstract:** This paper presents a novel data-driven multiagent reinforcement learning (MARL) controller for enhancing the running stability of independently rotating wheels (IRW) and reducing wheel–rail wear. We base our active guidance controller on the multiagent deep deterministic policy gradient (MADDPG) algorithm. In this framework, each IRW controller is treated as an independent agent, facilitating localized control of individual wheelsets and reducing the complexity of the required observations. Furthermore, we enhance the MADDPG algorithm with prioritized experience replay (PER), resulting in the PER-MADDPG algorithm, which optimizes training convergence and stability by prioritizing informative experience samples. In this paper, we compare the PER-MADDPG algorithm against existing controllers, demonstrating the superior simulation performance of the proposed algorithm, particularly in terms of self-centering capability and curve-negotiation behavior, effectively reducing the wear number. We also develop a scaled IRW vehicle for active guidance experiments. The experimental results validate the enhanced running performance of IRW vehicles using our proposed controller.

**Keywords:** active guidance control; independently rotating wheels (IRW); prioritized experience replay (PER); multiagent deep deterministic policy gradient (MADDPG)



**Citation:** Wei, J.; Lu, Z.; Yin, Z.; Jing, Z. Multiagent Reinforcement Learning for Active Guidance Control of Railway Vehicles with Independently Rotating Wheels. *Appl. Sci.* **2024**, *14*, 1677. <https://doi.org/10.3390/app14041677>

Academic Editor: Sakdirat Kaewunruen

Received: 2 January 2024

Revised: 16 February 2024

Accepted: 18 February 2024

Published: 19 February 2024



**Copyright:** © 2024 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) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Independently rotating wheels (IRW) are increasingly utilized in rail transit systems. Compared with solid-axle wheelsets, the unique structure of IRWs allows individual wheels on the same axle to rotate independently, eliminating the longitudinal creep forces generated from the wheel–rail contact interface and suppressing hunting motion [1–4]. However, IRWs no longer exhibit restoring torque in the yaw direction due to the decoupling of the wheels, which leads to dynamical instability and significantly increased wheel–rail wear [5].

Scholars have proposed various active solutions to restore the steering ability of IRWs. The basic steering strategy mimics the dynamic behavior of solid-axle wheelsets [6]. Controllers based on the proportional–integral–derivative (PID) have been extensively studied [7,8], but adjusting the gain parameters under varying vehicle operating conditions is difficult. The robust  $H_\infty$  algorithm is applied by representing nonlinear characteristics with uncertain parameters [9]. However, a trade-off between robustness and performance leads to a conservative controller. Other model-based controllers, such as sliding mode control [10], linear parameter-varying (LPV) control [11], and gain scheduling techniques [12], often rely on simplified mathematical models that have difficulty capturing the intricate dynamics of railway vehicles, especially the nonlinear interactions at the wheel–rail interface and changing operating conditions. This results in a significant gap between the reference model and the actual dynamics, leading to a lack of robustness in real applications. Recognizing the limitations of traditional control strategies, there is a growing interest in data-driven methods. Our previous research explored the application of deep reinforcement learning (DRL)-based controllers, including the deep deterministic policy gradient

(DDPG) and Ape-X DDPG [13,14], leveraging deep neural networks' ability to fit nonlinear systems. Nevertheless, several limitations are encountered: (a) the existing DRL-based controllers require multiple dynamic parameters from all IRWs, and the high dimensionality of observation spaces leads to slow convergence during training; (b) current strategies mainly focus on the centralized control of the entire vehicle without achieving local control for an individual IRW, potentially affecting computational efficiency in practical applications.

To overcome the limitations of current DRL controllers for IRWs, exploring multiagent reinforcement learning (MARL) algorithms is a promising area. MARL-based controllers can simultaneously enhance the adaptability advantages of DRL for complex systems and achieve local control of each IRW via the distributed control method [15,16]. Based on the centralized training and decentralized execution (CTDE) framework of the MARL algorithm, the multiagent deep deterministic policy gradient (MADDPG) algorithm was introduced [17]. The MADDPG equips each agent with individual actor and critic networks, enabling the consideration of other agents' influences while independently evaluating actions. This approach facilitates coordinated learning during training and independent decision making during execution. Thus, MADDPG enhances the global performance in multiagent scenarios, as evidenced in different MAS control issues [18–20]. In this study, we utilize the MARL algorithm to develop a novel distributed active guidance controller for IRW vehicles. Our decentralized approach assigns modular controllers to each independently rotating wheelset, enabling local control within a sensor–controller–motor subsystem. We adopt the MADDPG algorithm and integrate it with the prioritized experience replay (PER) mechanism, which we refer to as the PER-MADDPG algorithm. This approach efficiently optimizes the behavior policy of each agent by interacting with the vehicle's operating environment, aiming to maximize the IRWs' performance and the overall reward of the MARL model.

Our research contributions are summarized as follows: (a) We propose applying the MARL algorithm for distributed active guidance control of IRW vehicles and establishing a fully cooperative MARL framework for training each agent's local controllers. (b) Our research develops an MADDPG-based control algorithm founded on the CTDE framework, enhancing training stability by enabling each agent's critic network to access and optimize behavior strategies based on the global state and action information. (c) The proposed PER-MADDPG algorithm employs the PER mechanism to improve sample efficiency, prioritizing high-value experiences based on their temporal difference (TD) error for accelerated learning and training convergence.

## 2. System Modeling

### 2.1. Dynamic Model of Railway Vehicles with IRWs

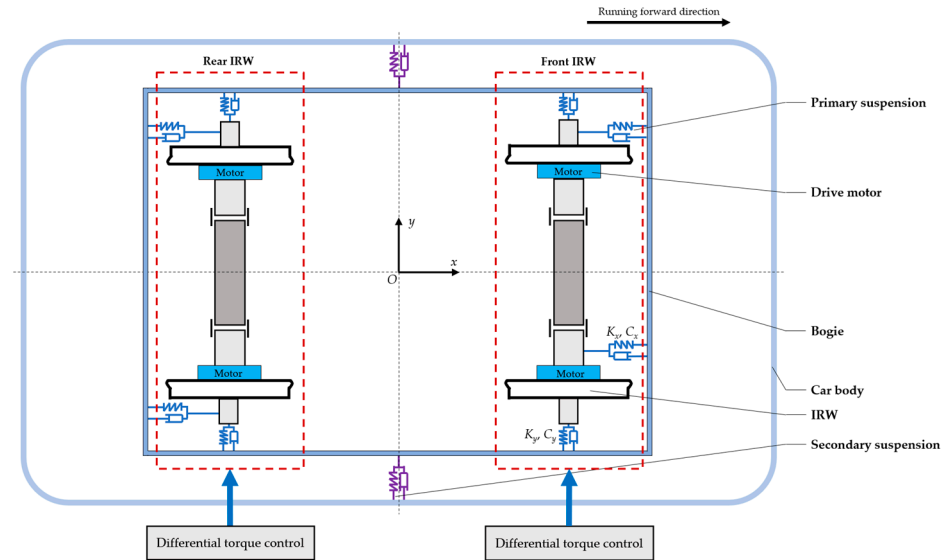
In this paper, we consider a two-axle prototype IRW vehicle, as shown in Figure 1. This vehicle comprises a car body and two sets of independently rotating wheelsets, and the corresponding parameters are detailed in Table A1. It has ten degrees of freedom, including the rotational dynamics of each IRW and the lateral position and yaw movements of both the car body and the wheelsets. Each IRW is driven by a motor providing guidance torque, typically using a permanent magnet synchronous motor (PMSM) in engineering practice. The wheel profile of each IRW is LMA, and the corresponding rail profile is UIC60.

The dynamic equations of lateral and yaw movements for the car body are shown in Equations (1) and (2), respectively.

$$m_b \ddot{y}_b + 2C_y(2\dot{y}_b - \dot{y}_{w1} - \dot{y}_{w2}) + 2K_y(2y_b - y_{w1} - y_{w2}) = \frac{m_b V_0^2}{2} \left( \frac{1}{R_{c1}} + \frac{1}{R_{c2}} \right) - \frac{m_b g}{2} (\theta_1 + \theta_2) \quad (1)$$

$$I_b \ddot{\psi}_b + 4(L_b^2 C_y + L_a^2 C_x) \dot{\psi}_b + 4(L_b^2 K_y + L_a^2 K_x) \psi_b - 2L_b C_y (\dot{y}_{w1} - \dot{y}_{w2}) - 2L_b K_y (y_{w1} - y_{w2}) - 2L_a^2 K_x (\psi_{w1} + \psi_{w2}) - 2L_a^2 C_x (\dot{\psi}_{w1} + \dot{\psi}_{w2}) = 0 \quad (2)$$

where  $L_a$  is half of the lateral length of the primary suspension,  $L_b$  is half of the wheelbase,  $m_b$  is the mass of the car body, and  $I_b$  is the rotational inertia of the car body around the  $z$ -axis.  $K_x$  and  $K_y$  are the longitudinal and lateral stiffnesses of the primary suspension, respectively.  $C_x$  and  $C_y$  are the longitudinal and lateral damping of the primary suspension, respectively.  $R_{ci}$  is the radius of the curved track, and  $\theta_i$  is the cant angle of the curve.  $y_b$  and  $\psi_b$  are the lateral displacement and yaw angle of the car body, respectively.  $y_{wi}$  and  $\psi_{wi}$  are the lateral displacement and yaw angle of each wheelset, respectively.



**Figure 1.** The railway vehicle model with independently rotating wheels (IRW).

The dynamic equations of the IRWs are shown in Equations (3)–(5), where  $i = 1$  and 2 represent the front and rear, respectively, of the independently rotating wheelsets.

$$m_w \ddot{y}_{wi} + \frac{2f_{22}}{V_0} \dot{y}_{wi} + 2C_y(\dot{y}_{wi} - \dot{y}_b) + 2K_y(y_{wi} - y_b) + K_g y_{wi} - 2f_{22} \psi_{wi} + (-1)^i \cdot 2L_b C_y \dot{\psi}_b + (-1)^i \cdot 2L_b K_y \psi_b = m_w \left( \frac{V_0^2}{R_{ci}} - g \theta_i \right) \quad (3)$$

$$I_{wz} \ddot{\psi}_{wi} + \frac{2f_{11} L_l^2}{V_0} \dot{\psi}_{wi} + \frac{2f_{11} \lambda L_l}{r_0} y_{wi} + 2L_a^2 C_x (\dot{\psi}_{wi} - \dot{\psi}_b) + 2L_a^2 K_x (\psi_{wi} - \psi_b) - K_\psi \psi_{wi} - \frac{2r_0 f_{11} L_l}{V_0} \dot{\beta}_{wi} = \frac{2f_{11} L_l^2}{R_{ci}} + \frac{2f_{11} \lambda L_l}{r_0} y_{ti} \quad (4)$$

$$I_{wy} \ddot{\beta}_{wi} + \frac{r_0 f_{11}}{V_0} \dot{\beta}_{wi} + f_{11} \lambda y_{wi} + \frac{r_0 f_{11} L_l^2}{V_0} \dot{\psi}_{wi} = \frac{r_0 f_{11} L_l}{R_{ci}} + f_{11} \lambda y_{ti} + T_{wi} \quad (5)$$

where  $L_l$  is half of the lateral distance between the rolling circles and  $m_w$  is the mass of the wheelset.  $I_{wz}$  and  $I_{wy}$  are the wheelset yaw and pitch moments of inertia, respectively.  $V_0$  is the longitudinal running speed of the vehicle,  $\lambda$  denotes the wheel conicity of the IRWs, and  $r_0$  is the nominal rolling radius of each IRW.  $K_g$  and  $K_\psi$  are the gravity stiffness and angular stiffness of the wheelset, respectively.  $f_{11}$  and  $f_{22}$  are the creep coefficients in the longitudinal and lateral directions, respectively.  $y_{ti}$  is the irregularity of the track.  $\beta_{wi}$  represents the speed difference of each wheelset, and  $T_{wi}$  is half of the steering torque difference between the wheels, as defined in Equations (6) and (7).

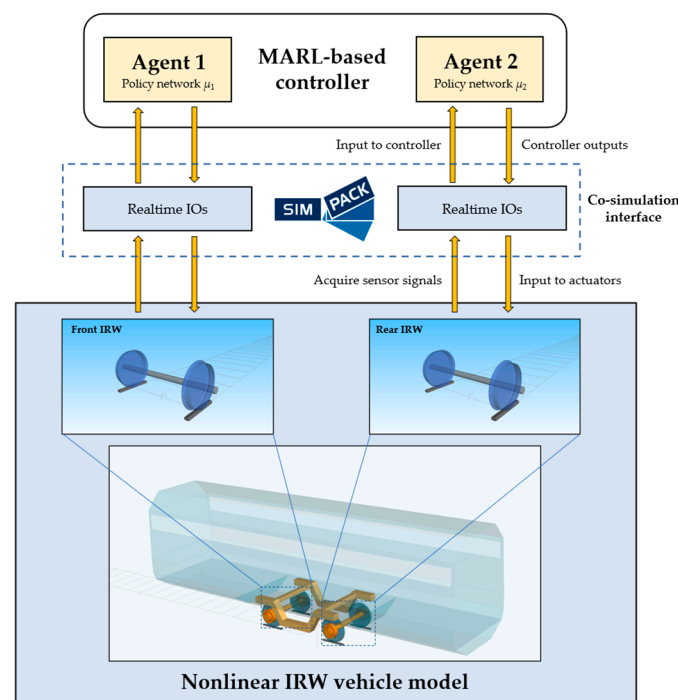
$$\dot{\beta}_{wi} = \frac{\dot{\beta}_{wLi} - \dot{\beta}_{wRi}}{2} \quad (6)$$

$$T_{wi} = \frac{T_{wLi} - T_{wRi}}{2} \quad (7)$$

where the subscripts  $L$  and  $R$  represent the speeds of the left and right wheels, respectively, or the differential torque control of the IRWs.

## 2.2. Closed-Loop Controller for IRWs Based on the MARL Algorithm

The nonlinear model of this IRW vehicle is built in the multibody simulation (MBS) software SIMPACK 2021, and the closed-loop controllers are shown in Figure 2. For the collaborative control system of the vehicle, two IRW controllers are set up for active control of the front and rear wheelsets. The state observation of the vehicle is transferred to the MARL-based controller via SIMPACK real-time input/output interfaces for agent–environment interactions. Based on sensor measurements, including the lateral displacement, yaw angle, and rotational speeds of the wheels, the controller computes the control output using the policy neural network. The torque command applied to the left and right wheels compensates for the longitudinal creep force between the IRW and the track, restoring straight-line stability and facilitating curve negotiation of the IRWs.



**Figure 2.** Closed-loop control via SIMPACK cosimulation.

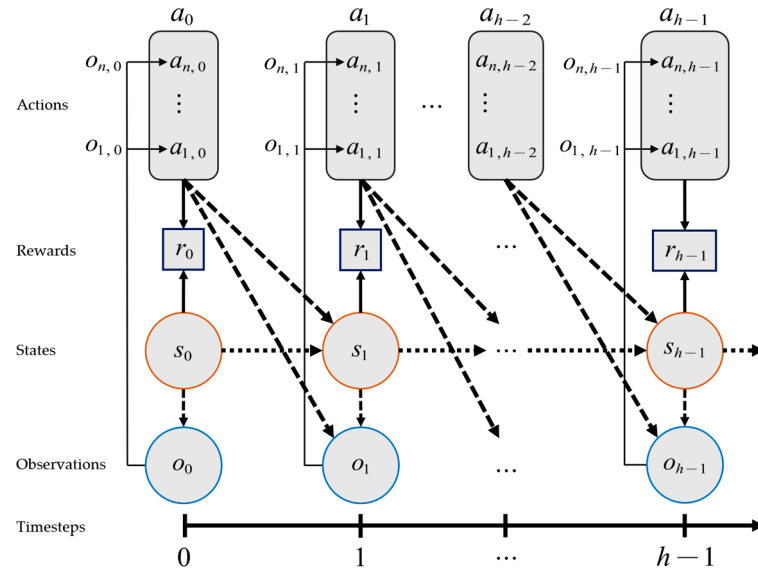
## 3. Active Guidance Controller Design for IRWs Based on MARL

In this section, we explore the theoretical framework of MARL and describe the control objectives using the decentralized partially observable Markov decision process (Dec-POMDP) model. Based on this, we apply the MADDPG algorithm for the active guidance control of IRW vehicles and detail the implementation process of the algorithm.

### 3.1. Cooperative Markov Games

In the context of MARL, the single-agent MDP approach is expanded to cooperative Markov games, where each agent's actions influence their outcomes and those of others, creating a dynamic environment of interdependent decision making. This process can be represented by the  $\{N, S, \{A_i\}, \{O_i\}, T, \{R_i\}, \gamma\}$  tuples, where  $N$  is the number of agents,  $S$  is the state space of the environment,  $A_i$  is the action space of the  $i$ -th agent,  $O_i$  is the observation space of agent  $i$ ,  $T: S \times A_1 \times \dots \times A_N \times S \rightarrow [0, 1]$  is the transition function, defining the probability of the environment changing to the next state after all agents act,  $R_i: S \times A_1 \times \dots \times A_N \rightarrow R$  denotes the reward function of agent  $i$ , providing instant rewards for each agent, and  $\gamma$  is the discount factor used to calculate the present value of future rewards.

Given the localized observation and independent decision-making characteristics of IRW collaborative controllers, we use the Dec-POMDP model for modeling, as shown in Figure 3. Each agent receives a local observation  $o_{i,t} \in O_i$  from the environment at time step  $t$  and selects action  $a_{i,t} \in A_i$  based on the behavior strategy  $\mu_i(\cdot | o_{i,t})$ . After the joint action  $a_t = (a_{1,t}, \dots, a_{N,t}) \in A$  is executed, the rewards  $r_{i,t} \in R_i$  are received by the agents. The environment then transitions to the next state  $S_{t+1}$  determined by the transition function  $T$ .



**Figure 3.** Block diagram of the decentralized partially observable Markov decision process (Dec-POMDP).

In this study, we consider a fully cooperative environment in which all agents work together to achieve a common goal rather than only promoting individual rewards. The policy parameters of the  $i$ -th agent are denoted as  $\theta_i$ , with the policy set of all agents as  $\mu = \{\mu_1, \dots, \mu_N\}$  and the policy parameter set as  $\theta = \{\theta_1, \dots, \theta_N\}$ . The target function for optimal control is shown in Equation (8). Each agent updates its behavior strategy to maximize its cumulative discounted reward, and the optimization target for each agent is expressed by Equation (9).

$$J(\theta) = E_{\tau \sim \mu} \left[ \sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^N R_i(s_t, a_t) \right] \quad (8)$$

$$J_i(\theta_i) = E_{s \sim T, a_i \sim \mu_i(\cdot | o_i), a_{-i} \sim \mu_{-i}(\cdot | o_{-i})} \left[ \sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_{i,t}, a_{-i,t}) \right] \quad (9)$$

where  $\tau = (o_{1,0}, a_{1,0}, \dots, o_{N,0}, a_{N,0}, o_{1,1}, a_{1,1}, \dots)$  is the sequence of observations and actions,  $\mu_i$  is the policy of agent  $i$ ,  $\mu_{-i}$  is the policy set of all agents except agent  $i$ ,  $o_{-i}$  is the collection of local observations of all other agents, and  $a_{-i}$  is the action set of all other agents.

### 3.2. MADDPG Algorithm

The MADDPG conforming to the Dec-POMDP is an extension of the DDPG algorithm and comprises critic and actor networks for each agent [15,17]. By operating as an off-policy algorithm, MADDPG employs a replay buffer, which is denoted  $D$ , archiving all agents' historical interactions with the environment, including global states, actions, subsequent states, and rewards, formatted as  $(s_t, a_{1,t}, \dots, a_{N,t}, s_{t+1}, r_{1,t}, \dots, r_{N,t})$  tuples. To optimize the critic network  $Q_i(s_t, a_{1,t}, \dots, a_{N,t})$  and its parameters  $\omega_i$ , the loss function  $L(\omega_i)$  is defined in Equation (10).

$$L(\omega_i) = \frac{1}{|D_s|} \sum_{j \in D_s} (y_i^j - Q_i(s^j, a_1^j, \dots, a_N^j))^2 \quad (10)$$

where  $D_s$  is a small batch from the replay buffer,  $j$  is the index of the samples, and  $y_i^j$  is the TD target, computed as per Equation (11).

$$y_i^j = r_i^j + \gamma Q'_i(s^{j+1}, a_1^{j+1}, \dots, a_N^{j+1}) \Big|_{a_k^{j+1} = \mu'_k(o_k^{j+1})} \quad (11)$$

where  $Q'_i$  and  $\mu'_i$  are the critic and actor networks defined in the target network, respectively.

The actor network's update considers the gradient of  $Q_i$  with respect to  $a_i$ , and the policy gradient for agent  $i$  is shown in Equation (12).

$$\nabla_{\theta_i} J(\theta_i) \approx \frac{1}{|D_s|} \sum_{j \in D_s} \nabla_{\theta_i} \mu_i(o_i^j) \nabla_{a_i} Q_i(s^j, a_1^j, \dots, a_N^j) \quad (12)$$

where  $a_k^j = \mu_k(o_k^j)$ ,  $k = 1, \dots, N$ , and  $Q_i(\cdot)$  is agent  $i$ 's action value function under policy  $\mu_i$ . The gradients  $\nabla_{\theta_i}(\cdot)$  and  $\nabla_{a_i}(\cdot)$  correspond to the policy parameter gradient and the action gradient, respectively.

### 3.3. Improved MADDPG with PER

The standard experience replay mechanism randomly resamples the stored experiences without considering sample quality. In our case, the quality of the stored data is influenced by variations in wheel–rail contact and system noise in IRWs under different control strategies. For example, during the early training stages, experiences restoring IRWs from large yaw angles or lateral displacements are more informative. These experiences, which could have been utilized more efficiently to improve the controller, have fewer chances of being chosen because of uniform random sampling.

To address this issue, we introduce the PER mechanism, a method widely used in single-agent off-policy reinforcement learning [21]. PER assigns priorities to experience samples based on the absolute value of the TD error, enhancing the probability of selecting high-value experiences for training. This prioritization ensures a more frequent selection of samples poorly predicted under the current policy.

In addition, we use a centralized replay buffer based on a SumTree structure to store experience and its TD-based prioritization. SumTree is a complete binary tree in which each leaf node contains an individual sample state transition tuple and its corresponding priority value. Each nonleaf node holds the cumulative sum of the priorities of its child nodes. When new experiences are added or the priorities of existing experiences are updated, only the specific path from the relevant leaf node to the root requires adjustment rather than continuously sorting samples according to priority in the experience buffer.

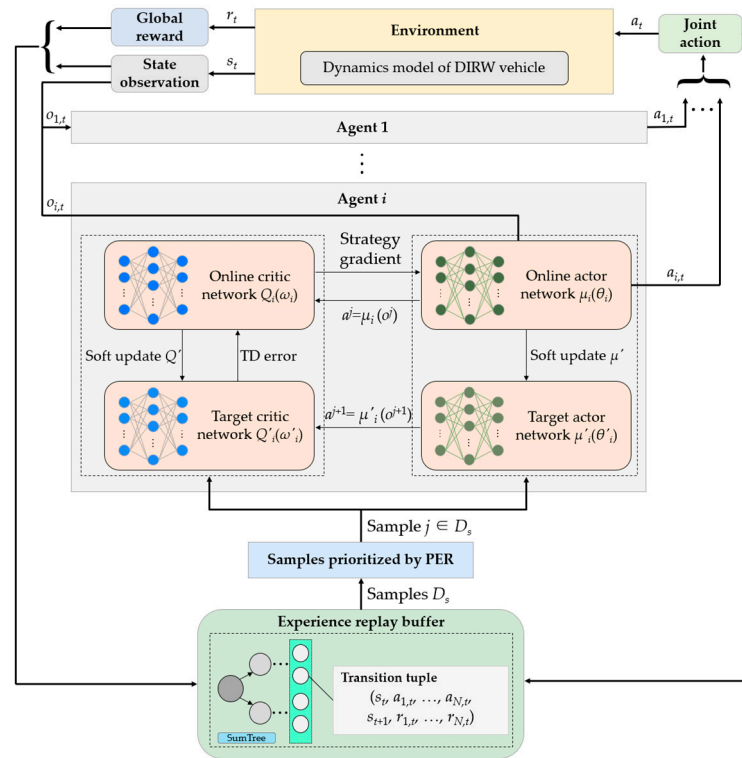
An illustrative representation of the PER-MADDPG algorithm framework is shown in Figure 4.

The TD error calculated by the critic network determines the deviation between the estimated and target  $Q$  values derived from samples. For a given experience tuple  $j$ , agent  $i$  calculates the current  $Q$  value dependent on the  $s^j$  and the collective actions, while the target critic network determines the  $Q$  value for  $s^{j+1}$ . The TD error can be formulated using Equation (13).

$$\delta_i^j = r_i^j + \gamma Q'_i(s^{j+1}, a_1^{j+1}, \dots, a_N^{j+1}) - Q_i(s^j, a_1^j, \dots, a_N^j) \quad (13)$$

where  $\delta_i^j$  denotes the TD error of experience sample  $j$  used by agent  $i$  and  $Q'_i(s^{j+1}, a_1^{j+1}, \dots, a_N^{j+1})$  represents the  $Q$  value predicted by the target-critic network for the subsequent state  $s^{j+1}$ .





**Figure 4.** PER-MADDPG algorithm framework.

When employing the PER method, the priority  $p_j$  of each experience based on the TD error is defined as  $p_j = |\delta^j| + \epsilon$ . Here,  $\epsilon$  is a tiny positive constant that ensures a nonzero probability of selection for all experiences. The sampling probability  $P_j$  for the  $j$ -th experience is calculated based on the priority  $p_j$  relative to the weighted sum of all experience priorities using the formula:  $P_j = (p_j)^\alpha / \sum_{k \in D} (p_k)^\alpha$ , with  $\alpha \in [0, 1]$  balancing random and greedy sampling.

Moreover, we apply importance sampling (IS) to rectify the introduced sampling bias via nonuniform sampling in the PER, and the IS weight  $w_j$  is calculated using the formula  $w_j = (|D| \cdot P_j)^{-\beta}$ , where  $|D|$  is the capacity of the experience replay buffer and  $\beta$  is an adjustment parameter that controls the bias correction, with  $\beta \in [0, 1]$ . The IS weights should also be normalized to ensure that no sample weight dominates the update step  $w_j \leftarrow w_j / \max_{k \in D} w_k$ , where  $k$  is the index of the sample with the maximum weight and  $\max w_k$  is the highest weight among all the experience data.

The training phase of the PER-MADDPG is shown in Algorithm 1. In the centralized training of the MADDPG framework, each agent trains value and policy networks, in which the value network of each agent can access information from all other agents when evaluating the potential benefits of actions. Once the training is complete, the policy network alone is sufficient to guide the independent execution of each agent, and the value network is no longer needed.

**Algorithm 1:** Training pseudocode of PER-MADDPG algorithm with  $N$  agents

**Input:** Maximum time steps  $N_{steps}$ ; maximum learning episodes  $E_m$ ; discount factor  $\gamma$ ; priority parameters  $\alpha, \beta$  for PER sampling; soft update rate  $\xi$ ; experience replay buffer  $D$ ; mini-batch size  $|D_s|$ .

**Output:** Trained actor networks  $\{\mu_1, \dots, \mu_N\}$  and critic networks  $\{Q_1, \dots, Q_N\}$ .

```

01:   Initialize the online actor networks  $\{\mu_1(\theta_1), \dots, \mu_N(\theta_N)\}$  with random parameters  $\{\theta_1, \dots, \theta_N\}$ .
02:   Initialize the online critic networks  $\{Q_1(\omega_1), \dots, Q_N(\omega_N)\}$  with random parameters  $\{\omega_1, \dots, \omega_N\}$ .
03:   Initialize the weights of target actor and critic networks with  $\omega'_i \leftarrow \omega_i$  and  $\theta'_i \leftarrow \theta_i$  ( $i = 1, \dots, N$ ).
04:   Initialize the SumTree structure for the experience replay buffer.
05:   for episode = 1 to  $E_m$  do
06:       Initialize the Ornstein–Uhlenbeck noise process  $N_{OU}$ .
07:       Initialize the environment and obtain initial state observation  $s = (o_{1,0}, \dots, o_{N,0})$ .
08:       for each time step do
09:           for each agent  $i = 1$  to  $N$  do
10:               Select action  $a_{i,t} = \mu_{i,t}(o_{i,t}; \theta_i) + N_{OU}(t)$  for exploration.
11:           end for
12:           Execute  $a_t = \{a_{1,t}, \dots, a_{N,t}\}$ .
13:           Observe the individual reward  $r_t = \{r_{1,t}, \dots, r_{N,t}\}$ , new state  $s_{t+1} = \{o_{1,t+1}, \dots, o_{N,t+1}\}$  and done flag.
14:           Store  $(s_t, a_t, s_{t+1}, r_t)$  with initial high priority in  $D$  using SumTree.
15:           Update state:  $s_t \leftarrow s_{t+1}$ .
16:           for each agent  $i = 1$  to  $N$  do
17:               Sample a mini-batch of  $|D_s|$  from  $D$  using PER.
18:               for each sampled tuple do
19:                   Compute the TD error:  $\delta_i^j = r_i^j + \gamma Q'_i(s^{j+1}, \mu'_i(s^{j+1}; \theta'_i); \omega'_i) - Q_i(s^j, a^j; \omega_i)$ .
20:                   Update the sample priority  $p_j$  in SumTree based on  $|\delta_i^j|$ .
21:                   Compute the sampling probability  $P_j$  using priority  $p_j$ .
22:                   Compute the normalized IS weight  $w_j$  based on  $P_j$ .
23:               end for
24:               Compute the TD target  $y_j$  for each sample using TD error.
25:               Update  $Q_i$  by minimizing the loss function via  $w_j$  weighting:

$$L(\omega_i) = \frac{1}{|D_s|} \sum_{j \in D_s} w_j \cdot (y_i^j - Q_i(s^j, a^j; \omega_i))^2.$$

26:               Update  $\mu_i$  by optimizing with the  $w_j$ -weighted policy gradient:

$$\nabla_{\theta_i} J(\theta_i) \approx \frac{1}{|D_s|} \sum_{j \in D_s} w_j \cdot \nabla_{\theta_i} \mu_i(o_i^j; \theta_i) \nabla_{a_i = \mu_i(o_i^j)} Q_i(s^j, a^j; \omega_i).$$

27:           end for
28:           Update  $\omega'_i$  and  $\theta'_i$  for each agent  $i$  using a soft update:

$$\omega'_i \leftarrow \xi \cdot \omega_i + (1 - \xi) \cdot \omega'_i$$
 for target critic network;  $\theta'_i \leftarrow \xi \cdot \theta_i + (1 - \xi) \cdot \theta'_i$  for target actor network.
29:           if done flag is true, then
30:               break
31:           end if
32:       end for
33:   end for

```

### 3.4. MARL-Based Controller Design

In the prototype vehicle shown in Figure 2, separate controllers are deployed for the IRWs located at the front and rear. As a two-axle vehicle travels on a track, both agents acquire dynamic state information about their respective IRWs through sensors. The controller's actions involve half of the torque differential exerted on the wheels on the same axle. Action outputs are augmented with Ornstein–Uhlenbeck (OU) noise to encourage the exploration of policies. The OU noise process, characterized by temporal correlation, mitigates the risk of extreme or sudden action changes and maintains a balance between exploration and exploitation [22]. The state feedback from the IRWs determines the expected reward, which the MARL algorithm attempts to maximize. The design of the MARL controller involves specifying critical elements such as the observation space, action space, reward function, and actor and critic networks, which are defined in this subsection.



### 3.4.1. Local and Global Observation Space

Each IRW observes only its local state without considering the dynamic states of the entire vehicle or other IRWs, which reduces the observation dimensionality of each agent, achieving local controllers and reducing communication costs. The observation state vector, based on the active guidance control objectives and vehicle dynamics model shown in Equations (1)–(7), is defined in Equation (14).

$$O_i = \begin{bmatrix} y_{wi} & \psi_{wi} & \dot{y}_{wi} & \dot{\psi}_{wi} & \omega_{Li} & \omega_{Ri} & \ddot{\beta}_{wi} \end{bmatrix} \quad (14)$$

where the subscript  $i = 1$  denotes the dynamic variables of the front IRW and  $i = 2$  denotes those of the rear IRW.  $\omega_{Li}$  and  $\omega_{Ri}$  represent the angular velocities of the left and right wheels of each IRW, respectively.

When using the PER-MADDPG algorithm, the local observation for each agent includes the lateral displacement and attack angle of the wheelset with respect to the centerline of the track, which are the primary control objectives. Since the rotational speeds of each IRW can derive both the speed difference between the wheels and the longitudinal running speed, they are also considered observation states. Additionally, the time derivative of the difference in speed between wheels is part of the state vector. Therefore, in the centralized training process, the global state is  $S = [O_1 \ O_2]$ .

### 3.4.2. Action Space Definition

During the operation of the IRW vehicle with two independently rotating wheelsets, each agent's output equals half of the torque differential exerted on the left and right IRWs. The input torque of each IRW is subjected to a torque of magnitude  $|T_{wi}|$ , and the input torque directions are opposite for the left and right IRWs on the same axle. Each agent's action is a one-dimensional vector, and the resulting joint action space is demonstrated in Equation (15).

$$A = \begin{bmatrix} T_{w1} & T_{w2} \end{bmatrix} \\ \text{s.t. } -T_{\max} \leq T_{wi} \leq T_{\max} (i = 1, 2) \quad (15)$$

where  $T_{\max}$  is the peak output of the controller, with the control torque to each wheel restricted between  $-T_{\max}$  and  $T_{\max}$ .

### 3.4.3. Reward Shaping

The design of the reward function is essential for guiding agents to optimize toward the control objective efficiently. For an IRW vehicle, the MARL-based controller aims to improve straight-line stability and curve-negotiation capabilities, which can be achieved by implementing control methodologies, including zero lateral clearance [23] and radial control [24]. Specifically, zero lateral clearance aims to maintain the geometric center alignment of the IRW with the track centerline, minimizing lateral displacement. Radial control ensures that each wheelset achieves a radial position in a curved line with a minimized attack angle. Thus, the reward function should reward agents more when wheelsets exhibit less lateral displacement and yaw angles. Moreover, considering the impact of actuator output on the energy efficiency of guidance control, the reward decreases as the control torque increases, promoting environmentally friendly control actions.

Based on the above discussion, the definition of the reward function is constructed using the weighted sum of the lateral displacement, yaw angle, and motor torque of the wheelsets for the IRW vehicles. Since the control torque and wheelset dynamics state are not of the same order of magnitude, normalization is needed between the parameters. In the fully cooperative multiagent environment of our proposed MARL-based controller, all agents share a global reward function  $R$ , and  $R$  is formulated as (16).

$$R = \kappa^2 \sum_{i=1}^{N=2} \left[ \eta_{i1} (y_{\max} - |y_{wi}|)^2 + \eta_{i2} (\psi_{\max} - |\psi_{wi}|)^2 \right] + \sum_{i=1}^{N=2} \eta_{i3} (T_{\max} - |T_{wi}|)^2 \quad (16)$$

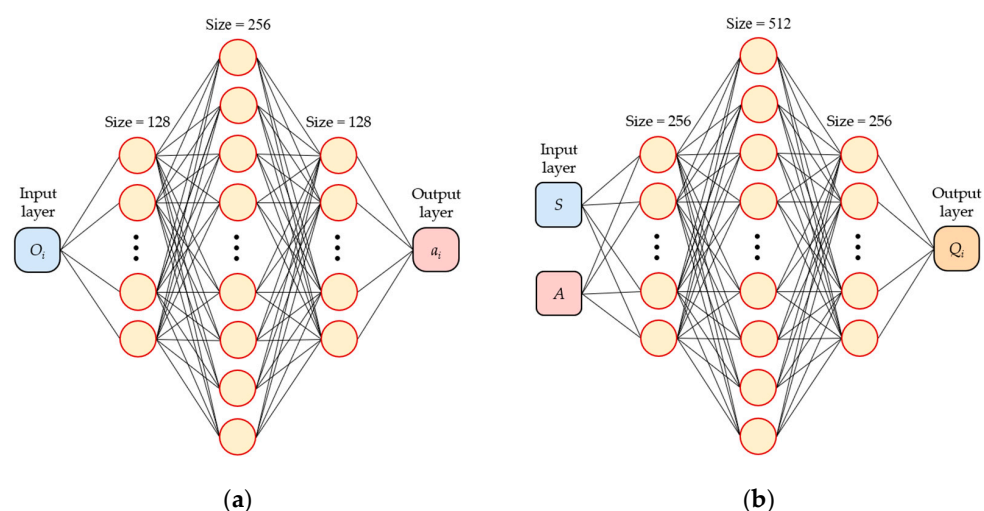
where  $y_{\max}$  and  $\psi_{\max}$  are the set upper bounds for the lateral position and attack angle of the wheelsets, respectively. The lateral displacement reward coefficients are denoted by  $\eta_{11}$  and  $\eta_{21}$ , those for the yaw angle by  $\eta_{12}$  and  $\eta_{22}$ , and those for the control output by  $\eta_{13}$  and  $\eta_{23}$ .  $\kappa$  denotes the conversion factor from meters to millimeters, and  $\kappa = 1000$ .  $T_{\max}$  denotes the maximum torque output for the actuators.

The first term of  $R$  represents the reward associated with the lateral displacement and attack angle, aiming to ensure that under active guidance control, the IRW vehicle maintains optimal steering performance while avoiding wheel flange contact on various tracks, which implies that the lateral movement and yaw angle of the wheelsets should remain within the predefined limits. The second term of the reward function concerning the control torque indicates that the controller should constrain the output torque of the motors without compromising the guidance performance. Furthermore, the front wheelset, which typically encounters larger yaw angles during curve negotiation, has higher reward weights than does the rear wheelset. Therefore, this approach prioritizes the improvement of the dynamic performance of the front wheelset, taking into account its more severe wheel–rail contact conditions in curved lines.

#### 3.4.4. Policy and Value Networks

Each agent in our MADDPG algorithm is equipped with two different neural networks: a policy network and a value network. The policy network is responsible for mapping the dynamically observed state to deterministic actions, while the value network assesses the quality of these actions given the current policy. As Equation (14) defines, the actor (policy) network's input layer corresponds to each agent's local observation vector, comprising seven neurons to match the dimensionality of the observation. Conversely, the input layer of the critic (value) network integrates the global state with all collective actions, resulting in an input dimension of 16.

The neural network architectures consist of three fully connected internal layers for both the policy and value networks, achieving a balance between capturing the dynamic characteristics of vehicles and computational demands, as detailed in Figure 5. The policy network features 128, 256, and 128 neurons across its three hidden layers and is designed to extract complex dynamic features for generating torque commands. The value network, with hidden layers of 256, 512, and 256 neurons, evaluates the state–action pairs. For both networks, the rectified linear unit (ReLU) activation function is utilized in the hidden layer to introduce nonlinearity and alleviate vanishing gradients. Moreover, the output layers employ the tanh function for normalization. Training is conducted using the adaptive moment estimation (Adam) optimizer, with learning rates of  $10^{-3}$  for the value network and  $10^{-4}$  for the policy network.



**Figure 5.** Definitions of the policy and value networks of the MADDPG. (a) Policy network, (b) Value network.

## 4. Training and Simulation

### 4.1. Training Process of the MARL-Based Controller

The learning process of the collaborative control strategy for IRW vehicles is detailed in Figure 6. The random selections of the operating conditions of the vehicle include different speeds, curve radii, and cant, in addition to lateral and vertical random excitations based on AAR5 track spectra, as shown in Table 1. In each episode, agents utilize virtual sensors installed on the wheelsets to measure the lateral displacement, yaw angle, and speed differential of the IRWs, thereby obtaining the necessary local observations for control decisions. The front and rear IRW active controllers then execute actions based on their current policies and these observed states, managing the torque differential applied to the IRWs. The dynamic response of the vehicle, as defined by the model described in Section 2, is computed along with the corresponding reward functions. The environment then transitions to the next state, during which the experience tuples are stored in the PER buffer.

**Table 1.** Simulation conditions.

Track No.	Type	Speed [km/h]	Curve Radius [m]	Cant [mm]	Track Irregularities
1	Straight track	120	—	—	AAR5
2	Straight track	200	—	—	AAR5
3	Curved track	100	600	80	AAR5
4	Curved track	150	1500	60	AAR5

During the centralized training phase, each agent updates its network parameters: the critic network parameters are refined by minimizing the loss function, reflecting the action-value function's estimation error, while the actor network parameters are optimized using the policy gradient method to maximize the expected reward. The training process involves continuous interaction with the environment, with networks being updated according to the PER-MADDPG mechanism until a terminal state is encountered, signaling the end of the current episode and the initialization of a new episode.

The reward function quantifies the steering capability of the IRWs, where higher rewards indicate more effective active steering control. Early termination of agent–environment interactions occurs if the lateral displacement or yaw angle exceeds the predefined thresholds in Table 2, prompting an environmental reset and agent penalization. This mechanism can guide agents toward more effective control strategies. If controlled IRWs do not exceed the state threshold or wheel flange contact occurs within the maximum time step  $N_{steps}$ , the centering and curve guidance abilities of the IRWs are successfully restored, thereby validating the effectiveness of the MARL controller.

**Table 2.** MARL training parameters.

Parameter	Value	Description
$\alpha$	0.5	Balance coefficient for prioritized sampling in PER
$\beta$	0.5	Bias correction coefficient for importance sampling
$\gamma$	0.99	Discount factor for future rewards
$\zeta$	0.005	Soft update coefficient for target networks
$\epsilon$	$10^{-4}$	Small positive constant ensuring nonzero sampling probability
$E_m$	$5 \times 10^5$	Total number of training episodes
$ D $	$1 \times 10^6$	Capacity of the experience replay buffer
$ D_s $	512	Default batch size for gradient descent
$N$	2	Number of agents, including front and rear wheelsets

Table 2. Cont.

Parameter	Value	Description
$N_{steps}$	2000	Maximum time steps per episode
$y_{max}$	10 mm	Threshold for lateral displacement termination
$\psi_{max}$	10 mrad	Threshold for yaw angle termination
$T_{max}$	2000 N·m	Maximum torque output for actuators
$\eta_{11}$	2	Front IRW lateral displacement reward weight
$\eta_{12}$	1	Front IRW yaw angle reward weight
$\eta_{21}$	0.5	Rear IRW lateral displacement reward weight
$\eta_{22}$	0.2	Rear IRW yaw angle reward weight
$\eta_{13}$	0.05	Front IRW torque control reward weight
$\eta_{23}$	0.02	Rear IRW torque control reward weight

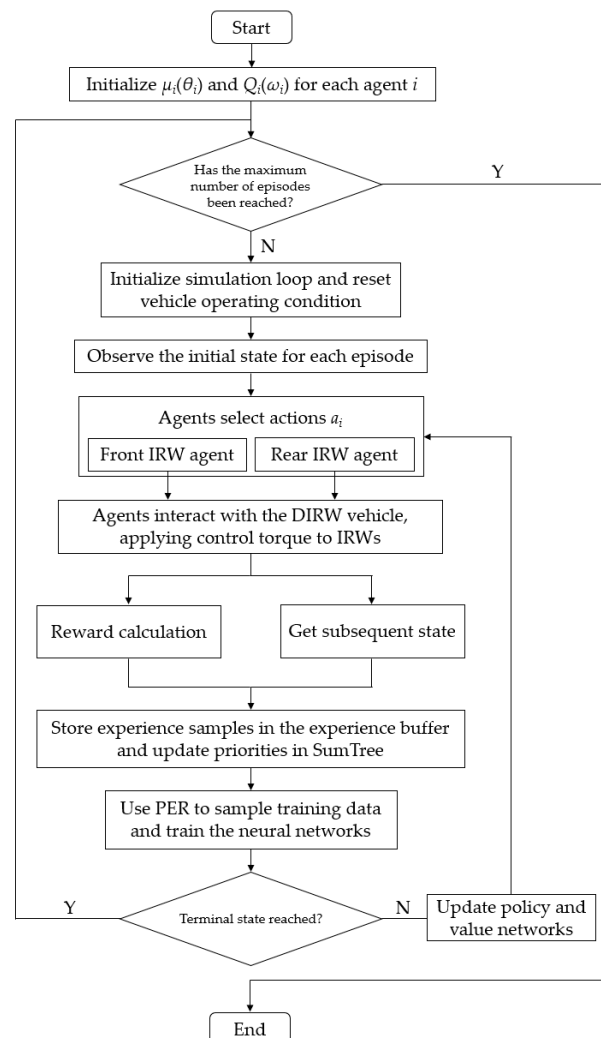


Figure 6. Flowchart of the training process.

The PER-MADDPG algorithm was implemented using Python 3.8 and the PyTorch 1.12 framework. We conducted the training on a workstation powered by an Intel Gold 6132 CPU and equipped with a GPU accelerator. The control frequency for the agents was set at 100 Hz, and each episode was set to last a maximum duration of 20 s, with a time step of 0.01 s. The size of the experience buffer was set at  $1 \times 10^6$ , utilizing the PER

scheme, where experiences are retained or discarded based on their TD priorities. The hyperparameters for the MARL algorithm training are detailed in Table 2.

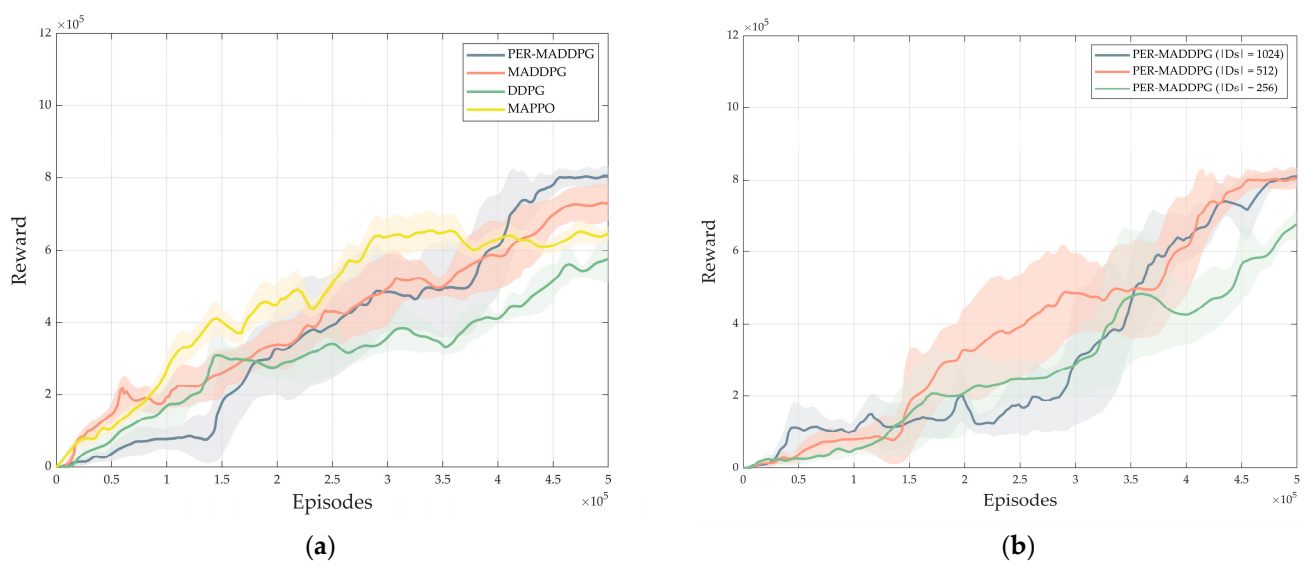
## 4.2. Training Results

### 4.2.1. Reward Comparison

Our study comprehensively evaluated the PER-MADDPG algorithm against existing MARL algorithms to verify its performance in active guidance control for IRW vehicles. The selected algorithms for reward analysis include the basic MADDPG and single-agent (SA)-based DDPG algorithms. We also introduce the multiagent proximal policy optimization (MAPPO), which is based on on-policy learning strategies, into the comparison. MAPPO, an extension of the proximal policy optimization (PPO) algorithm, employs a centralized value function method to optimize each agent's strategy; this method performs well in many cooperative Markov games and is often used as a benchmark for evaluating MARL performance [25,26]. Like the MADDPG algorithm, MAPPO uses a combination of value function and policy gradient methods based on the CTDE framework for implementing collaborative controllers.

To ensure the robustness of the evaluation, we used five random seeds to train the MARL agents. As the number of time steps in the training process increases, when the reward per episode becomes stable, the training of the MARL-based controller converges. We ensured uniformity in the main parameter settings across all algorithms for a fair comparison.

Table 3 and Figure 7a display the differences in the performance of the reward values among the different algorithms. The training results highlight the superior performance of the PER-enhanced MADDPG algorithm, which achieved the highest reward values and demonstrated convergence after approximately  $4.5 \times 10^5$  training episodes. In contrast, the basic MADDPG algorithm lags in convergence speed and final reward achievement due to its inefficient use of high-value samples. The DDPG controller, limited by its high-dimensional observation requirements, failed to converge within the predetermined maximum number of episodes, with reward values significantly lower than those of both MADDPG and PER-MADDPG. Our analysis also indicates that when applying MARL algorithms to IRW guidance control, on-policy DRL algorithms lag in sample efficiency compared to off-policy algorithms. The MAPPO algorithm showed a decrease in performance when frequently reusing low-value samples, potentially leading to their final reward values falling into local optima.



**Figure 7.** Reward comparison during training episodes. (a) Comparison of different DRL and MARL algorithms, (b) Comparison with different mini-batch sizes. The solid line shows the average return from five random initializations, and the shaded area indicates the standard deviation.

**Table 3.** Comprehensive evaluation of the trained MARL and DRL algorithms.

Algorithm	Final Mean Reward	Standard Deviation of Final Reward	Training Episodes	Sampling Policy	Training Framework
PER-MADDPG	$8.05 \times 10^5$	$2.96 \times 10^4$	$5 \times 10^5$	Off-policy with PER	CTDE
MADDPG	$7.31 \times 10^5$	$4.92 \times 10^4$	$5 \times 10^5$	Off-policy	CTDE
MAPPO	$6.48 \times 10^5$	$2.02 \times 10^4$	$5 \times 10^5$	On-policy	CTDE
DDPG	$5.76 \times 10^5$	$6.64 \times 10^4$	$5 \times 10^5$	Off-policy	SA

The above comparison results demonstrate the performance of the PER-MADDPG algorithm, which can obtain higher reward values and exhibit faster convergence, demonstrating its high sampling efficiency.

#### 4.2.2. Different Mini-Batch Sizes

The choice of mini-batch size in the MADDPG algorithm plays a critical role in balancing exploration capabilities with training stability, influencing both the convergence speed and reward values of the algorithm. Figure 7b shows the convergence performance of the PER-MADDPG algorithm under various mini-batch sizes, specifically for configurations of 256, 512, and 1024. We observed smoother gradient updates and more robust convergence results with greater mini-batch sizes, particularly 512 and 1024. In contrast, a mini-batch size of 256 was associated with less accurate gradient estimation, resulting in diminished reward values and a slow convergence speed. Considering the balance between computational resource demands and algorithmic efficiency, we adopted a mini-batch size of 512 for subsequent analyses of the PER-MADDPG algorithm.

#### 4.3. Comparative Analysis of the Control Effects in the MBS Simulations

To validate the effectiveness of the proposed control strategy, a cosimulation involving the integration of SIMPACK and Python was conducted on the prototype vehicle. The  $H_\infty$  control algorithm and the PER-MADDPG controller were tested for use in evaluating the dynamic response of the IRWs. The  $H_\infty$  algorithm involved in the comparison is a model-based state feedback controller based on the  $\mu$ -synthesis theory. This algorithm designs the controller  $K_{Hinf}$  to minimize the robust  $H_\infty$  performance established on IRW dynamic equations that include structural uncertainties, particularly at the wheel–rail contact interface. The  $H_\infty$  controller ensures that the closed-loop system remains quadratically stable for all permissible uncertainties, thus ensuring IRW system stability. Due to the varying bounds of external disturbances in different tracks, the robust  $H_\infty$  algorithm requires particular optimization for each condition. In contrast, the proposed algorithm does not require parameter adjustments for specific operating conditions.

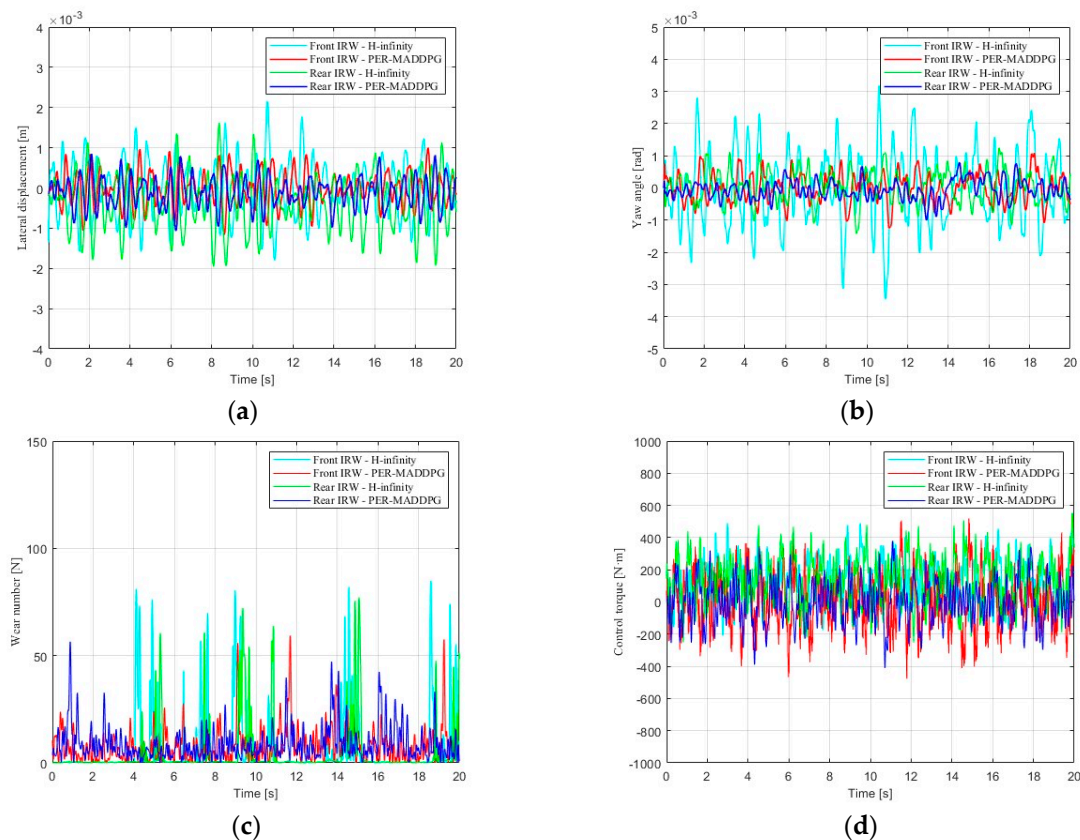
We focused on two specific operating conditions for our analysis: the straight-line condition at 120 km/h and the 1500 m radius curve at 150 km/h, as detailed in cases 1 and 4 of Table 2.

##### 4.3.1. Running Stability on a Straight Track

Figure 8 presents the simulation results for the active guidance performed in the straight line at 120 km/h. Under the control of the  $H_\infty$  algorithm, the maximum lateral displacements observed in the front and rear wheelsets were 2.21 mm and 1.93 mm, respectively. In contrast, the PER-MADDPG controller demonstrated superior performance, reducing the maximum lateral displacement to 1.15 mm for the front wheelsets and 1.05 mm for the rear wheelsets. This considerable improvement signifies enhanced straight-line centering capabilities. Similarly, marked reductions were observed in the yaw angles; the maximum yaw angles of the front and rear wheelsets decreased from 3.47 mrad and 1.45 mrad, respectively, under  $H_\infty$  control to 1.23 mrad and 0.95 mrad, respectively, when controlled by PER-MADDPG. This reduction in both lateral displacement and yaw angle



also led to a noticeable decrease in wheel–rail wear numbers. The maximum wear numbers for the front and rear wheelsets decreased from 83.93 N and 77.21 N under  $H_\infty$  control to 59.44 N and 56.16 N under PER-MADDPG control, with the average values decreasing from 15.8 N and 10.9 N to 7.7 N and 5.9 N, respectively. Thus, the proposed active control algorithm effectively reduced wheel–rail wear by more than 40%. Furthermore, the PER-MADDPG controller maintained maximum control torques of 467.1 N·m and 408.2 N·m for the front and rear wheelsets, respectively, and the average control torques were 183.9 N·m and 164.1 N·m, respectively, well within the controller's design capacity of 2 kN·m. The comparative results of the cosimulation show that the PER-MADDPG controller significantly improves the straight-line stability of IRWs relative to the model-based controller, as evidenced by the reductions in lateral displacement and attack angle, as well as in wheel–rail wear, thus validating the performance of the proposed algorithm.



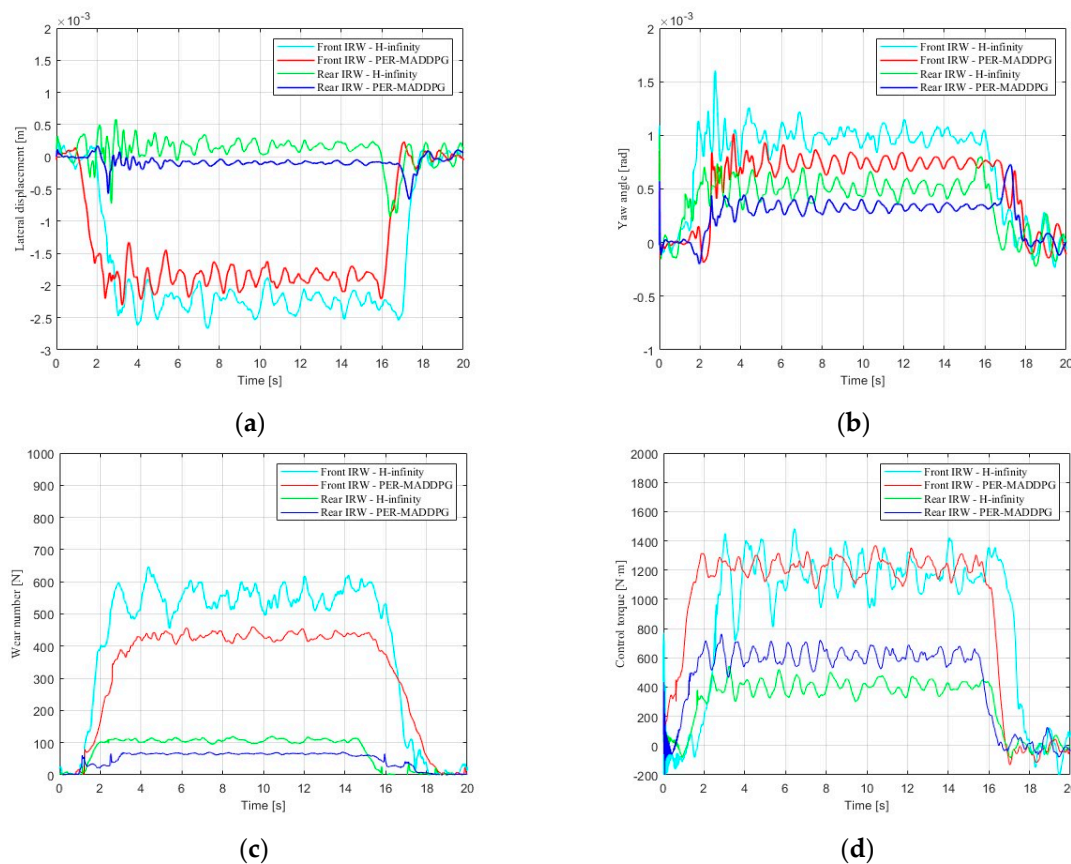
**Figure 8.** Straight line cosimulation results ( $V_0 = 120$  km/h). (a) Lateral displacement of the wheelsets, (b) Yaw angle of the wheelsets, (c) Wear number of the wheelsets, (d) Controller output.

#### 4.3.2. Curve-Negotiation Performance

Figure 9 shows the simulation results for the IRW vehicle running on a 1500 m radius curve at 150 km/h. The curved line includes a circular arc track between short, straight lines and transition curves. Compared with the  $H_\infty$  robust controller, the PER-MADDPG controller effectively reduced the maximum lateral displacement of the front IRW from 2.61 mm to 2.39 mm and the maximum lateral displacement of the rear IRW from 0.91 mm to 0.64 mm. The yaw angles of both independently rotating wheelsets also decreased significantly, from 1.62/0.79 mrad to 1.05/0.73 mrad. This reduction in yaw angle, resulting in diminished wheel–rail forces, led to a notable decrease in the maximum wear number of the wheelsets on the curve, from 644.4 N and 113.54 N under  $H_\infty$  control to 458.7 N and 65.4 N under PER-MADDPG control, representing reductions of 28.8% and 42.4%, respectively. The peak torques for both the front and rear wheelsets remained below the

2 kN·m maximum threshold. Additionally, the wheelset dynamic response indicated that upon exiting the curve, the active control swiftly reestablished straight-line centering.

Through this comparative analysis of the running performance of the front and rear IRWs under various straight and curved conditions, it is evident that the PER-MADDPG controller significantly enhances the dynamic performance of the IRWs. This improvement is revealed by better stability on straight tracks, improved curve-negotiation capabilities, and reduced wheel–rail wear, all while ensuring that the controller’s outputs remained within the design limit.

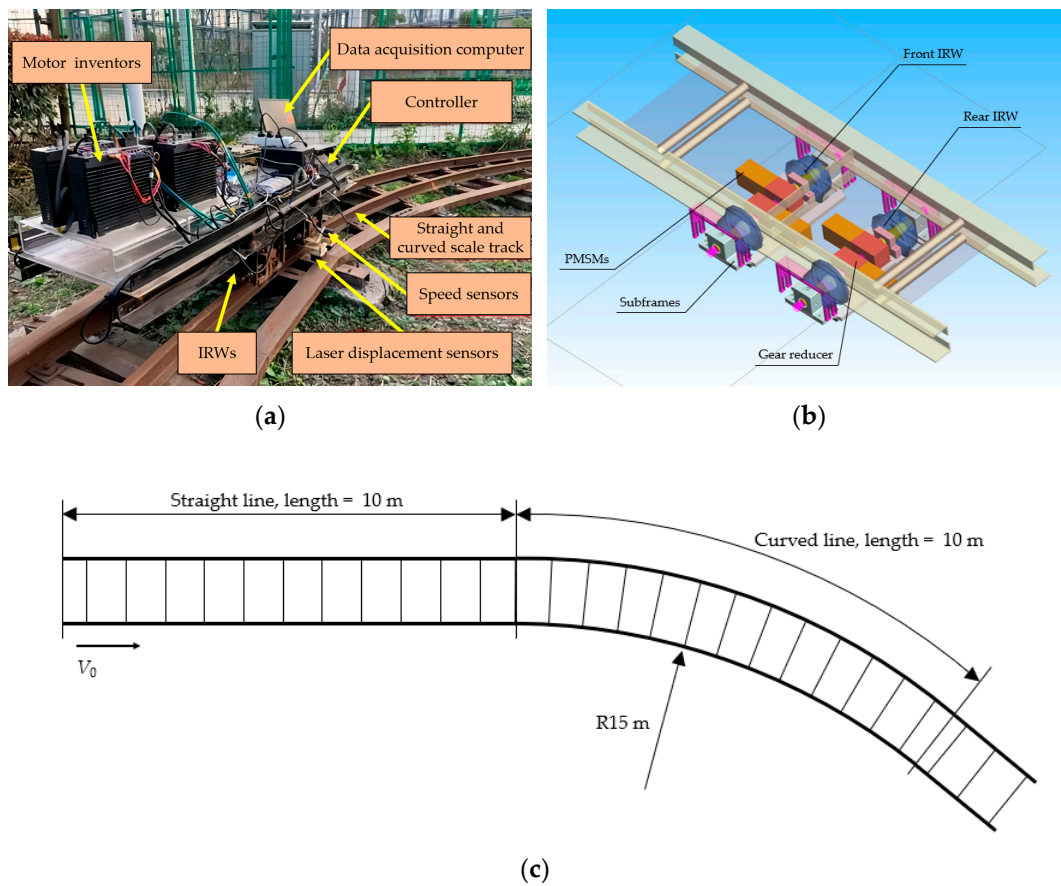


**Figure 9.** Curved line cosimulation results ( $V_0 = 150$  km/h,  $R_c = 1500$  m). (a) Lateral displacement of the wheelsets, (b) Yaw angle of the wheelsets, (c) Wear number of the wheelsets, (d) Controller output.

## 5. Experimental Validation

### 5.1. Scale Model of the IRW Vehicle

Experimental validation is crucial for verifying the efficacy of the active steering control strategy in real-world deployments. We designed a scale model of an IRW vehicle according to Iwnicki’s similarity laws [27] at a scale factor of 1:5, as shown in Figure 10a. The electromechanical structure of the scale vehicle comprises two sets of independently rotating wheelsets and subframes, 600 W PMSMs as drive motors, right-angle gear reducers, and primary and secondary suspension systems. Each IRW, mounted on a subframe through axle boxes, connects to the subframe via primary suspension. The drive motors exert control torque through gear reducers, combining traction and steering. The car body is connected to the subframe through secondary suspension and equipped with onboard devices such as controllers, a data acquisition PC, motor drives, and power inverters.



**Figure 10.** Schematic of the scale model experiment. (a) Scale model vehicle, (b) Digital model for MARL training, (c) Layout of the scale track.

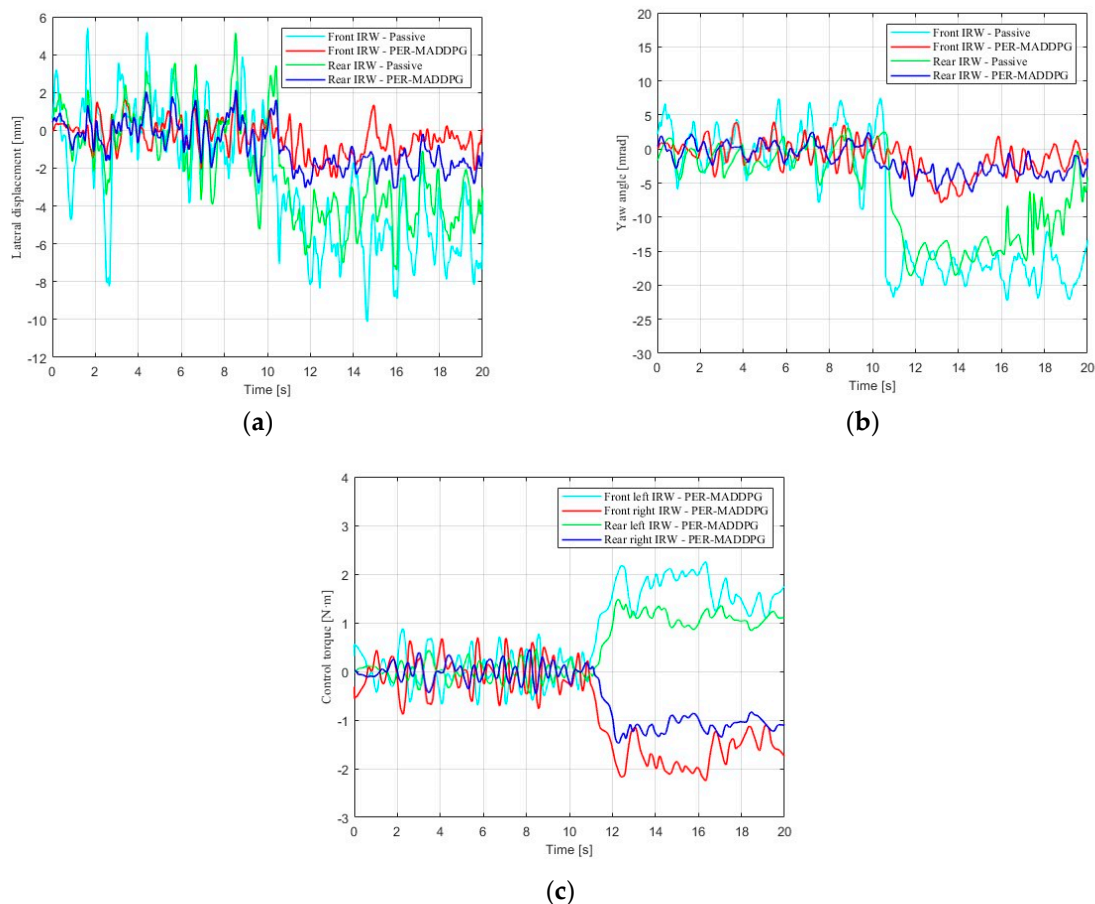
Each subframe of the scale model is equipped with two laser displacement sensors that track the movement of the IRW. The sensors measure the wheelset's lateral displacement and yaw angle, defined as the lateral position and angular deviation of the wheelset's geometric center from the track's centerline, consistent with the approach used in [2]. Wheel rotation is captured by 12-bit incremental encoders. The differentials of displacement, yaw angle, and speed difference are essential for the controller and are determined by applying backward difference calculations to the measured values. This method approximates the derivatives of these variables in discrete-time signals. The MARL algorithm is deployed on an NVIDIA Jetson Nano embedded computer. The policy networks for the two collaborating agents are trained using the PER-MADDPG algorithm within the simulated environment of the scale IRW model, as illustrated in Figure 10b. The inference applications of deep neural network models are optimized using TensorRT to enable real-time control of the GPU during scaled vehicle operation.

## 5.2. Experimental Results

The active steering control was tested on a 1:5 scale test railway track, the layout of which is illustrated in Figure 10c. The operation followed a “straight–curve–straight” arrangement, including 10 m straight sections and a 10 m curve with a 15 m radius. During the experiment, the control effect of the PER-MADDPG controller was evaluated. The scaled vehicle ran at 1 m/s, and the test lasted for 20 s.

The experimental data are shown in Figure 11, which shows the lateral displacement, attack angle, and active guidance torque of each wheelset. In the absence of active control, the maximum lateral displacement reached 8.02 mm on straight tracks and increased to 10.09 mm on curved tracks, indicating poor steering behavior. Conversely, with the proposed controller, the vehicle exhibited effective centering on straight tracks, limiting

the maximum lateral displacement to 2.05 mm. Flange guidance was avoided on the curved tracks, leading to strong curve-negotiation performance, with a maximum lateral displacement of 2.51 mm. The yaw angles were markedly reduced for both the front and rear IRWs and were maintained within 7.8 mrad. The control torques applied to the left and right wheels, which are equal in magnitude but in opposite directions, peaked at less than 2.5 N·m, satisfying the motor's rated maximum output. With each time step, the front and rear IRW agents independently make decisions on the control command, demonstrating the optimized dynamic performance of the scale vehicle for various operation tracks and underscoring the controller's real-time operational efficiency.



**Figure 11.** Test results for the scaled IRW vehicle. (a) Lateral displacement of the wheelsets, (b) Yaw angle of the wheelsets, (c) Control torque of the PMSMs.

## 6. Conclusions

In this study, we developed and validated an innovative active guidance control algorithm for IRW vehicles, leveraging the advantages of the MARL algorithm. We constructed a nonlinear dynamic model of a two-axle prototype IRW vehicle utilizing SIMPACK MBS software and established a feedback control loop based on the MARL. Within this framework, each independently rotating wheelset is assigned an independent decision-making agent to achieve local control, enabling distributed control strategies through collaboration with all other agents. We modeled the control of IRW vehicles as cooperative Markov games. We propose the PER-MADDPG algorithm based on CTDE, in which the MADDPG allows each agent to optimize its behavior strategy based on the global state through centralized training, and the PER mechanism is used to improve sample efficiency by sorting experiences stored in a SumTree structure according to their TD error. We also designed a reward function and corresponding observation and action spaces to guide the MARL model toward enhancing the running stability and curve-negotiation ability of IRWs. Our



proposed algorithm demonstrates convergence efficiency and higher final performance during reinforcement training, achieving a stable MARL learning process and surpassing the existing DRL and MARL algorithms. In MBS simulations, the PER-MADDPG algorithm improved the straight-line centering and curve-negotiation capabilities of IRWs while reducing wheel–rail wear compared to the classical model-based control strategy. Furthermore, we constructed a 1:5 scale vehicle based on Iwnicki’s similarity laws to test the controller in actual operation and deployed our proposed method on a GPU-based development board for real-time inference. The results of the experiments showed that our method significantly reduced the lateral displacement and yaw angle of the IRWs, with control outputs simultaneously meeting design requirements and avoiding flange–rail collisions, confirming its effectiveness. For the first time, we emphasize the ability of the MARL-based algorithm to adaptively learn railway vehicle dynamics, enhancing the overall steering performance through collaborative interactions among multiple IRW controllers and overcoming challenges in traditional controller designs that consider the nonlinear characteristics of vehicle models and homogenized treatment of IRWs. In addition, the proposed method eliminates the need to observe the entire vehicle’s dynamic state, reducing the observation dimensions and communication cost to accelerate the control response speed in practical applications.

Nevertheless, our study has several limitations that deserve further investigation. While we established various typical operational scenarios to train MARL agents for IRW guidance control, real-world applications involve additional complexities, such as varying wear levels on wheel tread profiles, deviations in suspension parameters, and track irregularities that could impact controller performance. Further investigations may focus on encompassing a more extensive range of operational scenarios in the learning environment and incorporating advanced machine learning strategies such as domain randomization [28], meta-learning [29], and cross-modal large models [30–32] to enhance controller robustness. Additionally, this research has focused mainly on lateral dynamic stability; however, an integrated approach that includes MARL with longitudinal characteristics from traction and braking systems is vital for comprehensive control optimization. In the future, we will extend our method to more widely used railway vehicle configurations with two bogies and four wheelsets or other wheelset types, such as the actuated solid-axle wheelset (ASW), instead of confining it to single-bogie vehicles primarily used in urban trams. Also, extensive experiments with MARL-based algorithms on full-size vehicles will be essential for deploying theoretical advancements in practical and scalable solutions for future railway vehicle control systems.

**Author Contributions:** Conceptualization, J.W. and Z.L.; methodology, J.W. and Z.L.; software, J.W. and Z.Y.; validation, J.W., Z.Y. and Z.J.; formal analysis, J.W. and Z.J.; investigation, J.W.; resources, J.W. and Z.J.; data curation, J.W. and Z.Y.; writing—original draft preparation, J.W.; writing—review and editing, Z.L. and Z.Y.; visualization, J.W.; supervision, Z.L.; project administration, Z.L. All authors have read and agreed to the published version of the manuscript.

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

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data that support the findings of this study are available from the corresponding author upon reasonable request.

**Acknowledgments:** We appreciate the comprehensive feedback and constructive recommendations provided by the peer reviewers, which have markedly improved the overall quality and clarity of the manuscript.

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

## Appendix A

**Table A1.** Vehicle parameters for Python-SIMPACK cosimulation.

Structural Components	Parameters	Value	Unit
Car Body	Mass	10,000	kg
	Roll inertia	10,000	kg·m <sup>2</sup>
	Pitch inertia	60,000	kg·m <sup>2</sup>
	Yaw inertia	60,000	kg·m <sup>2</sup>
Bogie Frame	Mass	1700	kg
	Roll inertia	1200	kg·m <sup>2</sup>
	Pitch inertia	900	kg·m <sup>2</sup>
	Yaw inertia	1300	kg·m <sup>2</sup>
Primary suspension	Longitudinal stiffness (per axle box)	500	kN·m <sup>-1</sup>
	Lateral stiffness (per axle box)	5000	kN·m <sup>-1</sup>
	Vertical stiffness (per axle box)	12,000	kN·m <sup>-1</sup>
	Longitudinal damping (per axle box)	50	kN·s·m <sup>-1</sup>
	Lateral damping (per axle box)	500	kN·s·m <sup>-1</sup>
	Vertical damping (per axle box)	500	kN·s·m <sup>-1</sup>
	Lateral span	1.1	m
	Longitudinal span	0.4	m
Secondary suspension	Longitudinal stiffness (per side)	150	kN·m <sup>-1</sup>
	Lateral stiffness (per side)	150	kN·m <sup>-1</sup>
	Vertical stiffness (per side)	600	kN·m <sup>-1</sup>
	Longitudinal damping (per side)	20	kN·s·m <sup>-1</sup>
	Lateral damping (per side)	20	kN·s·m <sup>-1</sup>
	Vertical damping (per side)	50	kN·s·m <sup>-1</sup>
	Lateral span	1.8	m
	Longitudinal span	0.5	m
IRW	Wheelbase	2.5	m
	IRW rolling radius	0.40	m

## References

1. Fu, B.; Giossi, R.L.; Persson, R.; Stichel, S.; Bruni, S.; Goodall, R. Active suspension in railway vehicles: A literature survey. *Railw. Eng. Sci.* **2020**, *28*, 3–35. [\[CrossRef\]](#)
2. Lu, Z.; Yang, Z.; Huang, Q.; Wang, X. Robust active guidance control using the  $\mu$ -synthesis method for a tramcar with independently rotating wheelsets. *Proc. Inst. Mech. Eng. Part F J. Rail Rapid Transit.* **2019**, *233*, 33–48. [\[CrossRef\]](#)
3. Heckmann, A.; Keck, A.; Grether, G. Active Guidance of a Railway Running Gear with Independently Rotating Wheels. In Proceedings of the IEEE Vehicle Power and Propulsion Conference, Gijon, Spain, 18 November–16 December 2020. [\[CrossRef\]](#)
4. Keck, A.; Schwarz, C.; Meurer, T.; Heckmann, A.; Grether, G. Estimating the wheel lateral position of a mechatronic railway running gear with nonlinear wheel–rail geometry. *Mechatronics* **2021**, *73*, 102457. [\[CrossRef\]](#)
5. Perez, J.; Mauer, L.; Busturia, J.M. Design of active steering systems for bogie-based railway vehicles with independently rotating wheels. *Veh. Syst. Dyn.* **2002**, *37* (Suppl. S1), 209–220. [\[CrossRef\]](#)
6. Ji, Y.; Ren, L.; Zhou, J. Boundary Conditions of Active Steering Control of Independent Rotating Wheelset Based on Hub Motor and Wheel Rotating Speed Difference Feedback. *Veh. Syst. Dyn.* **2018**, *56*, 1883–1898. [\[CrossRef\]](#)
7. Ahn, H.; Lee, H.; Go, S.; Cho, Y.; Lee, J. Control of the Lateral Displacement Restoring Force of IRWs for Sharp Curved Driving. *J. Electr. Eng. Technol.* **2016**, *11*, 1042–1048. [\[CrossRef\]](#)
8. Liu, X.; Goodall, R.; Iwnicki, S. Active control of independently rotating wheels with gyroscopes and tachometers—simple solutions for perfect curving and high stability performance. *Veh. Syst. Dyn.* **2021**, *59*, 1719–1734. [\[CrossRef\]](#)
9. Mei, T.X.; Goodall, R.M. Robust control for independently rotating wheelsets on a railway vehicle using practical sensors. *IEEE Trans. Control Syst. Technol.* **2001**, *9*, 599–607. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Lu, Z.; Sun, X.; Yang, J. Integrated Active Control of Independently Rotating Wheels on Rail Vehicles via Observers. *Proc. Inst. Mech. Eng. Part F J. Rail Rapid Transit.* **2017**, *231*, 295–305. [\[CrossRef\]](#)
11. Yang, Z.; Lu, Z.; Sun, X.; Zou, J.; Wan, H.; Yang, M.; Zhang, H. Robust LPV- $H_\infty$  Control for Active Steering of Tram with Independently Rotating Wheels. *Adv. Mech. Eng.* **2022**, *14*, 11. [\[CrossRef\]](#)



12. Grether, G.; Heckmann, A.; Looye, G. Lateral Guidance Control Using Information of Preceding Wheel Pairs. In *Advances in Dynamics of Vehicles on Roads and Tracks: Proceedings of the 26th Symposium of the International Association of Vehicle System Dynamics, IAVSD 2019, Gothenburg, Sweden, 12–16 August 2019*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020.
13. Lu, Z.; Wei, J.; Wang, Z. Active Steering Controller for Driven Independently Rotating Wheelset Vehicles Based on Deep Reinforcement Learning. *Processes* **2023**, *11*, 2677. [\[CrossRef\]](#)
14. Wei, J.; Lu, Z.; Yang, Z.; He, Y.; Wang, X. Data-Driven Robust Control for Railway Driven Independently Rotating Wheelsets Using Deep Deterministic Policy Gradient. In *The IAVSD International Symposium on Dynamics of Vehicles on Roads and Tracks*; Springer International Publishing: Cham, Switzerland, 2021.
15. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A Survey and Critique of Multiagent Deep Reinforcement Learning. *Auton Agent Multi-Agent Syst.* **2019**, *33*, 750–797. [\[CrossRef\]](#)
16. Su, J.; Adams, S.; Beling, P. Value-Decomposition Multi-Agent Actor-Critics. In Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 21, Virtual Conference, 2–9 February 2021; AAAI Press: Palo Alto, CA, USA, 2021.
17. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6382–6393.
18. Lu, K.; Li, R.; Li, M. MADDPG-Based Joint Optimization of Task Partitioning and Computation Resource Allocation in Mobile Edge Computing. *Neural Comput. Appl.* **2023**, *35*, 16559–16576. [\[CrossRef\]](#)
19. Xu, D.; Chen, G. Autonomous and Cooperative Control of UAV Cluster with Multi-Agent Reinforcement Learning. *Aeronaut. J.* **2022**, *126*, 932–951. [\[CrossRef\]](#)
20. Duan, W.; Tang, Z.; Liu, W.; Zhou, H. Autonomous Driving Planning and Decision Making Based on Game Theory and Reinforcement Learning. *Expert Syst.* **2023**, *40*, e13191. [\[CrossRef\]](#)
21. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. *arXiv* **2015**, arXiv:1511.05952. [\[CrossRef\]](#)
22. Plappert, M. Parameter Space Noise for Exploration in Deep Reinforcement Learning. *arXiv* **2017**, arXiv:1706.01905. [\[CrossRef\]](#)
23. Grether, G. Dynamics of a running gear with IRWs on curved tracks for a robust control development. *PAMM* **2017**, *17*, 797–798. [\[CrossRef\]](#)
24. Hur, H.; Shin, Y.; Ahn, D.; Ham, Y. Steering Performance Evaluation of Active Steering Bogie to Reduce Wheel Wear on Test Line. *Int. J. Precis. Eng. Manuf.* **2019**, *20*, 1591–1600. [\[CrossRef\]](#)
25. Yu, C.; Velu, A.; Vinitzky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. *arXiv* **2021**, arXiv:2103.01955. [\[CrossRef\]](#)
26. Schroeder de Witt, C.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P.H.S.; Sun, M.; Whiteson, S. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? *arXiv* **2020**, arXiv:2011.09533. [\[CrossRef\]](#)
27. Iwnicki, S.; Spiriyagin, M.; Cole, C.; McSweeney, T. *Handbook of Railway Vehicle Dynamics*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2019; pp. 826–864.
28. Chen, S.; Liu, G.; Zhou, Z.; Zhang, K.; Wang, J. Robust multi-agent reinforcement learning method based on adversarial domain randomization for real-world dual-uav cooperation. *IEEE Trans. Intell. Veh.* **2023**, early access. [\[CrossRef\]](#)
29. Wang, H.; Liu, Z.; Han, Z.; Wu, Y.; Liu, D. Rapid Adaptation for Active Pantograph Control in High-Speed Railway via Deep Meta Reinforcement Learning. *IEEE Trans. Cybern.* **2023**, early access. [\[CrossRef\]](#)
30. Qu, X.; Lin, H.; Liu, Y. Envisioning the future of transportation: Inspiration of ChatGPT and large models. *Commun. Transp. Res.* **2023**, *3*, 100103. [\[CrossRef\]](#)
31. Zheng, C.; Yan, Y.; Liu, Y. Prospects of eVTOL and modular flying cars in China urban settings. *J. Intell. Connect. Veh.* **2023**, *6*, 187–189. [\[CrossRef\]](#)
32. Liu, Y.; Wu, F.Y.; Liu, Z.Y.; Wang, K.; Wang, F.Y.; Qu, X.B. Can language models be used for real-world urban-delivery route optimization? *Innovation* **2023**, *4*, 100520. [\[CrossRef\]](#)

**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.