processes

MDPI

Article

Active Steering Controller for Driven Independently Rotating Wheelset Vehicles Based on Deep Reinforcement Learning

Zhenggang Lu, Juyao Wei * and Zehan Wang

Institute of Rail Transit, Tongji University, Shanghai 201804, China
* Correspondence: 1810988@tongji.edu.cn

Abstract: This paper proposes an active steering controller for Driven Independently Rotating Wheelset (DIRW) vehicles based on deep reinforcement learning (DRL). For the two-axle railway vehicles equipped with Independently Rotating Wheelsets (IRWs), each wheel connected to a wheel-side motor, the Ape-X DDPG controller, an enhanced version of the Deep Deterministic Policy Gradient (DDPG) algorithm, is adopted. Incorporating Distributed Prioritized Experience Replay (DPER), Ape-X DDPG trains neural network function approximators to obtain a data-driven DIRW active steering controller. This controller is utilized to control the input torque of each wheel, aiming to improve the steering capability of IRWs. Simulation results indicate that compared to the existing model-based H∞ control algorithm and data-driven DDPG control algorithm, the Ape-X DDPG active steering controller demonstrates better curving steering performance and centering ability in straight tracks across different running conditions and significantly reduces wheel–rail wear. To validate the proposed algorithm's efficacy in real vehicles, a 1:5 scale model of the DIRW vehicle and its digital twin dynamic model were designed and manufactured. The proposed control algorithm was deployed on the scale vehicle and subjected to active steering control experiments on a scaled track. The experimental results reveal that under the active steering control of the Ape-X DDPG controller, the steering performance of the DIRW scale model on both straight and curved tracks is significantly enhanced.

Keywords: independently rotating wheelsets; active steering; deep reinforcement learning; Ape-X DDPG algorithm

Citation: Lu, Z.; Wei, J.; Wang, Z. Active Steering Controller for Driven Independently Rotating Wheelset Vehicles Based on Deep Reinforcement Learning. *Processes* **2023**, *11*, 2677. https://doi.org/10.3390/pr11092677

Academic Editors: Ming-Jong Tsai and Ricky Min-Fan Lee

Received: 16 August 2023
Revised: 30 August 2023
Accepted: 4 September 2023
Published: 6 September 2023

1. Introduction

Independently Rotating Wheelsets (IRWs) allow the wheels on the same axle to rotate independently, decoupling the dependency between the yaw and lateral movement of the wheelsets. This configuration virtually eliminates the longitudinal creep force at the wheel–rail interface, which plays a significant role in self-guidance and steering in traditional solid-axle wheelsets. Consequently, while IRWs can inhibit hunting motion, the wheelsets' inherent self-steering capability and curving performance can also be compromised [1–3]. With the advancements in mechatronics and motor technologies for railway vehicles, Driven Independently Rotating Wheelsets (DIRWs) and the active steering control algorithms have been extensively studied in recent years. Railway vehicles with DIRWs, based on active steering control, apply steering torques directly to the IRWs through hub motors or wheel-side motors [4,5], which restores both the straight-track centering ability and the curve steering performance of the IRWs, while also eliminating unstable wheelset hunting motions, avoiding flange contact, and significantly reducing wheel–rail wear [6,7]. Moreover, vehicles equipped with DIRWs do not require additional actuators, enabling simultaneous traction and steering control [4,8,9], thereby reducing costs and enhancing reliability, representing a promising mechatronic solution for railway vehicles based on IRWs.

In existing designs of active steering controllers for vehicles, control algorithms like PID controllers [10–13], H∞ controllers [14,15], sliding mode controllers [16], and neural-

network-based controllers [17,18] have all been studied. Ahn [10] adopted a centering control approach, designed a PI controller, and conducted active steering control experiments on a small-scale roller rig using DIRWs driven by surface permanent magnet synchronous motors (SPMSMs). Liu [11] employed a series PID controller, using the wheelset's lateral displacement, yaw rate, and the differential rotational speed between the left and right wheels on the same axle as feedback. Lu [14], considering the wheel–rail nonlinearity and system parameter uncertainty as external disturbances, proposed a robust controller based on the μ-synthesis method and validated the results on a 1:5 scale track. In the Next Generation Train (NGT) project at the German Aerospace Center, Kurzeck [12] designed a PD controller for DIRWs, where the controller parameters were obtained through multibody simulation and pattern search, taking into account the impact of peak output torque.

Considering that the DIRW rail vehicle is a complex nonlinear system, with factors such as the geometric nonlinearity and the creep force nonlinearity of the wheel–rail contact of the railway vehicle, the variability of the suspension parameters, and other unmodeled dynamic characteristics, existing controller design methods find it difficult to achieve an optimal controller. Model-based controllers often neglect nonlinear factors in vehicle systems, design controllers according to a simplified linearized mathematical model, or treat nonlinear factors as external uncertainties, designing robust controllers that adapt to parameter changes. However, the simplified linearized dynamic model cannot accurately and fully consider vehicle dynamic characteristics [13,14,17,19], and the performance of controllers designed based on a simplified model is hard to guarantee in actual vehicles.

To address the problems in the design of active steering controllers mentioned above, this paper proposes an algorithm based on deep reinforcement learning (DRL) to enhance the straight-line centering performance and curve guidance ability. DRL is a branch of machine learning that builds on the foundation of the Markov Decision Process (MDP). The agent in DRL interacts with the environment and drives the system to maximize the expected reward under continuous stimulation from a reward–penalty system, overcoming the "curse of dimensionality" problem in solving high-dimensional sequential decision problems [20], using iterative training and deep neural networks to determine policy and value functions, obtaining an approximate optimal controller for nonlinear systems.

In the current research, DRL has been successfully applied to the control of different nonlinear systems, such as the intelligent driving of automobiles [21–24], active control of railway vehicles [25–28], and robotic control [29,30]. In automotive fault-tolerant control, a double Q-Learning algorithm has been proposed for the online determination of optimization weight factors by integrating DRL into a fault-tolerant coordinated controller, which ensures that vehicles can achieve optimal control strategies across various operating conditions [21]. In active safety control, DRL has been utilized to enhance the yaw motion stability of distributed drive electric vehicles using the Deep Deterministic Policy Gradients (DDPGs) to learn and control the vehicle's nonlinear dynamics [22]. A controller that combines DRL with Nonlinear Model Predictive Control (NMPC) has been proposed in [23] to achieve safe highway autonomous driving. Within a hybrid two-layer path planning architecture, a Double Deep Q-Network (DDQN) has been employed to train vehicles in choosing tactical behaviors according to the surrounding environment. DRL has been integrated with Mixed Traffic Flow (MTF) control [24], employing an Adam Optimization algorithm and Deep Q-Learning models to guide the longitudinal trajectory of Connected and Autonomous Vehicles (CAVs) on a typical urban roadway with signal-controlled intersections. In the field of railway transit, the Soft Actor–Critic (SAC) algorithm has been employed in the precise active control of pantograph–catenary systems (PCSs) in high-speed trains [25]. In terms of the decentralized management of energy storage systems in urban railways [26], the cooperative Markov game (MG) algorithm and the value decomposition network (VDN) have been introduced. Railway maintenance has also seen the application of the Advantage Actor–Critic (A2C) algorithm [27]. In the field of robotics, two types of DRL-based Deep Q-Learning algorithms, including DQN and DDQN, were used to enhance the autonomous learning abilities of mobile robots for collision avoidance

and navigation in unknown environments [29]. Similar methods have also been employed in obstacle avoidance for wheeled robots [30]. In summary, the diverse applications of DRL in areas ranging from automotive control to railway transit and robotics illustrate its capabilities for solving complex challenges in controller design.

This paper applies the Ape-X DDPG controller based on the DRL algorithm to the active steering control of DIRWs. The Ape-X DDPG algorithm is an improvement on the DDPG algorithm, which adopts a Distributed Prioritized Experience Replay (DPER) mechanism, further enhancing learning efficiency and stability by making efficient use of sample data. The Ape-X DDPG algorithm uses deep neural networks to learn the nonlinear dynamic characteristics of railway vehicle dynamics models, unmodeled characteristics, and parameter uncertainties, outputting the control torques required for active guidance through an end-to-end algorithm. Compared to the classical DDPG algorithm, Ape-X DDPG can better handle complex problems in high-dimensional, continuous action spaces and effectively utilize distributed computing resources to enhance the learning efficiency of controller training.

The remainder of this paper is organized as follows. Firstly, a dynamic model of a two-axle DIRW vehicle is established, with each wheel being connected to a wheel-side motor as an actuator. Secondly, we describe the algorithmic framework, set up the training loop for Ape-X DDPG, design controllers based on deep neural networks, and, within the vehicle dynamics simulation environment SIMPACK, compare them with the H∞ controller and the classical DDPG controller, demonstrating the superiority of the Ape-X DDPG controller in terms of control performance and training efficiency. Subsequently, a DRL controller trained based on the digital twin model is deployed on a 1:5 scale DIRW model, and active steering control experiments for DIRW are conducted on both straight and curved tracks to validate the control effects. Conclusions are drawn in the end.

2. DIRW Vehicle Dynamics Model

The two-axle DIRW vehicle, as illustrated in Figure 1, comprises two sets of IRWs, a bogie, drive motors, the vehicle body, and both primary and secondary suspensions. The vehicle body is connected to the IRWs through lateral and longitudinal spring–damper systems. The wheels on the left and right sides of the same axle of the IRW rotate independently, with the drive motors directly connected to each IRW. With the development of motor technology, wheel-side motors with high bandwidth and short response times have been widely adopted, capable of providing both the torque required for traction and active steering. The active steering controller controls the torque difference applied to the wheels on the left and right of the same axle, thereby generating a restoring torque in the yaw angle direction, ensuring the IRWs maintain straight-line stability and curve steering ability.

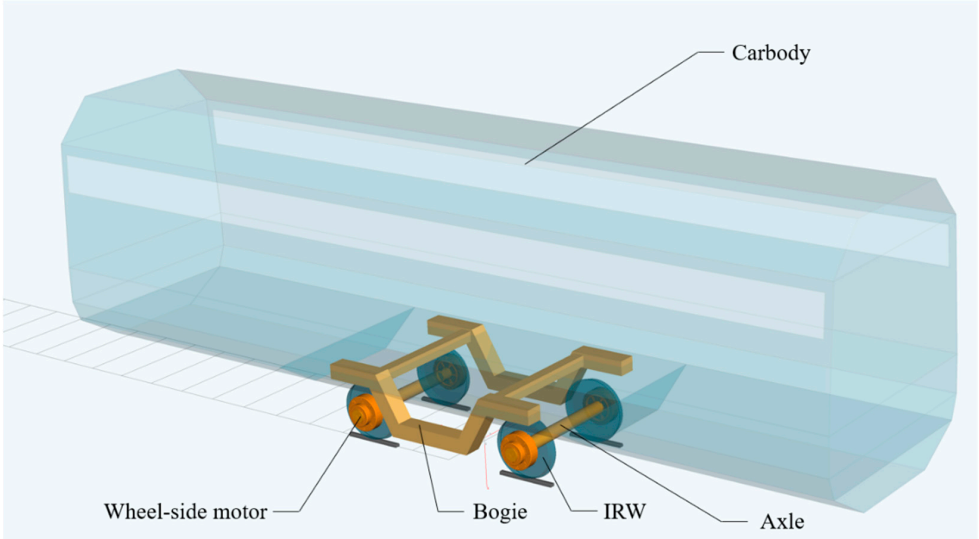

Figure 1. Structure of the DIRW vehicle.

The main focus of this paper is the active steering control system of the DIRW vehicle; hence, the dynamics analyzed pertain to the lateral dynamics model of the vehicle, as shown in Figure 2. In designing the active steering controller, the lateral displacements and yaw motions of the vehicle body and the front and rear IRWs, as well as the rotational motions of the wheels, are considered. The dynamics equations are presented as (1) to (10). All dynamic parameters are relative to the track coordinate system, where subscript 1 denotes the front wheelset, and subscript 2 denotes the rear wheelset.

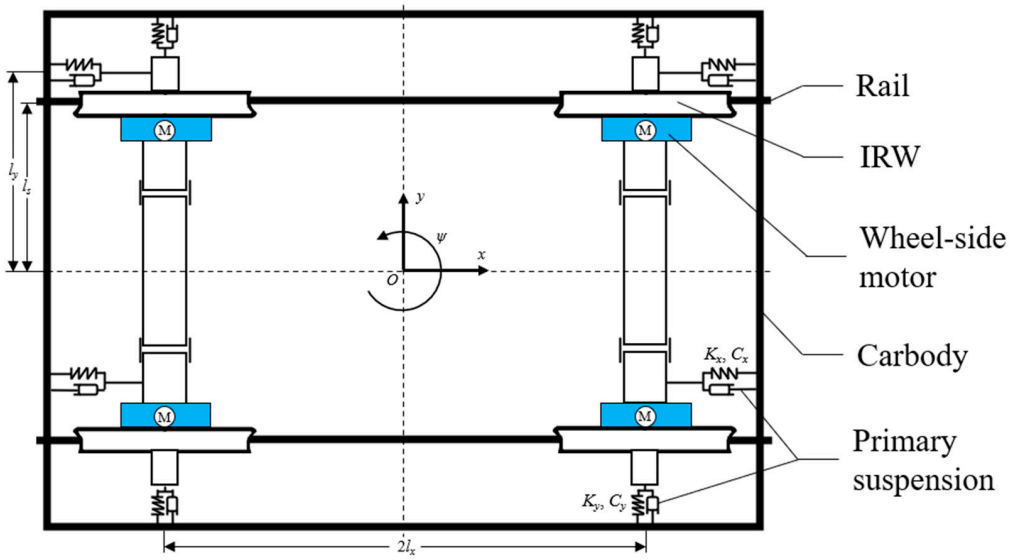

Figure 2. Lateral dynamics model.

The dynamics equations of the front IRW are as follows:

$$m_w\ddot{y}_{w1} + \frac{2f_{22}}{V_x}\dot{y}_{w1} + 2C_y(\dot{y}_{w1} - \dot{y}_b) + 2K_y(y_{w1} - y_b) - 2f_{22}\psi_{w1} - 2l_xC_y\dot{\psi}_b - 2l_xK_y\psi_b = m_w\left(\frac{V_x^2}{R_c} - g\theta\right) \quad (1)$$

$$I_{wz}\ddot{\psi}_{w1} + \frac{2f_{11}l_s^2}{V_x}\dot{\psi}_{w1} + \frac{2f_{11}\lambda l_s}{r_0}y_{w1} + 2l_y^2C_x\left(\dot{\psi}_{w1} - \dot{\psi}_b\right) + 2l_y^2K_x(\psi_{w1} - \psi_b) + \frac{2r_0f_{11}l_s}{V_x}\dot{\beta}_{w1} = \frac{2f_{11}l_s^2}{R_c} + \frac{2f_{11}\lambda l_s}{r_0}y_{t1} \quad (2)$$

$$I_{w1}\ddot{\phi}_{w1} + \frac{r_0^2f_{11}}{V_x}\dot{\phi}_{w1} + f_{11}\lambda y_{w1} + \frac{r_0f_{11}l_s}{V_x}\dot{\psi}_{w1} = \frac{r_0f_{11}l_s}{R_c} + f_{11}\lambda y_{t1} + T_{w1} \quad (3)$$

The dynamics equations of the rear IRW are as follows:

$$m_w\ddot{y}_{w2} + \frac{2f_{22}}{V_x}\dot{y}_{w2} + 2C_y(\dot{y}_{w2} - \dot{y}_b) + 2K_y(y_{w2} - y_b) - 2f_{22}\psi_{w2} + 2l_xC_y\dot{\psi}_b + 2l_xK_y\psi_b = m_w\left(\frac{V_x^2}{R_c} - g\theta_R\right) \quad (4)$$

$$I_{wz}\ddot{\psi}_{w2} + \frac{2f_{11}l_s^2}{V_x}\dot{\psi}_{w2} + \frac{2f_{11}\lambda l_s}{r_0}y_{w2} + 2l_y^2C_x\left(\dot{\psi}_{w2} - \dot{\psi}_b\right) + 2l_y^2K_x(\psi_{w2} - \psi_b) + \frac{2r_0f_{11}l_s}{V_x}\dot{\beta}_{w2} = \frac{2f_{11}l_s^2}{R_c} + \frac{2f_{11}\lambda l_s}{r_0}y_{t2} \quad (5)$$

$$I_{w2}\ddot{\phi}_{w2} + \frac{r_0^2f_{11}}{V_x}\dot{\phi}_{w2} + f_{11}\lambda y_{w2} + \frac{r_0f_{11}l_s}{V_x}\dot{\psi}_{w2} = \frac{r_0f_{11}l_s}{R_c} + f_{11}\lambda y_{t2} + T_{w2} \quad (6)$$

The dynamics equations of the carbody are as follows:

$$m_b\ddot{y}_b + 2C_y(2\dot{y}_b - \dot{y}_{w1} - \dot{y}_{w2}) + 2K_y(2y_b - y_{w1} - y_{w2}) = m_b\left(\frac{V_x^2}{R_c} - \theta_R\right) \quad (7)$$

$$\begin{aligned} &I_b\ddot{\psi}_b + 4\left(l_x^2C_y + l_y^2C_x\right)\dot{\psi}_b + 4\left(l_x^2K_y + l_y^2K_x\right)\psi_b - 2l_xC_y\left(\dot{y}_{w1} - \dot{y}_{w2}\right) \\ &-2l_xK_y(y_{w1} - y_{w2}) - 2l_y^2K_x(\psi_{w1} + \psi_{w2}) - 2l_y^2C_x\left(\dot{\psi}_{w1} + \dot{\psi}_{w2}\right) = 0 \end{aligned} \tag{8}$$

The angular velocities of the left and right wheels in the i-th IRW are as follows:

$$\omega_{Li} = \frac{V_x}{r_0} + \dot{\phi}_{wi}, \omega_{Ri} = \frac{V_x}{r_0} - \dot{\phi}_{wi}, i = 1, 2 \tag{9}$$

where m_w is the mass of the IRW; m_b is the mass of the vehicle body; I_{wz} is the yaw moment of inertia for the IRWs; I_{wy} is the rotational moment of inertia for each wheel; I_b is the yaw moment of inertia for the vehicle body; l_s is half the gauge of the IRWs; l_y and l_x are half the lateral and longitudinal distances of the suspension system, respectively; r_0 is the nominal rolling radius of the wheel; λ is the wheel conicity; f_{11} and f_{22} represent the longitudinal and lateral creep coefficients, respectively; V_x is the longitudinal vehicle speed; R_c is the curve radius; θ_R is the superelevation of curve line; y_{wi} is the lateral displacement of the IRWs; Ψ_{wi} is the yaw angle of the IRWs; T_{wi} is half the torque difference applied for active steering control on the left and right wheels of the same axle; ω_{Li} and ω_{Ri} are the rotational speeds of the left and right wheels of the IRWs, respectively; $\dot{\phi}_{wi}$ is half the speed difference between the left and right wheels of the IRWs; T_{Li} and T_{Ri} represent the input torques on the left and right wheels, respectively, equal in magnitude but opposite in rotational direction, satisfying Equation (10):

$$T_{Li} = T_{wi}, T_{Ri} = -T_{wi}, i = 1, 2 \tag{10}$$

Given that the DIRW vehicle is a complex nonlinear system, its creep coefficients, wheel conicity, and other dynamic parameters constantly change during operation. Unlike the design methods of many existing IRW active steering controllers, this study, when designing the controller based on the dynamic model as mentioned above, considers the wheel–rail contact parameters to be constantly changing values obtained through nonlinear wheel–rail contact calculations rather than pre-setting them as fixed values to achieve stronger robustness when deployed in real-world environments.

3. Controller Design Based on Ape-X DDPG Algorithm

The fundamental learning approach of reinforcement learning agents involves interacting with their environment, autonomously optimizing behavioral policies to achieve a higher expected reward. This expected reward is often defined based on the objectives of the target task. Controllers based on DRL can achieve end-to-end optimization and, through deep learning algorithms, produce agents with superior control outcomes. In the design of the DIRW active steering controller, the agent continuously trains and optimizes, enabling the IRWs under the agent's control to achieve improved centering performance and enhancing the IRWs' running stability and energy efficiency. This study adopts the Ape-X DDPG algorithm for controller implementation and collects data and trains the agent through interaction with the DIRW vehicle.

3.1. Basic DRL Theory

The mathematical foundation of reinforcement learning is the MDP [31]. When the DIRW active steering controller acts as an agent, the IRW's operation on straight and curved tracks can be described as shown in Figure 3. An MDP can be represented as a four-tuple (S, A, P, R). Here, S represents the state space of the agent, a stands for the agent's action space, P is the state transition function, and R is the reward function.

Within the DRL framework, the agent interacts with the environment E at each time step t to collect data, optimizing the behavioral policy μ based on the reward function R. This policy is parameterized by a deep neural network. The agent obtains the state s_t from observations in environment E and samples action $a_t = \mu(s_t)$ based on policy μ. After the

agent performs the action a_t, the environment transitions to the next state s_{t+1} according to the model dynamics $P(s_{t+1} \mid s_t, a_t)$ and receives a reward r_t from the environment. During this process, the agent stores the state and action transition (s_t, a_t, r_t, s_{t+1}) in the experience replay buffer D for subsequent learning.

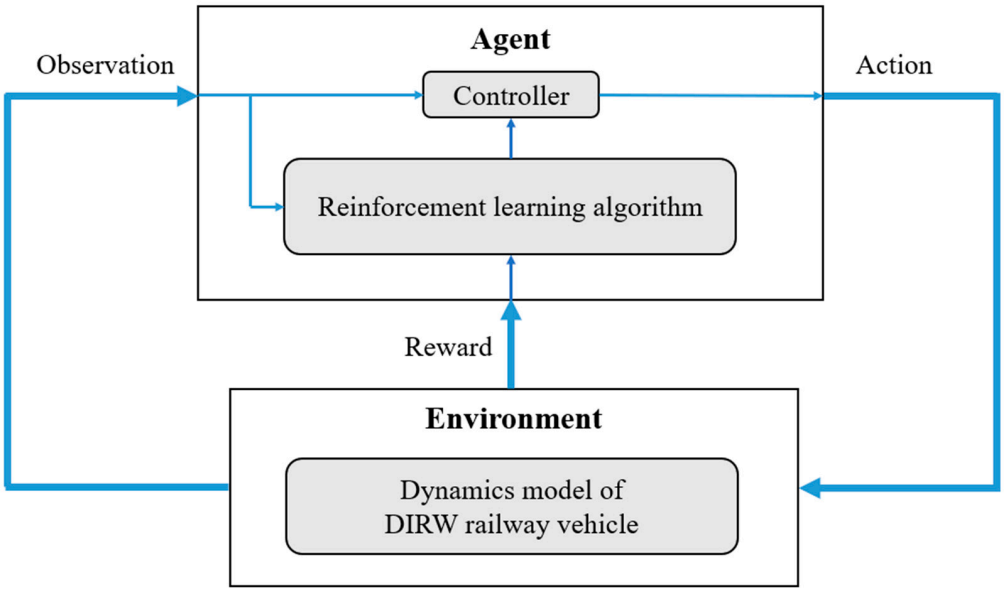

Figure 3. The overview of the DRL-based method.

The reinforcement learning algorithm aims to learn a policy that maximizes the expected return. To achieve this, the objective function that needs optimization is shown in Equation (11), which measures the quality of policy μ.

$$J(\mu) = E_{\rho\sim\mu}\left(\sum_{0}^{T}\gamma^t r(s_t, a_t)\right) \tag{11}$$

where $\rho = (s_0, a_0, s_1, a_1, \ldots)$ represents the trajectory of states and actions, γ is the discount factor adjusting the emphasis on immediate rewards and future rewards, and $E_{\rho\sim\mu}$ represents the mathematical expectation under policy μ. By selecting appropriate optimization algorithms, such as the policy gradient method and the Q-Learning method, one can find the optimal policy μ^* such that $J(\mu)$ is maximized as shown in Equation (12).

$$\mu^* = \underset{\mu}{\operatorname{argmax}}\, E_{\rho\sim\mu}\left(\sum_{0}^{T}\gamma^t r(s_t, a_t)\right) \tag{12}$$

3.2. DDPG Algorithm

To find the optimal policy μ^* for the active steering controller and maximize $J(\mu)$, this paper uses the DDPG algorithm as the basis for the DRL controller, further introducing the DPER to enhance the learning capability of the DRL algorithm.

The DDPG algorithm is an improved offline, model-free deep reinforcement learning algorithm based on the Deterministic Policy Gradient (DPG) suitable for continuous action space problems. DDPG combines the framework of Off-policy Deterministic Actor–Critic (OPDAC) and the training techniques of DQN, providing an effective way to learn deterministic policies in continuous action spaces [32]. In DDPG, both the policy function $\mu(S; \theta^\mu)$ and action value function $Q(S, A; \theta^Q)$ are represented using deep neural networks, forming the actor–critic algorithm, where θ^μ represents the policy network parameters, and θ^Q represents the action value network parameters. As the policy function, the actor network outputs continuous action values in a greedy way, as shown in Equation (13). The critic network estimates the action value function $Q(S, A)$ based on the state and the action

provided by the actor network. At each time t, Equation (14) is used to approximate the value function.

$$a_t = \mu(s_t; \theta^\mu) + N_t \tag{13}$$

$$q_t = Q(s_t, a_t; \theta^Q) \tag{14}$$

where N_t is the action perturbation used to enhance the efficiency of environment exploration during the learning process. We used Ornstein–Uhlenbeck noise with its standard deviation denoted as σ_{OU}. q_t is the estimation of the Q-value for the current state and action.

To stabilize the learning process, DDPG borrows the target network structure from DQN, constructing both the target policy network and the target value network. A soft update strategy for the target network is introduced, as shown in Equation (15).

$$\begin{cases} \theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \end{cases} \tag{15}$$

where $\theta^{\mu'}$ and $\theta^{Q'}$ represent the neural network parameters for the target networks μ' and $\theta^{Q'}$, respectively; τ is the soft update rate, used to control the extent of the updates to the target network.

For the policy and value networks, the DDPG algorithm uses the experience replay mechanism. DDPG stores the transition samples (s_t, a_t, r_t, s_{t+1}) in the experience buffer D. When optimizing the neural network parameters, DDPG randomly draws a batch of samples, denoted as I, from the experience buffer for training. This helps reduce the correlation between samples. Based on the target networks μ' and Q', the respective loss function for Q and the gradient update for μ are presented in Equations (16) and (17).

$$L\left(\theta^Q\right) = \frac{1}{|I|}\sum_i^I \left(r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}; \theta^{\mu'}); \theta^{Q'}) - Q(s_i, a_i; \theta^Q)\right)^2 \tag{16}$$

$$\nabla_{\theta^\mu} J \approx \frac{1}{|I|}\sum_i^I \nabla_{\theta^\mu} Q\left(s_i, \mu(s_i; \theta^\mu); \theta^Q\right) \cdot \nabla_{\theta^\mu} \mu(s_i; \theta^\mu) \tag{17}$$

Updating the parameters of the target network by smoothly updating the original network parameters helps stabilize the training process, suppresses learning oscillations in Q and μ during training, and achieves the purpose of stabilizing the algorithm's learning process. The neural network training process in DDPG is shown in Figure 4.

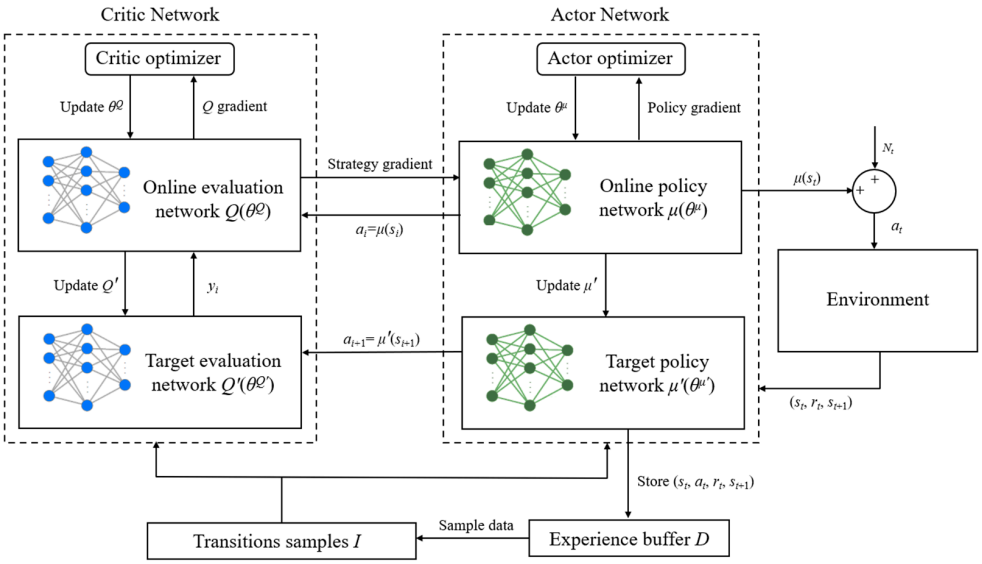

Figure 4. The DDPG algorithm structure framework.

3.3. Introduction of Prioritized Experience Replay

The conventional DDPG algorithm uses Uniform Random Sampling (URS) to draw samples from the experience replay (ER) buffer, effectively decoupling sample correlations to meet the requirements of offline training. However, URS does not leverage the significance of the samples efficiently. Some samples contribute greatly to the learning process but are not efficiently utilized due to the uniform sampling approach, resulting in a decreased learning efficiency for valuable experiences. To address this deficiency, the Ape-X DDPG algorithm adopted Prioritized Experience Replay (PER) in place of URS to enhance the utilization of high-value samples.

Ape-X DDPG first computes the priority of samples based on their Temporal Difference (TD) error and then employs Prioritized Weighted Exponential Normalization Sampling (PWENS) as the sampling probability, ensuring a more balanced sampling mechanism. In line with the DDPG algorithm, the TD error, y_i, for any sample i can be calculated as shown in Equation (18).

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}); \theta^{Q'}) - Q(s_i, a_i; \theta^Q) \tag{18}$$

PER uses the computed TD error to represent the value of a sample. A sample's priority is higher when its absolute TD error is larger. For different samples i, the probability p_i of each sample being chosen is calculated as indicated in Equation (19).

$$p_i = \frac{p_i^\alpha}{\sum_{k=1}^{|D|} p_k^\alpha} \tag{19}$$

where α is a priority adjustment parameter ranging from 0 to 1. α balances between uniform sampling and priority sampling. When α = 0, all experience samples have equal sampling probability, regressing to conventional ER without considering priorities. When α = 1, the experience samples are sampled strictly based on their priority in the ER buffer.

To further mitigate the effect of oversampled high-priority experiences, PER introduces Importance Sampling (IS) weights to correct the biases introduced by the priority-based sampling. This ensures the stability of the learning process. The IS weight, w_i, for the sample i is presented in Equation (20).

$$w_i = \left(\frac{1}{|D| \cdot p_i}\right)^\beta \tag{20}$$

where β is a hyperparameter ranging from 0 to 1, used to counteract the bias induced by the frequent replay of high-value samples. In the Ape-X DDPG algorithm, for the critic network, each sample's IS weight is used to weight the TD error, resulting in a weighted TD error which is then used to compute the loss function of the critic network, as shown in Equation (21).

$$L(\theta^Q) = \frac{1}{|I|} \sum_i^I w_i [y_i - Q(s_i, a_i; \theta^Q)]^2 \tag{21}$$

For the actor network, the IS weight of each sample is employed to weight the policy gradient, yielding a weighted policy gradient. This weighted policy gradient is then used to update the parameters of the actor network, as illustrated in Equation (22).

$$\bigtriangledown_{\theta^\mu} J \approx \frac{1}{|I|} \sum_i^I w_i [\bigtriangledown_A Q(s_i, \mu(s_i; \theta^\mu); \theta^Q) \cdot \bigtriangledown_{\theta^\mu} \mu(s_i; \theta^\mu)] \tag{22}$$

3.4. Structure of Ape-X DDPG Algorithm

The structure of the Ape-X DDPG algorithm is illustrated in Figure 5. It encompasses three components: the Learner, the Actor, and the global experience replay buffer. The Actor is responsible for collecting data samples generated during the simulation of the IRW's steering system under the agent's control. The Actor deploys multiple independent interaction environments using parallel, distributed computing techniques, composed of the local DDPG policy network, the local sample pool D_l, and the DIRW vehicle dynamics interaction environment. The local DDPG policy aims to enhance the guidance and steering ability of the IRWs while being controlled, which fetches the network parameters from the parameter buffer. The local sample pool serves as a cache for the experiences generated by the local Actor. When the volume of samples reaches the storage threshold D_m, these experiences are forwarded to the global experience replay buffer D.

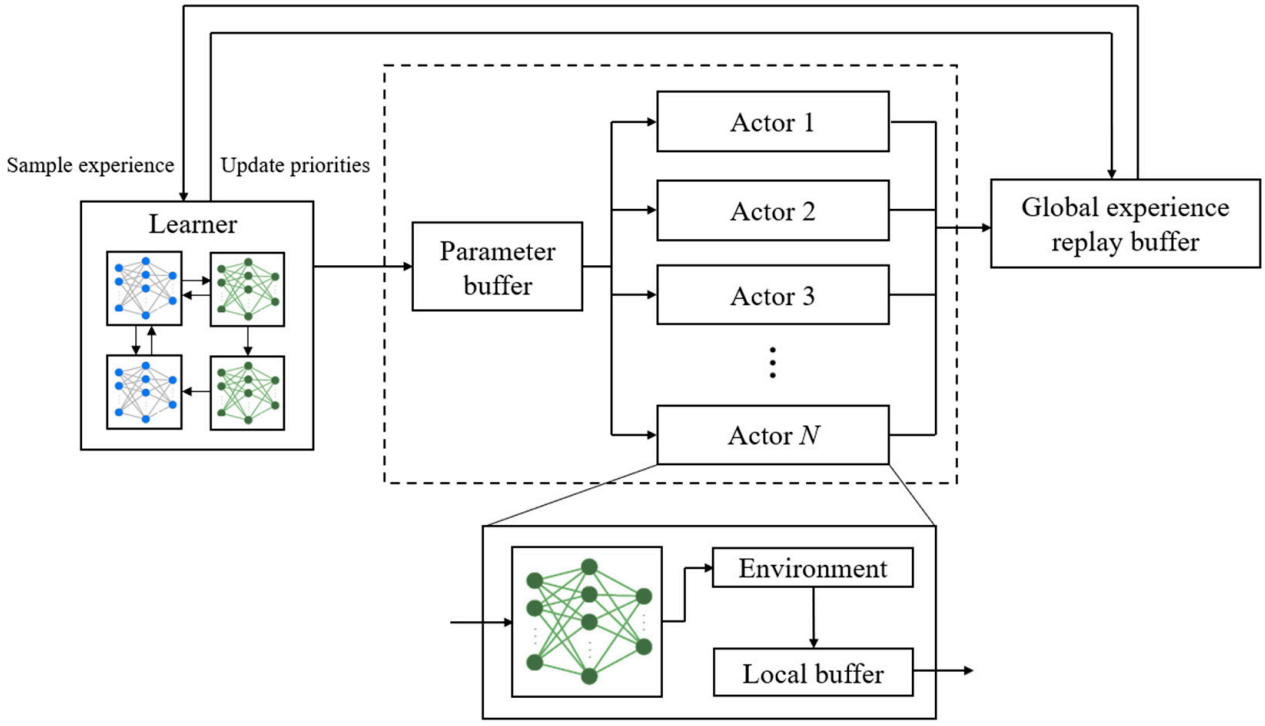

Figure 5. Ape-X DDPG algorithm structure.

The global experience replay buffer is dedicated to storing all samples generated during interactions. The Learner manages the actual learning process, drawing experiences from the global experience replay buffer and then updating the policy network and value network using the DDPG algorithm. After completing each training iteration, the Learner saves the trained network parameters into the parameter buffer and calculates the TD error for each sample to send to the global experience replay buffer. This TD error is used to update the sample's priority within the shared buffer.

Ape-X DDPG parallelly executes the interaction, learning, and storage processes, achieving the decoupling of interactions and training. By deploying multiple independent Actors, the training speed and efficiency of sample collection are significantly enhanced. The algorithms of Ape-X DDPG's Actor (Algorithm 1) and Learner (Algorithm 2) are described as follows:

Algorithm 1: Pseudocode of the Actor

Input: Initial state s_0 from the environment; policy network parameters θ^{μ} from the Learner's parameter buffer; local buffer size D_m; maximum interaction timesteps T
Output: Updated experience tuples and priorities to shared experience replay buffer D

Initialize local buffer D_l and shared experience replay buffer D.
Initialize state s_0.
for $t = 1$ **to** T **do**
    Calculate action $a_{t-1} = \mu(s_{t-1}; \theta^{\mu})$.
    Obtain reward r_t, next state s_t, and discount factor γ_t based on action a_{t-1}.
    Add experience tuple $(s_{t-1}, a_{t-1}, r_t, \gamma_t)$ to local buffer D_l.
    if the size of D_l reaches D_m **then**
        Retrieve experience tuples from D_l, denoted as τ_l.
        Calculate the priority p for each experience tuple in τ_l.
        Add τ_l and corresponding priorities p to D.
    end if
    Periodically fetch the latest network parameters θ^{μ} from the Learner.
end for
end procedure

Algorithm 2: Pseudocode of the Learner

Input: Maximum interaction timesteps T; initial value network parameters θ^{Q}; initial policy network parameters θ^{μ}; initial shared experience replay buffer D
Output: Updated network parameters θ^{Q} and θ^{μ}

Initialize value network parameters θ^{Q} and policy network parameters θ^{μ}.
Send θ^{Q} and θ^{μ} to the parameter buffer shared with Actors.
for $t = 1$ **to** T **do**
    Sample training samples I and IS weights w_i from D using PER sampling.
    Compute the loss function L according to Equation (21) based on I, θ^{Q} and θ^{μ}.
    Update the value network parameters θ^{Q} using L.
    Compute the policy network gradient according to Equation (22) and update θ^{μ}.
    Recalculate priorities p for sample I based on the TD error y_i for each training sample.
    Update the priorities of samples in D with the newly computed priorities p.
    if the size of D reaches maximum capacity **then**
        Remove the experiences with the lowest priorities from D.
    end if
end for
end procedure

3.5. DRL-Based Controller Design

For the training process of the DIRW vehicle's active steering controller based on DRL, the definition of the state space, action space, and reward function is very important. These directly influence the steering performance of the controller agent trained in the vehicle dynamics environment.

3.5.1. Definition of State Space

In this study, the state space S is defined as continuous and represents the vehicle dynamics data obtained through sensor observations. Specifically, the agent's state space includes the longitudinal speed V_x of the DIRW vehicle and the kinetic variables D_x of the two IRWs of the two-axle railway vehicle. The dynamic variables D_x of IRWs include the lateral displacement and the yaw angle of the front and rear IRWs relative to the track and the difference in rotation speed between the left and right wheels of each IRW. Among them, the lateral displacement and yaw angle of the IRWs can be indirectly measured by displacement sensors arranged on the wheelsets. The difference in wheel rotation speed

can be calculated from the motor encoder. Additionally, the first-order derivatives of these sensor measurements are also included in state space. Hence, the dynamic variables D_x of the two IRWs obtained through sensor observation in the state space are shown in Equation (23).

$$D_x = \begin{bmatrix} y_{w1} & \psi_{w1} & \dot{\phi}_{w1} & \dot{y}_{w1} & \dot{\psi}_{w1} & \ddot{\phi}_{w1} & y_{w2} & \psi_{w2} & \dot{\phi}_{w2} & \dot{y}_{w2} & \dot{\psi}_{w2} & \ddot{\phi}_{w2} \end{bmatrix} \quad (23)$$

Considering that the vehicle has different speeds under different running conditions, the longitudinal speed V_x of the vehicle is also one of the observations. The final state space S of the agent is represented as in Equation (24).

$$S = \begin{bmatrix} y_{w1} & \psi_{w1} & \dot{\phi}_{w1} & \dot{y}_{w1} & \dot{\psi}_{w1} & \ddot{\phi}_{w1} & y_{w2} & \psi_{w2} & \dot{\phi}_{w2} & \dot{y}_{w2} & \dot{\psi}_{w2} & \ddot{\phi}_{w2} & V_x \end{bmatrix} \quad (24)$$

3.5.2. Definition of Action Space

The active steering of IRWs is controlled through the torque of the wheel-side motors, with the DRL-based controller's output in a continuous action space. For the DIRWs, the controller's output separately controls the half torque difference between the left and right wheels for both the front and rear IRWs. The action space A is defined in Equation (25).

$$A = [T_{w1} T_{w2}] \quad (25)$$

where T_{w1} and T_{w2} are half of the input torque difference of the left and right wheels of the front and rear IRWs, respectively, satisfying $-T_{\max} \leq T_{w1}, T_{w2} \leq T_{\max}$, in which $T_{\max}$ represents the maximum output torque limit of the active steering controller.

Based on the defined action space, during the operation of the DIRW vehicle, the left and right wheels of each IRW will be subjected to force torques of equal magnitude but opposite directions. The left and right wheels of the front IRW receive steering torques of T_{w1} and $-T_{w1}$, respectively. Similarly, the left and right wheels of the rear IRW receive steering torques of T_{w2} and $-T_{w2}$, respectively.

3.5.3. Control Objectives and Reward Function

The active steering controller of DIRW vehicles is required to meet multiple objectives. First and foremost, the controller must improve the steering stability of the IRWs. Under the guidance of the active steering controller, the IRW should avoid flange contact on both straight and curved tracks. Therefore, the controller should be designed to minimize the lateral displacement and yaw angle of the IRWs relative to the track, ensuring the steering capability of the IRWs. Based on these control objectives, in the DRL reward function, the centering performance reward of the IRW is defined as r_1, as shown in Equation (26).

$$r_1 = \eta_{11} \cdot (y_{\max} - |y_{w1}|)^2 - \eta_{12} \psi_{w1}^2 + \eta_{21} \cdot (y_{\max} - |y_{w2}|)^2 - \eta_{22} \psi_{w2}^2 \quad (26)$$

where η_{11}, η_{12}, η_{21}, and η_{22} are the weighting coefficients for the lateral displacement and yaw angle of the two IRWs; $y_{\max}$ denotes the maximum lateral clearance between the wheels and rail. Since the front IRWs are more prone to losing steering capabilities, resulting in flange guidance when the vehicle runs on a curved track, the reward coefficients η_{11} and η_{12} for the front IRW are set to be greater than the weighting coefficients η_{21} and η_{22} for the rear IRW.

Furthermore, the control torque must be constrained within a reasonable range, which ensures the performance of the controller while reducing actuator output torque to improve energy efficiency. The reward function of motor output is defined as r_2, as illustrated in Equation (27).

$$r_2 = \eta_{13} \cdot (T_{\max} - |T_{w1}|) + \eta_{23} \cdot (T_{\max} - |T_{w2}|) \quad (27)$$

where η_{13} and η_{23} are the reward coefficients for the motor output torque.

The controller should also ensure the vehicle's smoothness, preventing the active steering control from decreasing comfort. We measure this aspect using the vehicle's lateral acceleration. r_3 is defined as the contribution to the agent's reward function for vehicle comfort, as shown in Equation (28). If the vehicle's lateral acceleration exceeds a threshold, a significant penalty is applied in the reward function design.

$$r_3 = \begin{cases} 0 & , |\ddot{y}_b| \leqslant a_{\max} \\ -r_p & , |\ddot{y}_b| > a_{\max} \end{cases} \tag{28}$$

where $a_{\max}$ is the maximum allowed lateral acceleration of the vehicle under active steering control; r_p is the penalty given to the agent when lateral acceleration surpasses the threshold.

In summary, the reward function is as shown in Equation (29).

$$r_t = r_1 + r_2 + r_3 \tag{29}$$

The training objective of the active steering controller is to maximize the designed reward function r_t, ensuring the controlled IRWs reduce the lateral displacement and attack angle and restore the running stability. Simultaneously, the agent should learn to decrease control torque while achieving the desired control effects.

3.5.4. DRL Interaction and Training Process

Within our Ape-X DDPG training framework, the neural network parameters for the policy network (actor network) and value network (critic network) are initialized randomly. For storing experience tuples, a shared experience replay buffer needs also to be initialized. We set up N parallel Actors that perform action sampling in the environment based on the current policy. Each episode has a maximum time step of 20 s, during which experience tuples are collected and stored in their respective local experience buffers.

During the data collection phase, each episode's training process for every Actor is shown in Figure 6. Inside each Actor unit, the active steering controller agent calculates actions based on the policy network. This action, representing half of the torque difference between the left and right wheels for both the front and rear IRWs, is applied accordingly. According to the dynamic response of the DIRW vehicle, the observation for the next time step is obtained as the next state, and the corresponding reward is also calculated. The agent's control frequency is 100 Hz, meaning that every 0.01 s, the IRW will receive active steering torque from the agent. During the DRL controller's training process, the interaction between the agent and the environment will be terminated, and the environment state will be reinitialized if any of the following four situations occur:

(a) The training episode reaches the maximum timestep T.
(b) The lateral displacement of the IRWs reaches the maximum wheel–rail clearance.
(c) The control torque output from the policy network exceeds the maximum limit.
(d) Under the effect of the active steering controller, the lateral acceleration of the vehicle body exceeds the threshold.

To enhance the adaptability of the agent under different vehicle operating conditions, every time the vehicle dynamics environment is initialized, the longitudinal speed, curve radius, and superelevation of the vehicle's operation are reset. The training conditions for each episode are randomly selected from five different operating speeds and curve radii. The operating conditions for the DIRW vehicle are shown in Table 1.

Experience tuples generated from the interaction between the agent and the vehicle dynamics environment are stored in their respective local experience buffers. When the local buffer reaches the storage threshold, experiences are batched and sent to the global shared experience replay buffer. At the same time, the Actor periodically retrieves the latest policy network parameters from the Learner.

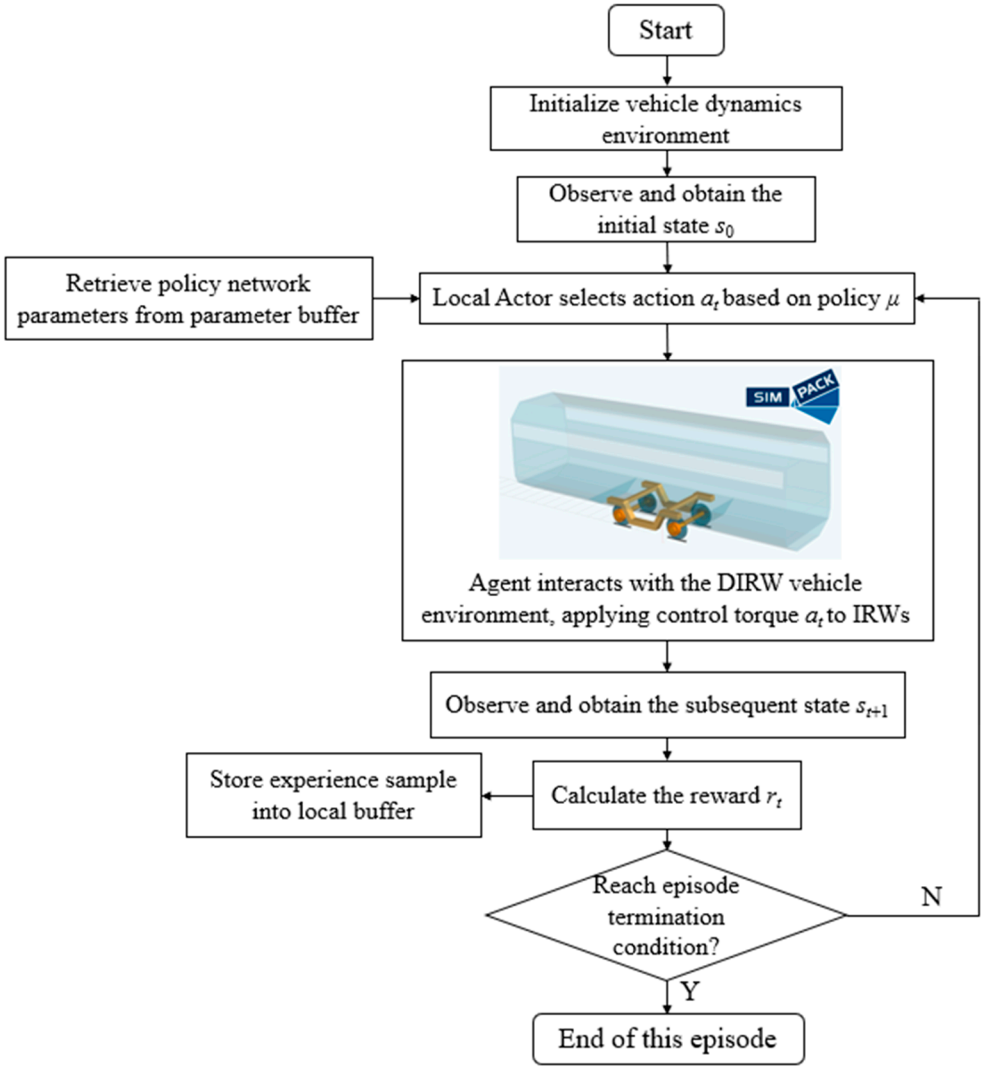

Figure 6. Interaction process of Actors for each episode.

Table 1. Operation conditions of the training agent.

	Track 1	**Track 2**	**Track 3**	**Track 4**	**Track 5**
Track Type	Straight line	Straight line	Curve line	Curve line	Curve line
Curve Radius (m)	—	—	70	250	600
Speed (km/h)	80	120	30	80	100
Cant (mm)	—	—	0	150	80
Track Irregularities	AAR5	AAR5	AAR5	AAR5	AAR5

During the model training phase, the Learner prioritizes sampling from the global experience replay buffer and extracts training samples to calculate the loss function. This is used to update the parameters of the value network and to update the policy network parameters using the gradient of the policy network. After updating the value network, the TD error for each sample is recalculated to update the priority of the samples in the global experience replay buffer. The training process is repeated continuously, enabling the continuous optimization of the policy network and value network until the model's performance converges.

4. Training and Simulation Results

4.1. Training Environment

This paper conducts co-simulation through the multibody dynamics simulation software SIMPACK and the Python-based reinforcement learning framework RLlib [33,34] to validate the learning and control capabilities of the Ape-X DDPG active steering controller. Specifically, RLlib is responsible for implementing Ape-X DDPG, while SIMPACK handles the dynamics simulation for the DIRW two-axle railway vehicle. The SIMPACK Realtime API facilitates data communication between SIMPACK and RLlib. In SIMPACK, the parameters for the two-axle DIRW railway vehicle dynamics model are shown in Table A1, and the UIC60 rail profile and S1002 wheel profile are used. The contact tangential forces are calculated using the FASTSIM method. The state values and rewards required for DRL training are calculated by SIMPACK and transferred to RLlib at each time step.

All DRL training was conducted on a server running Ubuntu 20.04, equipped with dual Intel Gold 6132 CPUs, clocked at 2.60 GHz, and an NVIDIA GeForce RTX 2080 Ti GPU used for deep learning computations. In RLlib, the Ape-X DDPG algorithm is implemented using Pytorch [35]. Heterogeneous training is adopted, with Actors interacting with the vehicle dynamics environment distributed across different CPU cores for parallel computing. In contrast, the Learner is assigned to the GPU for neural network gradient descent optimization. In the Learner unit, both the policy network μ and the value network Q use fully connected deep neural networks, each with four hidden layers. The unit counts for these layers are 200, 400, 400, and 50. ReLU is chosen as the activation function for the hidden layers, with Adam [36] being used as the optimizer for the neural network parameters. The input layer network unit count is determined by the state space, and Q receives the corresponding action in the second hidden layer. For the output layer activation function, both μ and Q use the tanh activation function. Detailed DRL neural network training parameters are presented in Table 2.

Table 2. Training parameters of DRL.

Parameter	Value	Description
LR_{μ}	10^{-4}	Learning rate of the policy network μ
LR_{Q}	10^{-3}	Learning rate of the value network Q
γ	0.99	Temporal discount rate
τ	10^{-4}	Soft update rate
batchsize	512	Batch size for training
$\|D\|$	10^{6}	Storage limit for the public sample pool
α	0.8	Exponent for priority sampling
β	0.3	IS hyperparameter
D_m	10^{4}	Storage limit for the local sample pool
N	10	Number of Actors
σ_{OU}	0.2	Standard deviation of OU disturbance noise
$T_{\max}$	1200 N·m	Maximum output control torque
y_{max}	9.2 mm	Maximum lateral clearance between wheel and rail
η_{11}	1	Reward coefficient for the lateral displacement of the front IRW
η_{12}	0.5	Reward coefficient for the yaw angle of the front IRW
η_{21}	0.2	Reward coefficient for the lateral displacement of the rear IRW
η_{22}	0.1	Reward coefficient for the yaw angle of the rear IRW
η_{13}	5	Reward coefficient for the steering torque of the front IRW
η_{23}	2	Reward coefficient for the steering torque of the rear IRW
$a_{\max}$	2.5 m·s^{-2}	Lateral acceleration limit
r_p	100	Penalty for exceeding lateral acceleration limit

4.2. Training Comparison of DRL

A comparative experiment was conducted to evaluate the influence of the number of Actors in Ape-X DDPG on its performance and to compare it against other DRL algorithms, such as DDPG, Proximal Policy Optimization (PPO), and Twin Delayed DDPG (TD3). In our experimental setup, PPO employs mini-batches of size 64, and TD3 uses a policy delay setting of 2. PPO aims to stabilize the training process by optimizing a surrogate objective function and limiting policy updates, as referenced in [37]. TD3, an extension of DDPG, uses dual value function networks to improve bootstrapping accuracy and introduces delayed policy updates to reduce estimation errors [38].

The impact of the number of Actors in Ape-X DDPG is significant. Therefore, we conducted simulation experiments with three different numbers of Actors—2, 5, and 10—to investigate their effects on learning efficiency and convergence speed. We measured the performance of each DRL algorithm through their respective reward curves after 2×10^5 training episodes, as shown in Figure 7.

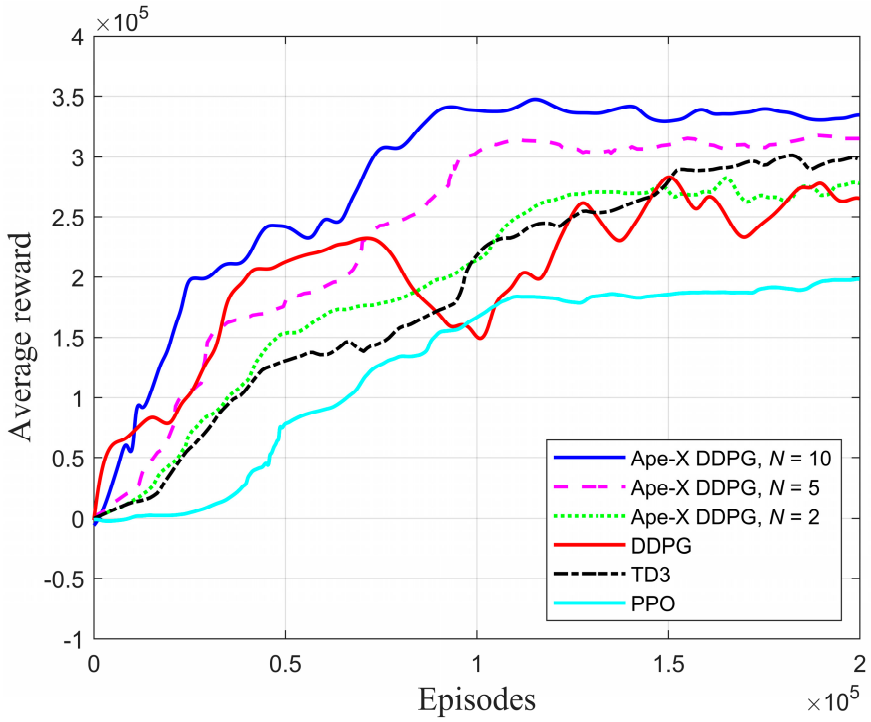

Figure 7. Average reward value vs. episodes.

Our results indicate that the average reward values after convergence for the Ape-X DDPG configurations are positively correlated with the number of Actors (N). Ape-X DDPG, a distributed version of DDPG, particularly excels when N is set to 5 or 10. In contrast, standard DDPG shows a slower learning curve and oscillations, achieving lower final rewards than Ape-X DDPG when $N = 10$. Ape-X DDPG not only converges more quickly but also demonstrates enhanced stability during training, leading to higher final rewards. While TD3 improves upon DDPG, it still does not outperform Ape-X DDPG when N is 5 or 10.

To test the robustness of these DRL algorithms, we conducted further experiments with 1000 randomly sampled groups from the initial state space. Each DRL algorithm used network parameters obtained from previous training. The test evaluated the ability of these algorithms to complete 20 s of active steering under various operating conditions for DIRW vehicles. Figure 8 plots the number of successful episodes for each controller.

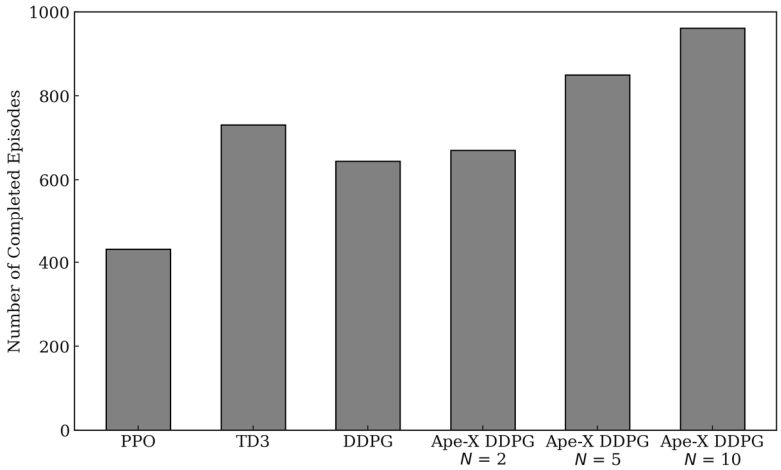

Figure 8. Number of completed episodes by each DRL algorithm in the test set.

In Figure 8, DDPG and Ape-X DDPG with two Actors show similar performance, while TD3 outperforms both. However, Ape-X DDPG with 10 Actors achieves the best performance, successfully completing 961 active steering episodes under various initial operating conditions without triggering early termination.

In summary, the Ape-X DDPG algorithm, particularly when configured with 10 Actors, exhibits marked superiority over other DRL algorithms like DDPG, PPO, and TD3 in terms of active steering control. It not only converges faster but also achieves higher final rewards. Furthermore, it demonstrates robust performance under various initial conditions. Therefore, Ape-X DDPG with 10 Actors will be the chosen algorithm for comparative studies in subsequent simulation experiments.

4.3. Comparative Analysis of Control Effects between Model-Based and Data-Based Algorithms

To validate the improvement in centering performance and curve steering ability of the DIRW vehicle by the Ape-X DDPG active controller, this study selected the data-driven classic DDPG algorithm and the model-based robust H∞ control [14] algorithm for comparison with our proposed Ape-X DDPG controller. The robust H∞ control, rooted in the μ-synthesis [39] method, is designed for MIMO systems characterized by uncertainties. In the design of the DIRW vehicle dynamics controller, wheel–rail creep contact and wheel conicity are considered as uncertainty parameters that vary within a certain range. Leveraging this robust H∞ control approach, an adaptive controller robust to the ever-changing wheel–rail contact dynamic characteristics is developed, relying on the D-K iteration process.

Three typical DIRW vehicle operating conditions from Table 1 were selected for detailed analysis. Given that leading IRW often plays a more critical guiding role than the rear IRW, the lateral displacement and yaw angle of the front IRW during curve running are generally significantly greater than those of the rear IRW. Therefore, in each operating condition, we compared the lateral displacement and yaw angle, wear number, and received steering control torque of the front IRW during running.

(1) Case 1: Operation on a straight track at 120 km/h.

The simulation results for Case 1 are shown in Figure 9. For the leading IRW wheelset, all three control algorithms avoid flange contact to prevent severe wheel–rail wear. Under the guidance of the Ape-X DDPG active steering controller, the maximum lateral displacement of the wheelset is 4.4 mm. In comparison to the 9.1 mm with the DDPG controller and 8.2 mm with the H∞ control controller, it demonstrates significant advantages in centering performance. With the controller proposed in this paper, the maximum attack angle of the front IRW under this operating condition is only 2.2 mrad. For comparison, the maximum wheelset attack angle under DDPG control is 5.5 mrad, and it is 4.6 mrad under H∞ control.

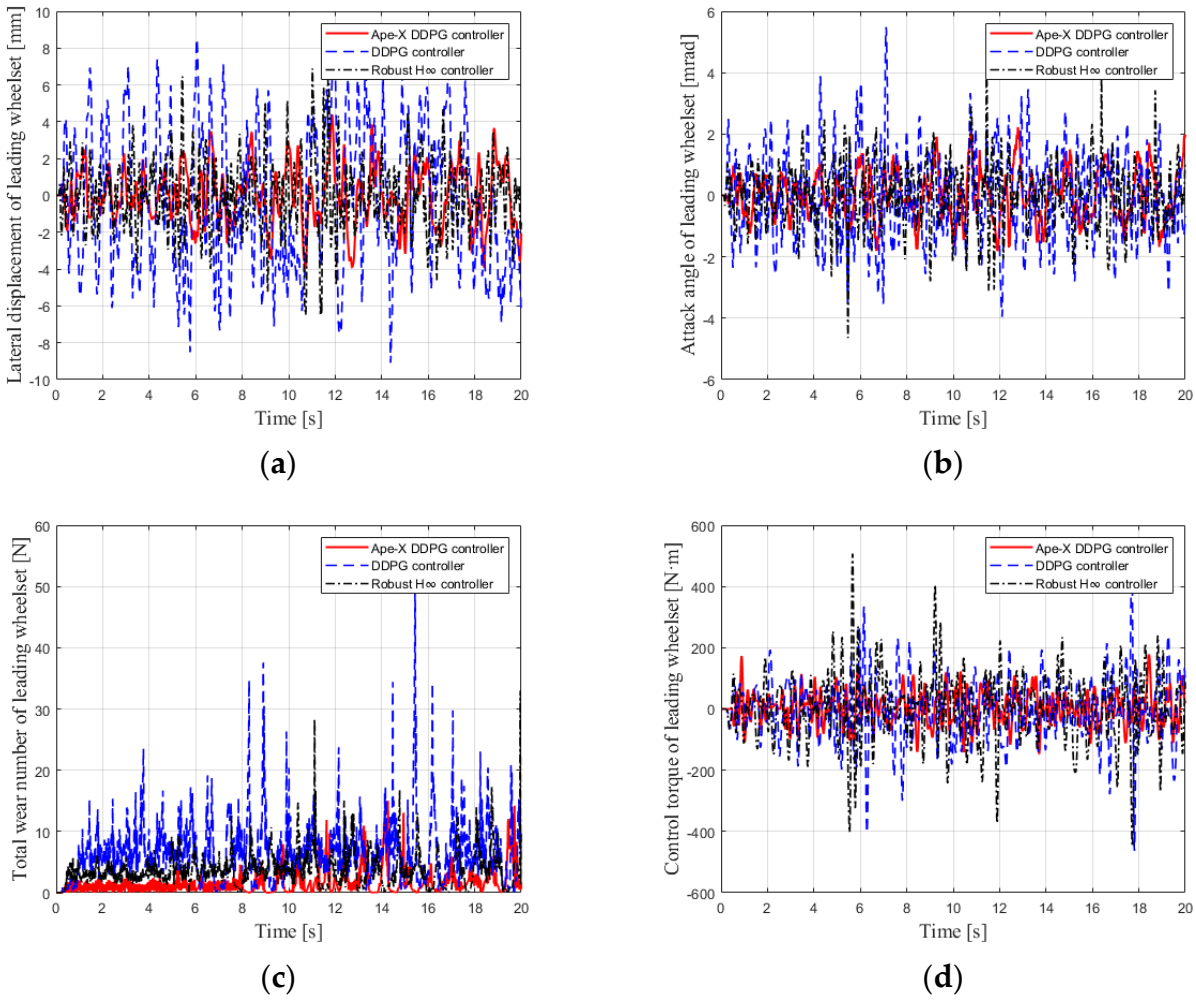

Figure 9. Case 1 simulation results (straight track, V_x = 120 km/h). (**a**) Lateral displacement. (**b**) Attack angle. (**c**) Wear number. (**d**) Control torque.

The fundamental purpose of active steering control is to reduce wear between the wheel and the rail. The simulation results indicate that by applying the active steering control proposed in this paper, the average wear number of the wheelset drops from 8.5 N under the DDPG controller and 4.6 N under the H∞ controller to 2.5 N. This means that Ape-X DDPG reduces wheel–rail wear by 71% and 46%, respectively, compared to the other two controllers. The maximum output torque of the Ape-X DDPG controller is 178 N·m, which is lower than the other controllers, indicating that the control torque adequately meets the design requirements.

(2) Case 2: Operation on a curved track with R_c = 70 m at 30 km/h.

The simulation results for Case 2 are shown in Figure 10. The results indicate that, in the case of a small curved line, the controller still maintains good curve guidance capability, avoiding wheel flange guidance and reducing wheel–rail wear. Under the Ape-X DDPG controller, the maximum lateral displacement of the leading wheelset is 5.1 mm, and the attack angle is 5.4 mrad. This is less than the 7.8 mm/8.5 mrad under the DDPG controller and 6.4 mm/6.5 mrad under the H∞ controller. The average wear number of our proposed controller on a small-radius curve is only 8.7 N, which is a 54% reduction compared to the DDPG controller and a 37% reduction compared to the H∞ controller. The maximum wear number is 28.6 N, which is less than the other two controllers used for comparison. The Ape-X DDPG controller has a maximum output torque of 725 N·m and an average torque of 431 N·m, reducing the requirements for control output peaks and control bandwidth compared to the other two controllers.

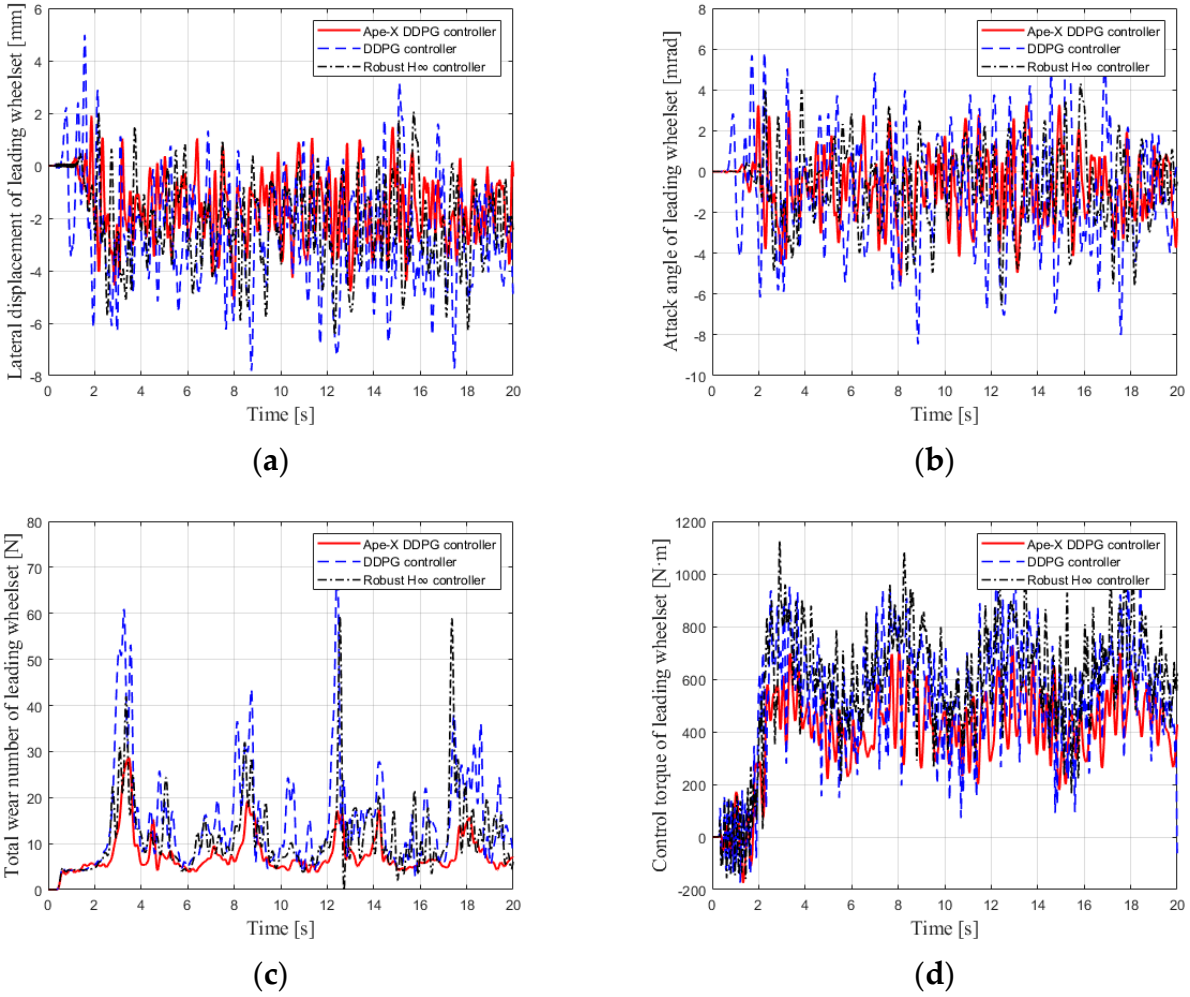

Figure 10. Case 2 simulation results (curved track with R_c = 70 m, V_x = 30 km/h). (**a**) Lateral displacement. (**b**) Attack angle. (**c**) Wear number. (**d**) Control torque.

(3) Case 3: Operation on a curved track with R_c = 250 m at 80 km/h.

The simulation results for Case 3 are shown in Figure 11. Due to the increase in curve radius, the control effect is better relative to Case 2. The Ape-X DDPG controller allows the IRW to stay almost radially positioned, with both the front wheelset's lateral displacement and attack angle kept at low levels. The maximum lateral displacement is 1.37 mm, and the maximum attack angle is 1.3 mrad. These values are superior to the 3.9 mm/4.3 mrad of the DDPG controller and the 2.5 mm/2.4 mrad of the H∞ controller. The total wear number remains at a low value, with an average wear number of 2.6 N for our proposed controller. The peak control torque is 318 N·m, only a small fraction of the motor's maximum output torque. Additionally, thanks to the improved experience replay strategy, the output of the Ape-X DDPG controller is smoother than that of the DDPG controller.

Comparing the control effects of different controllers under various conditions, the Ape-X DDPG controller significantly improves the IRW's straight-track centering performance and curve steering capability and can greatly reduce wheel–rail wear. At the same time, the controller's output torque is stable and less than the motor's peak torque.

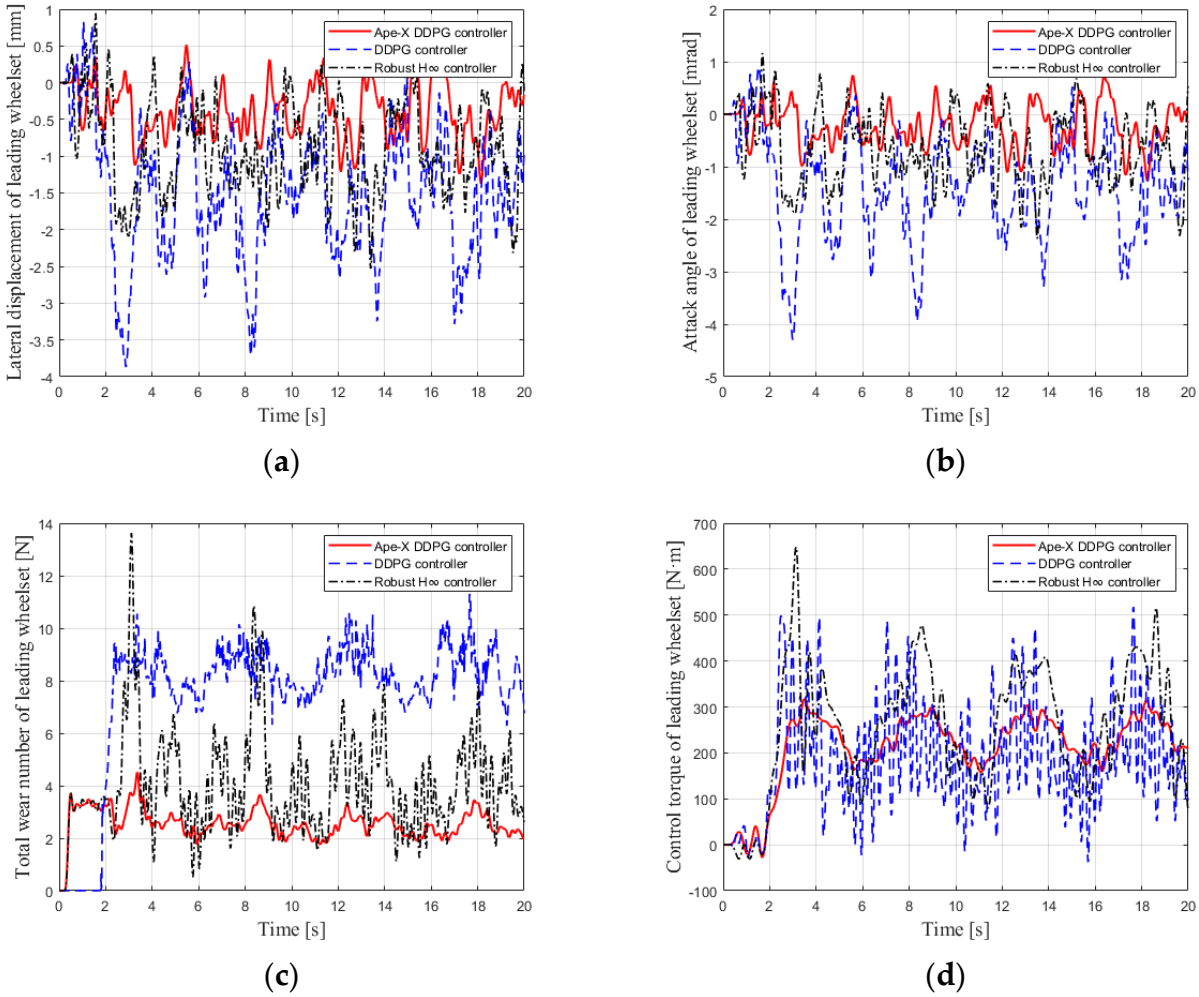

Figure 11. Case 3 simulation results (curved track with R_c = 250 m, V_x = 80 km/h). (**a**) Lateral displacement. (**b**) Attack angle. (**c**) Wear number. (**d**) Control torque.

5. Scale Model Experiments

5.1. Scale DIRW Vehicle

To verify the performance of the Ape-X DDPG controller in a real vehicle operating environment, we designed and manufactured a 1:5 scale IRW model for experimental verification, as shown in Figure 12. The DRL-based controller is trained using the digital twin dynamics model of the scaled DIRW vehicle and is deployed on the test vehicle. This scaled model comprises a bogie and a carbody and two subframes, each equipped with IRWs. The primary and secondary suspensions are steel springs and rubber bushings, respectively, and they support the mass of the upper carbody. The left and right wheels are connected to the reduction gearboxes via short axles. Four 600 W servo motors transfer the active steering control torque to the IRWs through the reduction gearboxes. Each drive system and IRW are located on the same subframe, ensuring synchronized movement in lateral, yaw, and other directions.

The block diagram of the scaled model's control system is shown in Figure 13.

The scaled model is powered by an AC source, converting the input voltage into a three-phase AC high-voltage circuit for the drive motor via an inverter, as well as a low-voltage for the controller and sensors. The active steering controller's hardware is based on the NVIDIA Jetson Nano edge computing development board. Once the controller is trained using reinforcement learning algorithms, it is exported in the Open Neural Network Exchange (ONNX) format, which contains the neural network structure

and weight parameters. This exported model is subsequently transformed into CUDA executable code using TensorRT and is then deployed on the Jetson Nano.

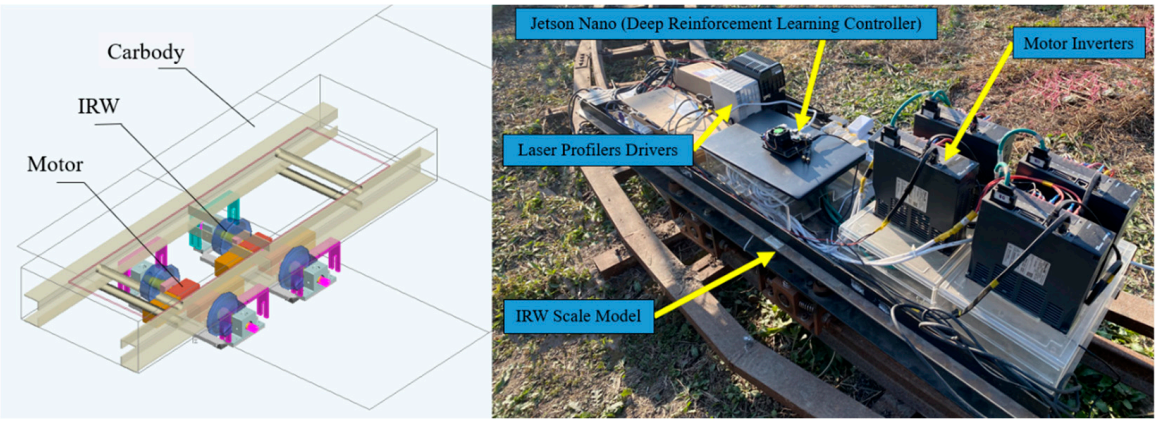

Figure 12. Scaled model vehicle and digital twin model.

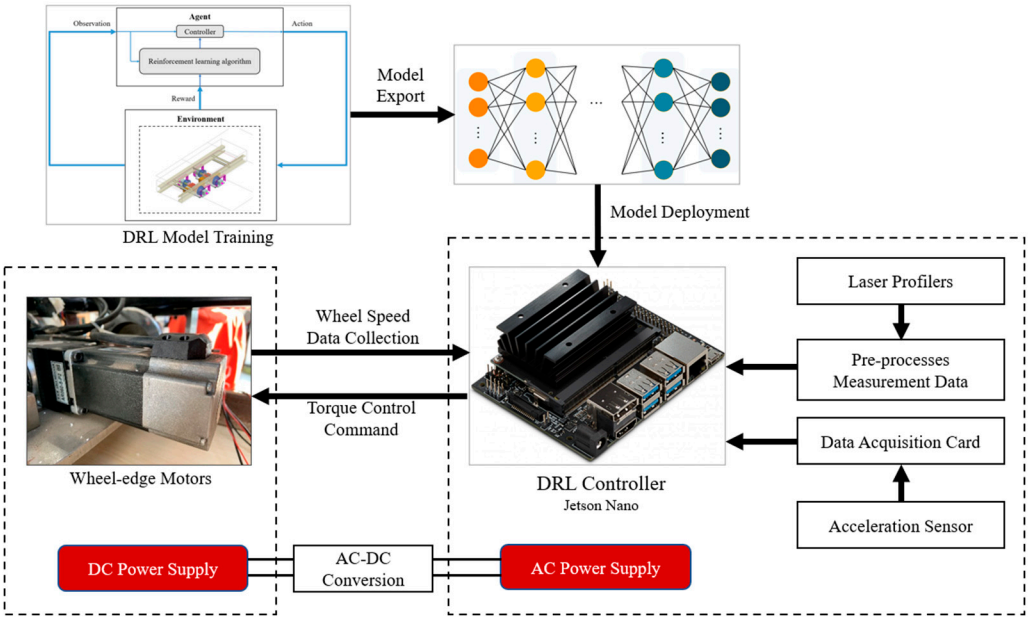

Figure 13. Control architecture of the active guidance test in the scaled model.

5.2. Acquisition of Sensor Signals

The wheel–rail relative position needed by the active steering controller, including the lateral displacement and yaw angle of the IRW relative to the track, is obtained through laser profilers mounted on each IRW. Compared to point laser displacement sensors and eddy current displacement sensors, the laser profilers offer a broader measurement range and higher accuracy. Even when there is significant lateral displacement and attack angle between the wheel and rail, the laser profilers can still ensure the measurement beam projects on the rail surface, avoiding the loss of reflected laser measurement points which could lead to control failure.

The laser profilers are mounted on the subframes, located both at the front and back of each IRW, with the laser emitting vertically downwards towards the top of the rail, as depicted in Figure 14. Each set of laser profilers obtains the two-dimensional coordinates of the rail's outer contour during vehicle operation. By fitting these coordinates with a quartic curve and comparing the peak coordinates of the contour to the baseline (when the lateral displacement is zero), the current lateral displacement between the laser profiler and rail can be calculated.

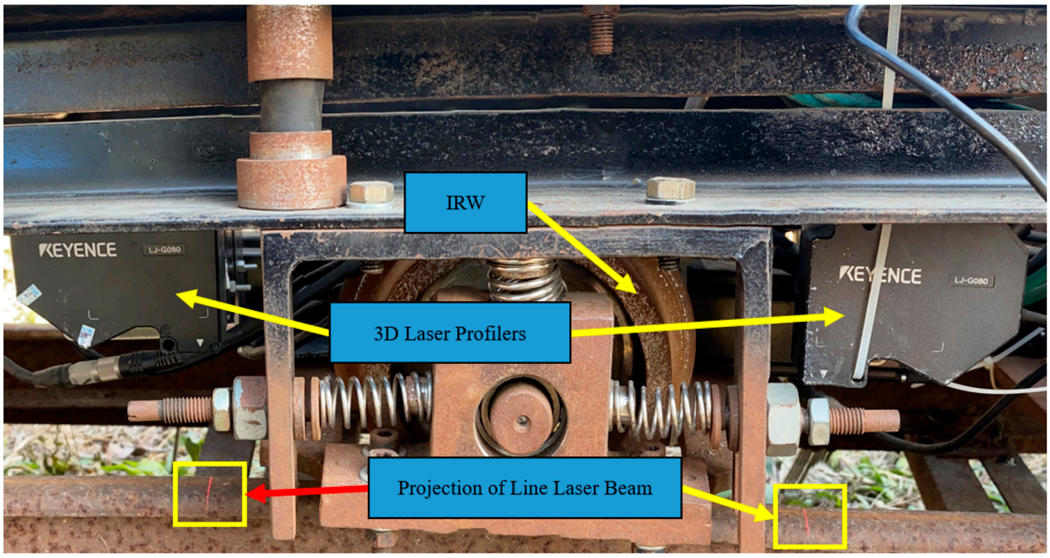

Figure 14. Measurement of the wheel–rail relative position based on laser profile sensors.

The wheelset's lateral displacement and yaw angle can be calculated using Equations (30) and (31).

$$y_{wi} = -\frac{y_{s1,i} + y_{s2,i}}{2} \tag{30}$$

$$\psi_{wi} = -\frac{y_{s1,i} - y_{s2,i}}{L_s} \tag{31}$$

where $y_{s1,i}$ and $y_{s2,i}$ are the lateral displacement values of the two sets of laser profilers at each IRW, with subscript i = 1,2 indicating the front and rear IRWs, respectively. L_s represents the longitudinal interval distance between the two sets of laser profilers at each IRW.

5.3. Experimental Results

The active control experiment of the DIRW scaled model was conducted on a 1:5 railway test track, which includes a 10 m straight track and a 10 m curved track with a radius of 15 m. The vehicle runs through the test track at a speed of 2 m/s, and each experiment of the scaled model lasts for 10 s. We deployed four control methods: passive travel, H∞ control, DDPG controller, and Ape-X DDPG controller.

The experimental results of the scaled model are shown in Figure 15, displaying data for the front IRW's lateral displacement, yaw angle, and steering torque. Without active steering control in the IRW, the maximum lateral displacement was 6.9 mm on the straight track and 9.8 mm on the curved track due to wheel flange guiding, causing intense wheel–rail wear. Using the Ape-X DDPG controller proposed in this paper, the lateral displacement of the IRW can be controlled within ±0.5 mm on the straight track and ±2 mm on the curved track. The maximum control torque occurred at the transition of the curve and straight track, at 5.3 N·m, which is below the motor's peak output torque. Relatively speaking, both the DDPG-based controller and the H∞-based controller had greater maximum lateral displacements and attack angles during operation compared to our proposed controller. Additionally, the Ape-X DDPG controller responds smoothly, reducing the dynamic response requirements for the motor. Therefore, we conclude that the Ape-X DDPG controller effectively meets the design requirements.

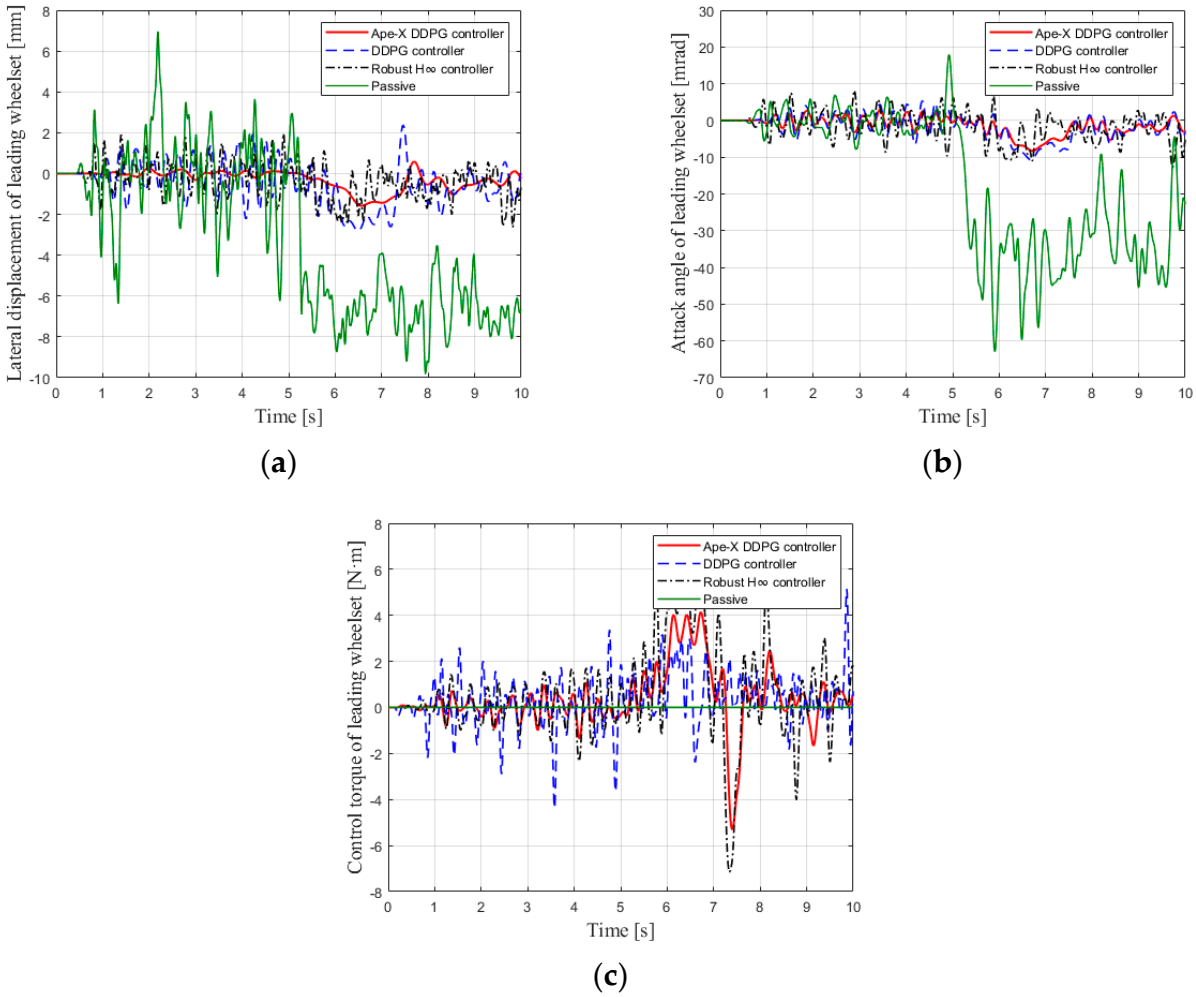

Figure 15. Experimental results of the scaled model. (**a**) Lateral displacement. (**b**) Attack angle. (**c**) Control torque.

6. Conclusions

This study introduced an active steering controller for DIRW vehicles based on the DRL algorithm, effectively enhancing the guidance capability of IRWs and reducing wheel–rail wear. Given the complex wheel–rail contact dynamics in the railway vehicle system, traditional design methodologies struggle to effectively design controllers for highly nonlinear systems. However, reinforcement learning, through constant trial and error and maximizing the reward function, combined with the deep neural network's capability to fit nonlinear systems, allows the controller to adaptively train and attain optimal strategies. We integrated DPER with the DDPG algorithm in our DRL controller's training and interaction, leading to the implementation of the Ape-X DDPG algorithm. This was to address the classic DDPG algorithm's challenges, such as low sample efficiency and propensity to get stuck in local optima. Using the multibody dynamics theory, we established a nonlinear DIRW vehicle model and trained the Ape-X DDPG controller, and the control efficacy was validated under multiple vehicle operational conditions. Both simulation results and scaled model experiments demonstrate that our control strategy outperforms previous methods. With the intervention of the Ape-X DDPG active guidance controller, the IRWs have enhanced straight-line centering performance and curve guidance ability while diminishing wheel–rail wear. This research underscores the significance of DRL-based controllers in terms of efficacy and feasibility when the large-scale application of DIRW's active steering control is considered.

Author Contributions: Conceptualization, Z.L.; Methodology, Z.L.; Validation, Z.L.; Formal analysis, J.W.; Investigation, J.W.; Resources, Z.W.; Data curation, Z.W.; Writing—original draft, Z.L.; Writing—review & editing, J.W. and Z.W. 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 that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Parameters of the DIRW vehicle.

Dynamic Parameters	Value	Unit
Wheel base	2.0	m
Lateral length of primary suspension	1.08	m
Longitudinal length of primary suspension	0.39	m
Lateral length of secondary suspension	1.80	m
Longitudinal length of secondary suspension	0.50	m
Mass of carbody	9000	kg
Pitch moment of inertia of carbody	80,000	$kg \cdot m^2$
Yaw moment of inertia of carbody	80,000	$kg \cdot m^2$
Roll moment of inertia of carbody	10,000	$kg \cdot m^2$
Mass of bogie	1600	kg
Pitch moment of inertia of bogie	900	$kg \cdot m^2$
Yaw moment of inertia of bogie	1300	$kg \cdot m^2$
Roll moment of inertia of bogie	1500	$kg \cdot m^2$
Mass of IRW	1250	kg
Pitch moment of inertia per wheel	30	$kg \cdot m^2$
Yaw moment of inertia of IRW	600	$kg \cdot m^2$
Primary longitudinal stiffness per axle box	600	$kN \cdot m^{-1}$
Primary lateral stiffness per axle box	8000	$kN \cdot m^{-1}$
Primary vertical stiffness per axle box	15,000	$kN \cdot m^{-1}$
Primary longitudinal damping per axle box	100	$kN \cdot m \cdot s^{-1}$
Primary lateral damping per axle box	500	$kN \cdot m \cdot s^{-1}$
Primary vertical damping per axle box	800	$kN \cdot m \cdot s^{-1}$
Secondary longitudinal stiffness per side	200	$kN \cdot m^{-1}$
Secondary lateral stiffness per side	200	$kN \cdot m^{-1}$
Secondary vertical stiffness per side	600	$kN \cdot m^{-1}$
Secondary longitudinal damping side	50	$kN \cdot m \cdot s^{-1}$
Secondary lateral damping per side	50	$kN \cdot m \cdot s^{-1}$
Secondary vertical damping per side	50	$kN \cdot m \cdot s^{-1}$

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. Goodall, R.; Mei, T.X. Mechatronic strategies for controlling railway wheelsets with independently rotating wheels. In Proceedings of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Proceedings (Cat. No.01TH8556), Como, Italy, 8–12 July 2001; Volume 1, pp. 225–230.
3. Oh, Y.J.; Liu, H.-C.; Cho, S.; Won, J.H.; Lee, H.; Lee, J. Design, modeling, and analysis of a railway traction motor with independently rotating wheelsets. *IEEE Trans. Magn.* **2018**, *54*, 8205305. [CrossRef]
4. Pérez, J.; Busturia, J.M.; Mei, T.X.; Vinolas, J. Combined active steering and traction for mechatronic bogie vehicles with independently rotating wheels. *Annu. Rev. Control* **2004**, *28*, 207–217. [CrossRef]
5. Mei, T.X.; Li, H.; Goodall, R.M.; Wickens, A.H. Dynamics and control assessment of rail vehicles using permanent magnet wheel motors. *Veh. Syst. Dyn.* **2002**, *37* (Suppl. 1), 326–337. [CrossRef]
6. Chudzikiewicz, A.; Gerlici, J.; Sowinska, M.; Sowinska, M.; Stelmach AWawrzyński, W. Modeling and simulation of a control system of wheels of wheelset. *Arch. Transp.* **2020**, *55*, 73–83. [CrossRef]

7. 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]
8. Mei, T.X.; Qu, K.W.; Li, H. Control of wheel motors for the provision of traction and steering of railway vehicles. *IET Power Electron.* **2014**, *7*, 2279–2287. [CrossRef]
9. Heckmann, A.; Schwarz, C.; Keck, A.; Bünte, T. Nonlinear observer design for guidance and traction of railway vehicles. 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: Cham, Switzerland, 2020.
10. 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]
11. 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]
12. Kurzeck, B.; Heckmann, A.; Wesseler, C.; Rapp, M. Mechatronic track guidance on disturbed track: The trade-off between actuator performance and wheel wear. *Veh. Syst. Dyn.* **2014**, *52* (Suppl. 1), 109–124. [CrossRef]
13. Grether, G. Dynamics of a running gear with IRWs on curved tracks for a robust control development. *PAMM* **2017**, *17*, 797–798. [CrossRef]
14. Lu, Z.; Yang, Z.; Huang, Q.; Wang, X. Robust active guidance control using the μ-synthesis method for a tramcar with independently rotating wheelsets. *Proc. Inst. Mech. Eng. Part F J. Rail Rapid Transit* **2019**, *233*, 33–48. [CrossRef]
15. Yang, Z.; Lu, Z.; Sun, X.; Zou, J.; Wan, H.; Yang, M.; Zhang, H. Robust LPV-H∞ control for active steering of tram with independently rotating wheels. *Adv. Mech. Eng.* **2022**, *14*, 16878132221130574. [CrossRef]
16. 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]
17. 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 Proceedings of the IAVSD International Symposium on Dynamics of Vehicles on Roads and Tracks, Saint Petersburg, Russia, 17–19 August 2021; Springer International Publishing: Cham, Switzerland, 2021.
18. Ramos-Fernández, J.C.; López-Morales, V.; Márquez-Vera, M.A.; Pérez JM, X.; Suarez-Cansino, J. Neuro-fuzzy modelling and stable PD controller for angular position in steering systems. *Int. J. Automot. Technol.* **2021**, *22*, 1495–1503. [CrossRef]
19. 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]
20. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018; p. 342.
21. Deng, H.; Zhao, Y.; Nguyen, A.T.; Huang, C. Fault-tolerant predictive control with deep-reinforcement-learning-based torque distribution for four in-wheel motor drive electric vehicles. *IEEE/ASME Trans. Mechatron.* **2023**, *28*, 668–680. [CrossRef]
22. Wei, H.; Zhao, W.; Ai, Q.; Zhang, Y.; Huang, T. Deep reinforcement learning based active safety control for distributed drive electric vehicles. *IET Intell. Transp. Syst.* **2022**, *16*, 813–824. [CrossRef]
23. Albarella, N.; Lui, D.G.; Petrillo, A.; Santini, S. A Hybrid Deep Reinforcement Learning and Optimal Control Architecture for Autonomous Highway Driving. *Energies* **2023**, *16*, 3490. [CrossRef]
24. Cheng, Y.; Hu, X.; Chen, K.; Yu, X.; Luo, Y. Online longitudinal trajectory planning for connected and autonomous vehicles in mixed traffic flow with deep reinforcement learning approach. *J. Intell. Transp. Syst.* **2023**, *27*, 396–410. [CrossRef]
25. Wang, H.; Han, Z.; Liu, Z.; Wu, Y. Deep reinforcement learning based active pantograph control strategy in high-speed railway. *IEEE Trans. Veh. Technol.* **2022**, *72*, 227–238. [CrossRef]
26. Zhu, F.; Yang, Z.; Lin, F.; Xin, Y. Decentralized cooperative control of multiple energy storage systems in urban railway based on multiagent deep reinforcement learning. *IEEE Trans. Power Electron.* **2020**, *35*, 9368–9379. [CrossRef]
27. Sresakoolchai, J.; Kaewunruen, S. Railway infrastructure maintenance efficiency improvement using deep reinforcement learning integrated with digital twin based on track geometry and component defects. *Sci. Rep.* **2023**, *13*, 2439. [CrossRef]
28. Farazi, N.P.; Zou, B.; Ahamed, T.; Barua, L. Deep reinforcement learning in transportation research: A review. *Transp. Res. Interdiscip. Perspect.* **2021**, *11*, 100425.
29. Lee, M.F.R.; Yusuf, S.H. Mobile robot navigation using deep reinforcement learning. *Processes* **2022**, *10*, 2748. [CrossRef]
30. Moriya, N.; Shigemune, H.; Sawada, H. A robotic wheel locally transforming its diameters and the reinforcement learning for robust locomotion. *Int. J. Mechatron. Autom.* **2022**, *9*, 22–31. [CrossRef]
31. Li, Y. Deep reinforcement learning: An overview. *arXiv* **2017**, arXiv:1701.07274.
32. 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* **2015**, arXiv:1509.02971.
33. Moritz, P.; Nishihara, R.; Wang, S.; Tumanov, A.; Liaw, R.; Liang, E.; Elibol, M.; Yang, Z.; Paul, W.; Jordan, M.I.; et al. Ray: A distributed framework for emerging AI applications. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA, USA, 8–10 October 2018.
34. Liang, E.; Liaw, R.; Nishihara, R.; Moritz, P.; Fox, R.; Goldberg, K.; Gonzalez, J.; Jordan, M.; Stoica, I. RLlib: Abstractions for distributed reinforcement learning. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.

35. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; Volume 32.
36. Kingma, D.P.; Jimmy, B. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
37. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
38. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.
39. Skogestad, S.; Postlethwaite, I. Multivariable feedback control: Analysis and design. In *Multivariable Feedback Control: Analysis and Design*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2005.