Learning to Drift With Individual Wheel Drive: Maneuvering Autonomous Vehicle at the Handling Limits

Yihan Zhou, Yiwen Lu, Bo Yang, Jiayun Li, and Yilin Mo, *Member, IEEE*

***Abstract*—Drifting, characterized by controlled vehicle motion at high sideslip angles, is crucial for safely handling emergency scenarios at the friction limits. While recent reinforcement learning approaches show promise for drifting control, they struggle with the significant simulation-to-reality gap, as policies that perform well in simulation often fail when transferred to physical systems. In this letter, we present a reinforcement learning framework with GPU-accelerated parallel simulation and systematic domain randomization that effectively bridges the gap. The proposed approach is validated on both simulation and a custom-designed and open-sourced 1/10 scale Individual Wheel Drive (IWD) RC car platform featuring independent wheel speed control. Experiments across various scenarios from steady-state circular drifting to direction transitions and variable-curvature path following demonstrate that our approach achieves precise trajectory tracking while maintaining controlled sideslip angles throughout complex maneuvers in both simulated and real-world environments.**

***Index Terms*—Motion control, reinforcement learning, autonomous drifting.**

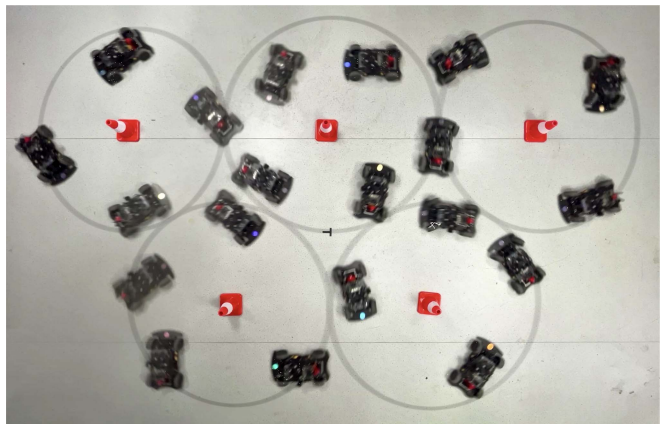

Fig. 1. Top-down view of drifting along the path inspired by the Olympic rings. Five tangent circles form the reference path marked by gray curves, with red cones marking the centers. Multiple car positions captured in this long-exposure photograph demonstrate consecutive drifting maneuvers.

I. Introduction

In the realm of motorsport, high-speed cornering with significant sideslip angles, commonly referred to as drifting, represents an attractive yet challenging skill mastered by professional drivers [1]. It demands not only a profound understanding of vehicle dynamics but also the agility to respond instantaneously to changing environmental conditions. Beyond serving as a thrilling spectacle, drifting constitutes a fundamental technique for trajectory tracking at the limits of vehicle handling [2]. As autonomous vehicles are expected to handle extreme scenarios with capabilities matching or exceeding those of human drivers, research on drifting control has substantial implications for the safety of autonomous vehicles under critical conditions such as sudden changes in road friction or emergency obstacle avoidance [3], [4], [5].

Received 29 March 2025; accepted 22 July 2025. Date of publication 1 August 2025; date of current version 14 August 2025. This article was recommended for publication by Associate Editor R. Camoriano and Editor J. Kober upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant 62461160313, Grant 62273196, and Grant 62192752, and in part by BNRist Project under Grant BNR2024TD03003. *(Corresponding author: Yilin Mo.)*

The authors are with the Department of Automation and BNRist, Tsinghua University, Beijing 100084, China (e-mail: zhouyh23@mails.tsinghua.edu.cn; luyw20@mails.tsinghua.edu.cn; yang-b21@mails.tsinghua.edu.cn; lijiayun22@mails.tsinghua.edu.cn; ylmo@tsinghua.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3595024, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3595024

Several works have explored autonomous drifting through various approaches. Model-based methods leverage vehicle dynamics principles [3], [6], [7], typically identifying vehicle parameters and then formulating drifting as optimal control problems [8] or designing modular control architectures [3]. While these approaches provide interpretability and theoretical guarantees, they face practical limitations including computational complexity and sensitivity to model uncertainties.

Learning-based methods have emerged as a complementary paradigm. Recent work by Djeumou et al. [9] demonstrated how learned dynamics can enhance control by incorporating a diffusion model into an MPC framework, achieving robust drifting across varying vehicles and conditions. In parallel, reinforcement learning (RL) represents another promising branch, with early work by Cutler et al. [10] demonstrating feasibility for circular drift stabilization on small-scale platforms, though with limited trajectory flexibility. Subsequent research by Cai et al. [1] and Domberg et al. [11] extended RL capabilities to high-speed racing scenarios in simulation. However, these approaches encounter significant limitations that hinder practical deployment. The simulation-to-reality gap remains particularly pronounced in drifting control, as evidenced in [11] where policies performing well in simulation failed to complete maneuvers when transferred to physical platforms. Additionally, conventional RL implementations often require extensive training time, with approaches like [1] requiring over 11 hours of training time, extending the development and tuning cycle of the algorithm.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Consequently, there exists a need for a framework that can both accelerate the training process and effectively bridge the sim-to-real gap, enabling robust drifting control across diverse trajectory patterns.

Research platforms for drifting control have predominantly utilized either rear-wheel drive (RWD) [12], [13] or all-wheel drive (AWD) [4], [5], [14] with coupled wheel speeds. The advancement of electric vehicles has accelerated the development of Individual Wheel Drive (IWD) architectures that enable independent control of each wheel's speed and torque, expanding the available control authority compared to conventional drivetrains. However, despite their increasing prevalence in the automotive industry, research platforms for developing and validating IWD control strategies remain limited.

In this letter, we present an integrated hardware-software solution that addresses the aforementioned limitations in autonomous drifting control. From an algorithmic perspective, we propose a reinforcement learning framework that successfully bridges the simulation-to-reality gap through systematic domain randomization techniques. This framework leverages custom GPU-accelerated parallel simulation, reducing training time from hours to minutes. From the hardware perspective, our methodology is implemented on a custom-designed Individual Wheel Drive (IWD) platform that we have developed and open-sourced, delivering enhanced control authority through independent wheel actuation. The integration enables the exploration of complex drift maneuvers while maintaining computational efficiency suitable for embedded systems. Through experimental validation, we demonstrate that the approach achieves precise drifting control across diverse trajectories, including challenging scenarios such as eight-shaped pattern and variable-curvature track, with consistent performance in both simulation and real-world environments.

Contributions: This work advances autonomous drifting control through: (1) A reinforcement learning approach for autonomous drifting control on IWD vehicles that enables effective sim-to-real transfer without real-world fine-tuning; (2) An open-source IWD platform implementation providing enhanced maneuverability through independent wheel speed control with precise torque vectoring capabilities; and (3) Experimental demonstration of challenging autonomous drifting maneuvers including direction reversals and variable-curvature tracking. The complete implementation is publicly available to promote reproducibility and further research in this domain.

The remainder of this letter is organized as follows: Section II reviews the related work in autonomous drifting. Section III presents the platform design and system modeling, including the IWD RC car platform, vehicle dynamics model, and the GPU-accelerated parallel car simulator. Section IV formulates the consecutive drifting problem and details our method for agile maneuvering. Section V presents experimental results, and Section VI concludes the letter.

II. Related Works

A. Model-Based Approaches for Drifting Control

Autonomous drifting has been extensively studied using model-based approaches. Early works by Voser et al. [6] and Hindiyeh and Gerdes [7] established fundamental frameworks by analyzing equilibrium states and developing nested-loop control structures for drift stabilization. Goh et al. [3], [12], [15] extended these concepts to arbitrary path tracking by incorporating curvilinear coordinates and nonlinear model inversion. Various optimal control techniques have been applied to drifting, including MPC [16], NMPC [8], [13], [17], and LQR [18], [19]. While these approaches offer theoretical guarantees, they face practical challenges due to their computational complexity and reliance on accurate system identification and tire modeling, though some limitations can be addressed through expert knowledge-based methods [20].

Parameter	Value
Vehicle mass	4.84 kg
Wheelbase	0.35 m
Track width	0.26 m
Wheel radius	0.0565 m
Max steering angle	$\pm 26.35^\circ$

Fig. 2. Xcar platform and key specifications.

B. Learning-Based Methods for Drifting Control

Recent work on learning-based approaches for autonomous drifting has shown significant progress through two main directions. Djeumou et al. enhanced model-based control frameworks by integrating the learned tire [21] or vehicle model [9], improving robustness to varying conditions. Meanwhile, reinforcement learning represents an effective alternative to model-based control approaches. Early explorations using model-based RL frameworks like PILCO [22] demonstrated initial feasibility of learning-based methods for drift control in both simulation [23] and on small-scale RC cars [10], though their applications were limited to circular trajectories. Recent advances in model-free RL have tackled more complex scenarios: Cai et al. [1] developed a SAC-based approach combining professional driver demonstrations for high-speed drifting across various race tracks in the CARLA simulator, while Domberg et al. [11] proposed a PPO-based method for arbitrary trajectory tracking that maps vehicle states and path information directly to control commands. However, these methods face two main challenges. First, the simulation-to-reality gap limits the real-world performance, particularly on complex trajectories, leading to degraded performance when transferring from simulation to physical systems [11]. Second, the training efficiency remains a practical concern, with training times exceeding 11 hours in [1], which slows the iterative refinement process.

III. Platform and System Modeling

To facilitate research in agile vehicle maneuvering, we introduce an open-source 1/10 scale RC car testbed featuring independent wheel drive (IWD) capabilities. This section details our platform design and develops its mathematical model which is then leveraged in a GPU-accelerated parallel simulation environment to enable efficient reinforcement learning.

A. Individual Wheel Drive RC Car

Our experimental platform, Xcar (shown in Fig. 2), draws inspiration from several established research vehicles including the Berkeley autonomous race car [24], MIT RACECAR [25],

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

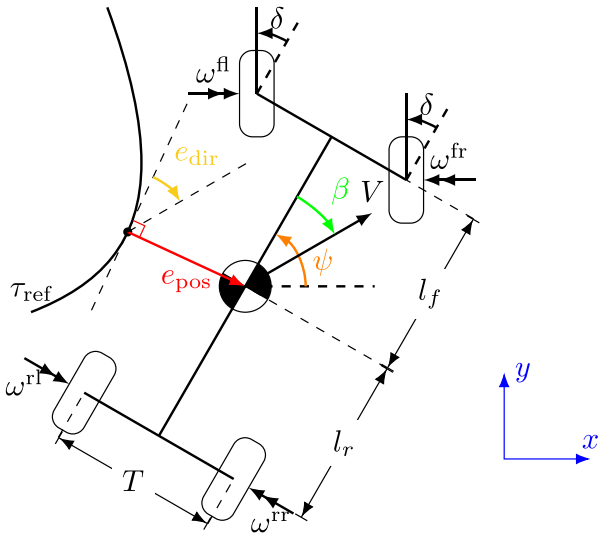

Fig. 3. Illustration of IWD vehicle model, the reference trajectory τ_{ref} and the definition of errors.

MuSHR [26], RoSCAR [27] and F1TENTH [28]. Unlike previous platforms, Xcar incorporates a distinct drivetrain configuration with four independent motors powered by a 4S LiPo battery, each controlled by a VESC Mini V6.7 motor controller, which provides additional degrees of freedom for vehicle dynamics control enabling precise torque vectoring and enhanced yaw moment control. The onboard computational system utilizes an NVIDIA Jetson Orin Nano, serving as the ROS main node and executing the control policy. To establish accurate localization and state estimation, the car is equipped with a motion capture system. Using this high-precision pose data, a Kalman filter is implemented to estimate linear and angular velocities and accelerations. A detailed list of hardware components is available at https://zhou-yh19.github.io/xcar-hardware.

B. Modeling

To balance model fidelity with computational efficiency, we adopt a simplified vehicle dynamics model based on 7-DOF models [18], [29] by constraining motion to a 2D plane and omitting motor dynamics, while retaining the essential Pacejka tire model.

The IWD vehicle's state vector $\mathbf{x} \in \mathcal{X}$ consists of six components representing its position and motion (illustrated in Fig. 3):

$$\mathbf{x} = [x, y, \psi, \dot{x}, \dot{y}, \dot{\psi}], \tag{1}$$

where (x, y) denotes the vehicle's position in global coordinates, ψ represents the heading angle, and $(\dot{x}, \dot{y}, \dot{\psi})$ are their respective time derivatives.

The control input vector $\mathbf{u} \in \mathcal{U}$ encompasses the steering angle and individual wheel angular velocities:

$$\mathbf{u} = [\delta, \omega^{\mathrm{fl}}, \omega^{\mathrm{fr}}, \omega^{\mathrm{rl}}, \omega^{\mathrm{rr}}], \tag{2}$$

where δ represents the steering angle and ω^{ij} denotes the angular velocity of each wheel (subscripts: f/r - front/rear, l/r - left/right).

The dynamics of the IWD vehicle are governed by the following equations:

$$\begin{aligned}
m\ddot{x} &= \cos(\delta+\psi)F_x^{\mathrm{f}} - \sin(\delta+\psi)F_y^{\mathrm{f}} + \cos\psi F_x^{\mathrm{r}} - \sin\psi F_y^{\mathrm{r}} \\
m\ddot{y} &= \sin(\delta+\psi)F_x^{\mathrm{f}} + \cos(\delta+\psi)F_y^{\mathrm{f}} + \sin\psi F_x^{\mathrm{r}} + \cos\psi F_y^{\mathrm{r}} \\
I_z\ddot{\psi} &= [\cos\delta F_y^{\mathrm{f}} + \sin\delta F_x^{\mathrm{f}}]l_f - F_y^{\mathrm{r}} l_r \\
&\quad + [\cos\delta\Delta F_x^{\mathrm{f}} - \sin\delta\Delta F_y^{\mathrm{f}} + \Delta F_x^{\mathrm{r}}]T/2
\end{aligned}$$

where m is the vehicle mass, I_z is the moment of inertia, l_f and l_r are the distances from the center of mass to the front and rear axles, T is the track width. The force terms are defined as: $F_{x/y}^{\mathrm{f}} = F_{x/y}^{\mathrm{fl}} + F_{x/y}^{\mathrm{fr}}$ (front axle forces), $F_{x/y}^{\mathrm{r}} = F_{x/y}^{\mathrm{rl}} + F_{x/y}^{\mathrm{rr}}$ (rear axle forces), $\Delta F_{x/y}^{\mathrm{f}} = F_{x/y}^{\mathrm{fr}} - F_{x/y}^{\mathrm{fl}}$ (front differential forces), and $\Delta F_{x/y}^{\mathrm{r}} = F_{x/y}^{\mathrm{rr}} - F_{x/y}^{\mathrm{rl}}$ (rear differential forces), with $F_{x/y}^{\mathrm{ij}}$ representing the longitudinal/lateral tire forces at each wheel. These tire forces are derived from the control inputs through the Pacejka tire model based on wheel slip and vertical loads. Our implementation includes longitudinal load transfer but assumes equal lateral distribution.

For analyzing and controlling drift maneuvers, three coordinate-free variables play a fundamental role:

- The sideslip angle $\beta = \tan^{-1}(\frac{\dot{y}}{\dot{x}}) - \psi$.
- The yaw rate r (equivalent to $\dot{\psi}$).
- The velocity magnitude $V = \sqrt{\dot{x}^2 + \dot{y}^2}$.

These variables relate to the vehicle motion through the kinematic equations:

$$\dot{x} = V\cos(\beta+\psi), \quad \dot{y} = V\sin(\beta+\psi), \quad \dot{\psi} = r. \tag{3}$$

Therefore, $[r, \beta, V]$ can be viewed as a rotation- and translation-invariant description of the vehicle motion, and therefore particularly important for learning algorithm design.

C. GPU-Accelerated Parallel Car Simulation

Based on the vehicle model described above, we develop a GPU-accelerated car simulator inspired by NVIDIA Isaac Gym [30] to facilitate efficient reinforcement learning training. The simulator supports various drivetrain configurations including IWD, RWD, and AWD. Compared to traditional CPU-based simulators like CarSim and F1TENTH [31], our simulator offers two key advantages: (1) enabling massive parallelization on a single machine to meet the data requirements of deep reinforcement learning, and (2) eliminating CPU-GPU data transfer overhead during neural network training.

The simulator is implemented in PyTorch [32] using the Euler integration method. Running on an NVIDIA GeForce RTX 3080 GPU with a 0.01 s time step, it achieves parallel simulation of 10^6 car instances at 30 steps per second, corresponding to a speedup factor of 3×10^5 relative to real-time.

This high-throughput approach prioritizes rapid data generation over absolute fidelity, acknowledging that even high-fidelity models face reality gaps due to parametric uncertainties. The task of bridging this gap is addressed through domain randomization techniques during reinforcement learning training, as detailed in Section IV-C.

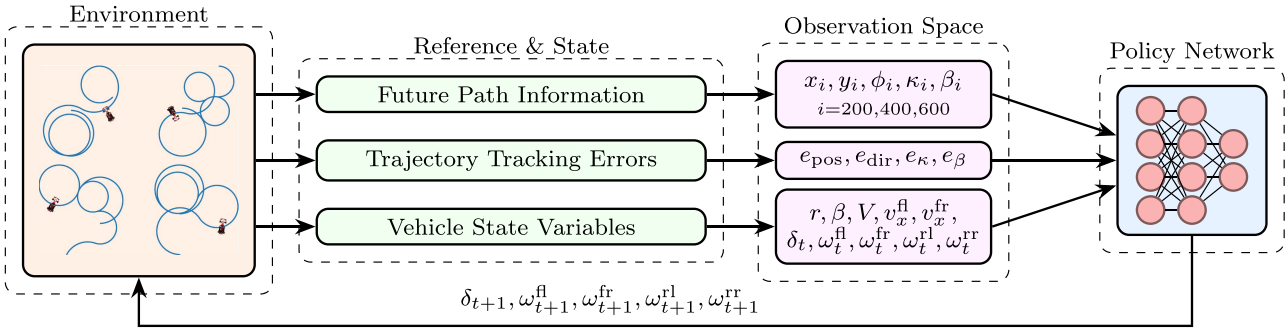

Fig. 4. Overview of the reinforcement learning framework for autonomous drifting: The GPU-accelerated parallel simulation environment accelerates computation with multiple vehicle instances across various trajectories. For coordinate-free policy learning, the observation space is constructed from translation and rotation invariant quantities including vehicle velocities and slip angles, along with tracking errors and future waypoints computed in a curvilinear coordinate system aligned with the reference trajectory, leading to end-to-end control of individual wheel speeds and steering angles.

IV. Reinforcement Learning for Maneuvering

In this section, we present a reinforcement learning framework (Fig. 4) that enables autonomous vehicles to perform agile drifting maneuvers at the handling limits while addressing the fundamental challenge of sim-to-real transfer. The framework trains a policy network that controls individual wheel speeds and steering angle for precise trajectory tracking and stable drift behavior. We design systematic domain randomization strategies across reference trajectories, initial conditions, tire model parameters, and dynamic disturbances to bridge the gap between simulation and physical implementation.

A. Problem Formulation

We formulate autonomous drifting control as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where the state space $\mathcal{S}$ encompasses both vehicle states and reference trajectory information, $\mathcal{A}$ represents the action space including steering angle and individual wheel speeds, $\mathcal{P}$ denotes the transition dynamics governed by the IWD vehicle model, $\mathcal{R}$ is the reward function designed to achieve precise trajectory tracking, stable drift performance with significant sideslip angles, and smooth control quality, γ is the discount factor. The objective is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maps states to control actions while maximizing the expected return:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t\right) \right]. \tag{4}$$

B. Learning Framework

Building upon the problem formulation, we specify the concrete design of each component in our framework, including a carefully designed observation space, a physically constrained action space, and a structured reward function.

1) Observation Space: While the MDP state space $\mathcal{S}$ encompasses the complete vehicle dynamics and environmental information, direct policy learning from raw states often suffers from sampling inefficiency, primarily due to dimensionality issues and lack of invariance properties. Therefore, we design an observation space $\mathcal{O}$ that extracts the essential information from $\mathcal{S}$ and transforms it into a representation efficient for policy learning, which includes:

- **Future Waypoints Information** enables the policy to anticipate path curvature changes and adjust control inputs for smoother transitions. For each preview point along the reference path, we express its position, heading angle, and sideslip angle in the vehicle frame.
- **Trajectory Tracking Errors** measure the vehicle's deviation from the reference trajectory, including lateral position error, heading angle error, path curvature discrepancy, and sideslip angle deviation.
- **Vehicle State Variables** describe the instantaneous dynamics through coordinate-independent quantities comprising yaw rate r, sideslip angle β, velocity V, wheel velocities, and previous control inputs (δ and ω^{ij}).

These observations provide the policy with comprehensive state information, enabling it to learn effective control strategies for stable drifting along the reference trajectories.

2) Action Space: The steering angle is bounded to $[-0.46, 0.46]$ radians (approximately $\pm 26.35°$), corresponding to the physical limits of the steering mechanism. The wheel linear velocities are constrained within $[1, 7]$m/s, where the lower bound prevents the vehicle from stopping or moving too slowly to maintain drift conditions, while the upper bound ensures controllable drifting behavior.

3) Reward Design: The reward function integrates multiple objectives to achieve stable drift behavior while ensuring precise trajectory following:

$$r = r_{\text{tracking}} + r_{\text{control}} + r_{\text{aux}}. \tag{5}$$

Trajectory tracking rewards r_{tracking} quantify the vehicle's tracking accuracy relative to the reference path through quadratic penalties on tracking errors:

$$r_{\text{tracking}} = w_{\text{pos}} r_{\text{pos}} + w_{\text{dir}} r_{\text{dir}} + w_{\text{curv}} r_{\text{curv}} + w_{\text{drift}} r_{\text{drift}}, \tag{6}$$

where:

$$r_{\text{pos}} = -e_{\text{pos}}^2, r_{\text{dir}} = -e_{\text{dir}}^2, r_{\text{curv}} = -e_\kappa^2 = -(\kappa - \kappa_{\text{ref}})^2,$$
$$r_{\text{drift}} = -e_\beta^2 = -(\beta - \beta_{\text{ref}})^2. \tag{7}$$

Here, e_{pos} and e_{dir} represent position and direction errors defined in a curvilinear coordinate system along the reference trajectory as shown in Fig. 3, κ and κ_{ref} represent the actual and reference path curvature, and β and β_{ref} denote the actual and reference sideslip angles.

Control quality terms r_{control} impose constraints on the control inputs to ensure vehicle stability:

$$r_{\text{control}} = w_{\text{smooth}} r_{\text{smooth}}. \tag{8}$$

The smoothness term r_{smooth} penalizes rapid changes in consecutive commands:

$$r_{\text{smooth}} = -(\delta_t - \delta_{t-1})^2 - 10^{-4} \sum_{ij \in \{\text{fl,fr,rl,rr}\}} (\omega_t^{\text{ij}} - \omega_{t-1}^{\text{ij}})^2, \tag{9}$$

where subscripts t and $t-1$ denote current and previous time steps, and the scaling factor 10^{-4} balances the magnitudes of wheel speed variations relative to steering changes.

To further shape the learning process and ensure proper drift execution, we incorporate auxiliary terms:

$$r_{\text{aux}} = w_{\text{slip}} r_{\text{slip}} + w_{\text{speed}} r_{\text{speed}} + w_{\text{prog}} r_{\text{prog}}. \tag{10}$$

The wheel slip regulation term limits longitudinal slip at the front wheels to preserve steering effectiveness:

$$r_{\text{slip}} = -(v_x^{\text{fl}} - \omega^{\text{fl}} R)^2 - (v_x^{\text{fr}} - \omega^{\text{fr}} R)^2, \tag{11}$$

where v_x^{ij} represents the wheel's longitudinal velocity and R is the wheel radius.

A velocity incentive encourages the vehicle to maintain sufficient speed necessary for sustained drift execution:

$$r_{\text{speed}} = \min(0, V - 0.5). \tag{12}$$

In addition, to encourage stable and consistent forward movement along the reference trajectory, we incorporate a progression reward r_{prog} that is proportional to the distance traveled along the reference trajectory, clipped to a maximum value corresponding to a fixed distance. This reward term incentivizes the agent to maintain forward progress while preventing excessive rewards from the policy exploiting shortcuts.

C. Domain Randomization Strategy

To bridge the simulation-to-reality gap in autonomous drifting, we implement domain randomization techniques that address key uncertainties in real-world vehicle dynamics.

1) Trajectory Randomization for Policy Generalization: We create continuous paths by integrating curvature profiles with varying signs and magnitudes, maintaining smoothness ($C^{(1)}$ continuity) across transitions, which enables the policy to learn both stable drifting and smooth transitions between opposite drift directions. Each generated path provides context through position coordinates, velocity direction, and the desired direction of sideslip angle determined by local path curvature.

2) Initial State Randomization for Robust Learning: To handle real-world uncertainties and unexpected states during drift execution, we implement state randomization in our training, initializing vehicles at varied positions along reference trajectories, with controlled Gaussian perturbations added to starting positions ($\Delta x, \Delta y \sim \mathcal{N}(0, \sigma_{\text{pos}}^2)$) and heading angles ($\Delta\psi \sim \mathcal{N}(0, \sigma_{\text{heading}}^2)$). We further randomize dynamic states including velocities, sideslip angles, and yaw rates based on local trajectory curvature, which ensures the policy learns robust control strategies that generalize beyond training conditions.

3) Tire Model and Dynamic Disturbance Randomization: To account for unmodeled dynamics and varying tire-road interactions, we implement disturbances at two levels. We randomize Pacejka tire model parameters (B, C, D) sampled uniformly at each episode initialization to represent different tire-road conditions. We further implement an auto-regressive disturbance process that injects structured noise into the tire forces. The disturbance follows $d_{t+1} = ad_t + w\varepsilon_t$, where d_t represents the disturbance vector affecting tire forces, a controls temporal correlation, and w scales the Gaussian innovation ε_t. This dual-layer approach accounts for both persistent and transient uncertainties in tire-road interactions.

The contribution of each randomization component is analyzed through ablation studies (see Appendix).

TABLE I
TRAINING PARAMETERS

Reward Weights				Randomization Parameters			
Reward	Weight	Reward	Weight	Param.	Value	Param.	Range
r_{pos}	2.4	r_{speed}	0.1	σ_{pos}	0.1 m	r_{init}	[1,3] rad/s
r_{dir}	0.5	r_{smooth}	0.015	σ_{heading}	0.1 rad	β_{init}	[-1,1] rad
r_{curv}	0.15	r_{slip}	0.005	a	0.95	V_{init}	[0,3] m/s
r_{drift}	1.6	r_{prog}	0.2	w	0.1		

Tire model parameters: $B \in [0.8, 1.0]$, $C \in [2.0, 2.5]$, $D \in [0.3, 0.4]$

V. EXPERIMENTAL RESULTS

We validate our approach through extensive experiments in both simulation and real-world environments, demonstrating the policy's ability to handle different drifting scenarios. The code is available at https://github.com/zhou-yh19/xcar-rlgpu.

A. Training Setup

We train the policy using Proximal Policy Optimization (PPO) algorithm [33], known for its training stability and requiring little hyperparameter tuning [34]. The policy network consists of three fully connected layers with (64, 32, 16) neurons respectively. The reference trajectories are parameterized as a sequence of waypoints with a fixed distance of 0.005 m between consecutive points. Our implementation utilizes approximately 10^5 parallel simulation environments on an NVIDIA GeForce RTX 3080 GPU, achieving convergence within 10.762 minutes on average. The training parameters are summarized in Table I.

B. Experimental Validation

We evaluate the trained policy across three distinct trajectory patterns:

- **Steady-State Circular Drift:** Basic drift stabilization along a circular path (radius = 1 m)
- **Eight-Shaped Maneuver:** Controlled drift through trajectory intersections requiring drift direction reversals
- **Variable-Curvature Track:** Advanced trajectory featuring varying path curvature while maintaining consistent drift states

C. Simulation Results

The policy demonstrates robust performance across all trajectory patterns while maintaining significant sideslip angles ($45° - 55°$) and stable velocities (1.5-2.5 m/s). Detailed analysis of each trajectory type reveals the following characteristics:

Fig. 5 shows the performance for circular trajectory (radius = 1 m). The controller achieves precise tracking with position

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

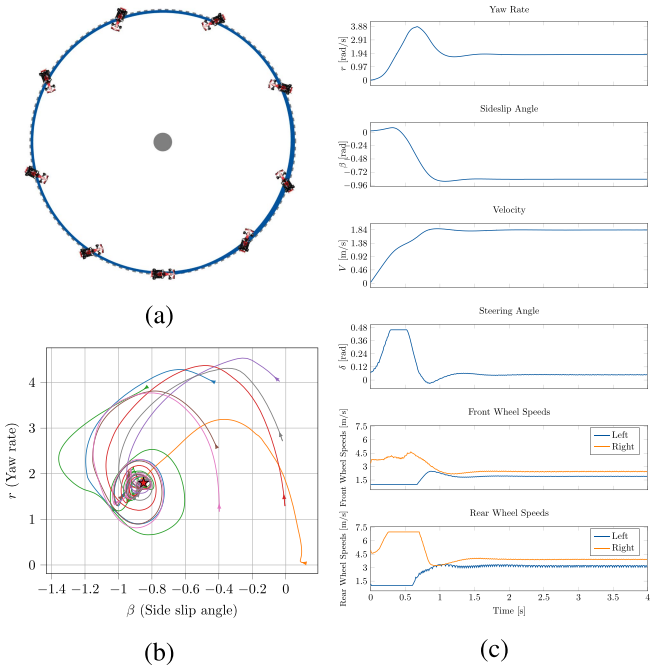

Fig. 5. Performance of the IWD drifting policy on a circular trajectory. (a) Paths for circular trajectory tracking (blue for actual vehicle path, gray for reference circle); filled gray circle marks the center, race cars show vehicle poses at sampled positions. (b) State convergence in the r-β phase plane, showing trajectories from different initial conditions converging to a natural drift equilibrium. (c) Time histories of states (r, β, V) and control inputs from zero initial conditions.

errors below 0.1 m, as observed in Fig. 5(a). Fig. 5(b) illustrates the convergence behavior in the r-β plane. Despite starting from various initial conditions, the system consistently converges to a natural drift equilibrium ($r = 1.85$ rad/s, $\beta =$ -0.85 rad, $V = 1.84$ m/s), which suggests that the learned policy has discovered a stable operating point that satisfies both the vehicle dynamics constraints and the geometric requirements for maintaining circular drifting.

We further analyze the time histories of states and control inputs from zero initial conditions in Fig. 5(c), which illustrates the distinct characteristics of IWD control during drift maneuvers. During the initiation phase (0–1.0 s), the controller exploits differential wheel speeds to induce drift efficiently - the rear right wheel maintains significantly higher angular velocity compared to the rear left wheel, with a similar but less pronounced differential on the front wheels. Once the system reaches steady-state drifting, the controller maintains asymmetric wheel speeds across the left and right sides of the vehicle, but with a reduced differential magnitude.

To validate the advantages of IWD's enhanced control authority demonstrated above, we also train policies on RWD configurations using identical hyperparameters. As shown in Fig. 6, the RWD system achieves stable circular drifting but requires approximately 1.5 s to reach drift states compared to IWD's 1 s, shows slightly larger deviations from the reference path, operates at smaller sideslip angles (-0.76 rad vs -0.85 rad), and demonstrates higher steering dependency.

Beyond circular trajectories, we evaluate the IWD system on more complex trajectory patterns. The eight-shaped path adds complexity through drift direction reversals. As shown in

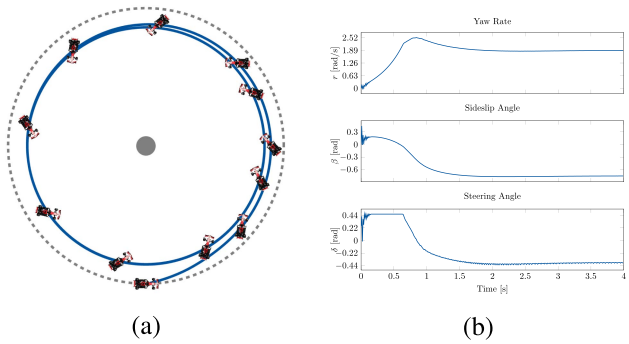

Fig. 6. Performance of the RWD drifting policy on a circular trajectory. (a) Paths for circular trajectory tracking. (b) Time histories of states (r, β) and the steering angle (δ).

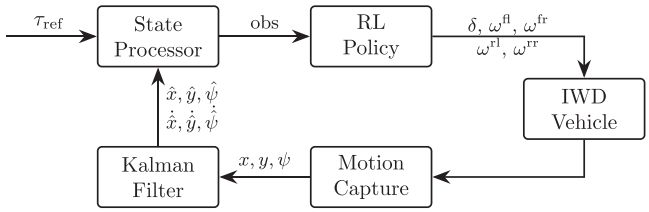

Fig. 7. Real-world deployment architecture.

Fig. 8(a,c), the vehicle effectively tracks the reference path while maintaining mean sideslip angles of 47° and vehicle speeds of 1.81 m/s. During the direction reversal phase (highlighted in red), the controller employs strategic differential wheel speeds: the front right wheel decelerates to 1.0 m/s while the left increases to 5.2 m/s. The rear wheels show even larger differentials, with the right at 1.0 m/s and left reaching 7.0 m/s, creating the substantial yaw moment needed for rotation. The transition can be characterized by three phases: right wheel deceleration (3.2–3.4 s), maximum wheel speed differential (3.4–3.6 s), and convergence to a new drift equilibrium (3.7–3.9 s), revealing the controller's ability to manage drift direction changes.

The variable-curvature track combines tight circles ($\kappa = 1.0$), gentle curves ($\kappa = 0.5$), and smooth transitions, inspired by [11], [15]. The controller maintains a consistent sideslip angle around -50° as shown in Fig. 8(c), despite the changing path geometry. For the red segment in Fig. 8(b), wheel speed differentials adapt smoothly to local curvature variations, demonstrating stable drift behavior throughout the trajectory.

The simulation results demonstrate effective autonomous drifting across multiple trajectory complexities, from basic circular tracking to challenging maneuvers requiring direction reversals. The comparative analysis demonstrates that IWD systems achieve better tracking precision and faster drift initiation through additional yaw moments generated by differential wheel speed control.

D. Real-World Validation

We implement our approach on a 1/10 scale IWD RC car platform mentioned in Section III-A. The trained policy transfers directly to the real platform without any additional fine-tuning, executing at 100 Hz on the onboard computer. The deployment architecture is illustrated in Fig. 7. Video demonstrations of

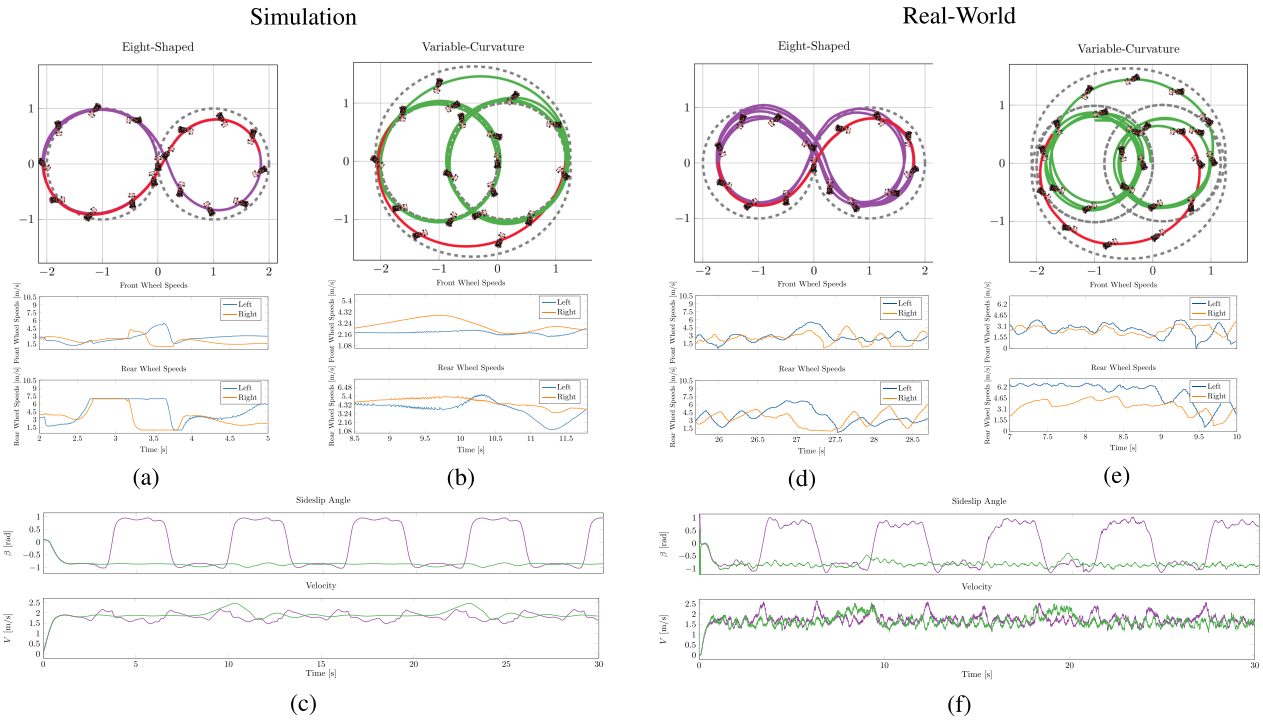

Fig. 8. IWD Performance comparison between simulation (left) and real-world experiments (right). Path tracking results for (a,d) eight-shaped path and (b,e) variable-curvature track, showing actual vehicle paths (purple/green) against reference paths (gray) with corresponding wheel speed time histories for highlighted segments (in red). (c,f) Comparison of sideslip angles and velocities for both trajectories (purple for eight-shaped path, green for variable-curvature track).

TABLE II
POSITION ERROR RMSE (MEAN ± STD, N=6)

Environment	Eight-Shaped Path	Variable-Curvature Path
Simulation	0.138 ± 0.001 m	0.075 ± 0.002 m
Real-World	0.221 ± 0.022 m	0.231 ± 0.037 m

the real-world experiments are available at https://youtu.be/6ovkdj5_1Yk.

To validate our approach, we focus on the eight-shaped path and the variable-curvature track, as shown in Fig. 8. Real-world experiments demonstrate remarkable consistency with simulation in maintaining stable sideslip angles and velocities, though with slightly more conservative trajectories characterized by smaller turning radius. For the eight-shaped path, during direction reversals as highlighted in Fig. 8(d), the controller achieves rapid drift transitions by dramatically increasing left wheel speeds, effectively exploiting the IWD configuration for yaw control. On the variable-curvature track, while in simulation drifting is maintained through higher right wheel speeds relative to left, the physical system achieves stable drifting through an opposite strategy of higher left wheel speeds, yet successfully maintains consistent drift states throughout the trajectory with the compromise of smaller sideslip angles. To quantify the tracking performance, we compute the root mean square error (RMSE) of position tracking for both simulation and real-world experiments, as summarized in Table II.

To further evaluate our approach, we test the vehicle on a path inspired by the Olympic rings where sustained drifting must be maintained through five tangent circles while repeatedly reversing drift direction at tangent points. The consistent state transitions shown in Fig. 1 demonstrate the controller's ability to maintain stable drift through multiple reversals, highlighting the reliability of our approach.

VI. CONCLUSION

This letter presents a reinforcement learning approach for autonomous drifting control using Individual Wheel Drive vehicles. The proposed framework achieves effective sim-to-real transfer through GPU-accelerated parallel simulation and systematic domain randomization, enabling successful deployment without fine-tuning.

Experimental validation demonstrates effectiveness across diverse drifting scenarios, including direction reversals and variable-curvature paths, with real-world experiments confirming consistent performance. IWD systems generate additional yaw moments through differential wheel speeds, enabling fast drift initiation and precise trajectory tracking. The open-source platform and codebase facilitate further research in IWD vehicle control and autonomous drifting applications.

Future work could incorporate lateral load transfer and suspension dynamics to better capture weight redistribution effects during aggressive cornering, potentially enhancing policy robustness and performance in extreme drift scenarios.

APPENDIX

To validate the effectiveness of our domain randomization strategy, we conduct ablation experiments by training policies with different randomization configurations. Each policy is evaluated on 100 trials of eight-shaped trajectories under challenging test conditions with broader parameter distributions: Pacejka parameters $B \in [0.2, 3]$, $C \in [1.5, 3]$, $D \in [0.2, 0.5]$ and increased disturbance scaling.

TABLE III
DOMAIN RANDOMIZATION ABLATION STUDY RESULTS

Method	Success Rate	RMSE (m)	Sideslip Angle (°)
Full randomization	75.0%	0.146±0.053	0.81±0.09
w/o tire rand.	73.0%	0.158±0.054	0.88±0.09
w/o init. state rand.	46.0%	0.233±0.049	0.79±0.06
w/o dynamic disturb.	80.0%	0.159±0.062	0.76±0.10
w/o traj. rand.	73.0%	0.222±0.053	0.71±0.07

The results in Table III demonstrate that initial state randomization is the most critical component, with its removal causing a 29% drop in success rate and 60% increase in tracking error. Trajectory randomization also significantly impacts performance, showing 52% increase in RMSE when removed.

While removing dynamic disturbance shows slight simulation improvement (80% vs 75%), only the full randomization configuration successfully transfers to real-world deployment. This indicates that dynamic disturbance is essential for handling unmodeled dynamics in physical systems.

REFERENCES

[1] P. Cai, X. Mei, L. Tai, Y. Sun, and M. Liu, "High-speed autonomous drifting with deep reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1247–1254, Apr. 2020.

[2] J. Betz et al., "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 458–488, 2022.

[3] J. Y. Goh, T. Goel, and J. C. Gerdes, "Toward automated vehicle control beyond the stability limits: Drifting along a general path," *J. Dyn. Syst., Meas., Control*, vol. 142, no. 2, 2020, Art. no. 021004.

[4] B. Yang, Y. Lu, X. Yang, and Y. Mo, "A hierarchical control framework for drift maneuvering of autonomous vehicles," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 1387–1393.

[5] Y. Lu, B. Yang, J. Li, Y. Zhou, H. Chen, and Y. Mo, "Consecutive inertia drift of autonomous RC car via primitive-based planning and data-driven control," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2023, pp. 4835–4840.

[6] C. Voser, R. Y. Hindiyeh, and J. C. Gerdes, "Analysis and control of high sideslip manoeuvres," *Veh. Syst. Dyn.*, vol. 48, no. S1, pp. 317–336, 2010.

[7] R. Y. Hindiyeh and J. C. Gerdes, "A controller framework for autonomous drifting: Design, stability, and experimental validation," *J. Dyn. Syst., Meas., Control*, vol. 136, no. 5, 2014, Art. no. 051015.

[8] T. P. Weber and J. C. Gerdes, "Modeling and control for dynamic drifting trajectories," *IEEE Trans. Intell. Veh.*, vol. 9, no. 2, pp. 3731–3741, Feb. 2023.

[9] F. Djeumou et al., "One model to drift them all: Physics-informed conditional diffusion model for driving at the limits," in *Proc. 8th Annu. Conf. Robot Learn.*, 2025, pp. 604–630.

[10] M. Cutler and J. P. How, "Autonomous drifting using simulation-aided reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2016, pp. 5442–5448.

[11] F. Domberg, C. C. Wembers, H. Patel, and G. Schildbach, "Deep drifting: Autonomous drifting of arbitrary trajectories using deep reinforcement learning," in *Proc. Int. Conf. Robot. Automat.*, May 2022, pp. 7753–7759.

[12] J. Y. Goh and J. C. Gerdes, "Simultaneous stabilization and tracking of basic automobile drifting trajectories," in *Proc. IEEE Intell. Veh. Symp.*, 2016, pp. 597–602.

[13] S. Meijer, A. Bertipaglia, and B. Shyrokau, "A nonlinear model predictive control for automated drifting with a standard passenger vehicle," in *IEEE Int. Conf. Adv. Intell. Mechatronics (AIM)*, 2024, pp. 284–289.

[14] X. Tian, S. Yang, Y. Yang, W. Song, and M. Fu, "A multi-layer drifting controller for all-wheel drive vehicles beyond driving limits," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 2, pp. 1229–1239, Apr. 2024.

[15] J. Y. Goh, T. Goel, and J. C. Gerdes, "A controller for automated drifting along complex trajectories," in *Proc. 14th Int. Symp. Adv. Veh. Control*, 2018, vol. 7, pp. 1–6.

[16] C. Hu, L. Xie, Z. Zhang, and H. Xiong, "A novel model predictive controller for the drifting vehicle to track a circular trajectory," *Veh. Syst. Dyn.*, vol. 63, no. 3, pp. 537–566, 2025.

[17] Z. Shi, H. Chen, S. Yu, R. Findeisen, and H. Guo, "Nonlinear model predictive control for autonomous vehicle drifting," *Int. J. Robust Nonlinear Control*, vol. 35, pp. 2760–2779, 2025.

[18] E. Velenis, E. Frazzoli, and P. Tsiotras, "On steady-state cornering equilibria for wheeled vehicles with drift," in *Proc. 48 h IEEE Conf. Decis. Control Held Jointly 2009 28th Chin. Control Conf.*, 2009, pp. 3545–3550.

[19] E. Velenis, D. Katzourakis, E. Frazzoli, P. Tsiotras, and R. Happee, "Steady-state drifting stabilization of RWD vehicles," *Control Eng. Pract.*, vol. 19, no. 11, pp. 1363–1376, 2011.

[20] E. Joa, H. Cha, Y. Hyun, Y. Koh, K. Yi, and J. Park, "A new control approach for automated drifting in consideration of the driving characteristics of an expert human driver," *Control Eng. Pract.*, vol. 96, 2020, Art. no. 104293.

[21] F. Djeumou, J. Goh, U. Topcu, and A. Balachandran, "Autonomous drifting with 3 minutes of data via learned tire models," in *Proc. Int. Conf. Robot. Automat.*, 2023, pp. 968–974.

[22] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 465–472.

[23] S. Bhattacharjee, K. D. Kabara, R. Jain, and K. Kabara, "Autonomous drifting rc car with reinforcement learning," *Dept. Comput. Sci., Univ. Hong Kong*, Tech. Rep, 2018.

[24] J. M. Gonzales, *Plan. and Control of Drift Maneuvers With the Berkeley Auton. Race Car*. California, Berkeley, USA: University of California, Berkeley, 2018.

[25] S. Karaman et al., "Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at mit," in *Proc. Integr. STEM Educ. Conf.*, 2017, pp. 195–203.

[26] S.S. Srinivasa et al., "Mushr: A low-cost, open-source robotic racecar for education and research," 2019, *arXiv:1908.08031*.

[27] K. Hart et al., "RoSCAR," in *Proc. 2014 Workshop Mobile Augmented Reality Robot. Technol.-Based Syst. - MARS 14*, ACM Press, 2014, pp. 3–8.

[28] M. O'Kelly et al., "F1/10: An open-source autonomous cyber-physical platform," 2019, *arXiv:1901.08567*.

[29] S. Milani, H. Marzbani, and R. N. Jazar, "Vehicle drifting dynamics: Discovery of new equilibria," *Veh. Syst. Dyn.*, vol. 60, no. 6, pp. 1933–1958, 2022.

[30] V. Makoviychuk et al., "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021, *arXiv:2108.10470*.

[31] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," *Proc. Mach. Learn. Res.*, vol. 123, pp. 77–89, 2020.

[32] A. Paszke et al., "Automatic differentiation in PyTorch," in *NIPS Workshop Autodiff*, 2017.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[34] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.