vehicles

Article

Sim-to-Real Application of Reinforcement Learning Agents for Autonomous, Real Vehicle Drifting

Szilárd Hunor Tóth [1,2], Zsolt János Viharos [2,3], Ádám Bárdos [1] and Zsolt Szalay [1,*]

[1] Department of Automotive Technologies, Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics, 6 Stoczek St., Building J, H-1111 Budapest, Hungary; bardos.adam@kjk.bme.hu (Á.B.)
[2] Intelligent Processes Research Group, Research Laboratory on Engineering & Management Intelligence, HUN-REN Institute For Computer Science and Control (SZTAKI), 13-17 Kende u., H-1111 Budapest, Hungary
[3] Department of Management and Business Law, Faculty of Economics and Business, John von Neumann University, 10. Izsáki u., H-6000 Kecskemét, Hungary
* Correspondence: szalay.zsolt@kjk.bme.hu

Abstract: Enhancing the safety of passengers by venturing beyond the limits of a human driver is one of the main ideas behind autonomous vehicles. While drifting is mostly witnessed in motorsports as an advanced driving technique, it could provide many possibilities for improving traffic safety by avoiding accidents in extreme traffic situations. The purpose of the research presented in this article is to provide a machine learning-based solution to autonomous drifting as a proof of concept for vehicle control at the limits of handling. To achieve this, reinforcement learning (RL) agents were trained for the task in a MATLAB/Simulink-based simulation environment, using the state-of-the-art Soft Actor–Critic (SAC) algorithm. The trained agents were tested in reality at the ZalaZONE proving ground on a series production sports car with zero-shot transfer. Based on the test results, the simulation environment was improved through domain randomization, until the agent could perform the task both in simulation and in reality on a real test car.

Keywords: autonomous vehicles; automated drifting; reinforcement learning; sim-to-real; vehicle motion control; varying road surfaces; vehicle dynamics

Citation: Tóth, S.H.; Viharos, Z.J.; Bárdos, Á.; Szalay, Z. Sim-to-Real Application of Reinforcement Learning Agents for Autonomous, Real Vehicle Drifting. *Vehicles* **2024**, *6*, 781–798. https://doi.org/10.3390/vehicles6020037

Academic Editor: Lei Zhang

Received: 14 March 2024
Revised: 22 April 2024
Accepted: 26 April 2024
Published: 30 April 2024

1. Introduction

In the rapidly evolving landscape of transportation, autonomous vehicles stand out as one of the most anticipated and researched fields in recent years. In the area of autonomous vehicle control (AVC), advancements in technology will soon allow the application of automated driving systems (ADS) that are capable of acquiring specific human driving skills. These include, of course, lower SAE (Society of Automotive Engineers) levels [1] like cruise control, lane keeping, and basic parking [2], and also higher SAE levels like L2+ and L3 vehicle automation. However, motion control beyond the handling limit is still mostly unresolved, and its solutions primarily exist only in virtual reality.

Vehicle stability refers to the ability to maintain equilibrium or resume its original status after experiencing a disturbance such as a wind gust, uneven road surface, or a sudden small change in the steering [3]. Stability analysis can be performed in many ways [4]. In most cases, it refers to estimating a stability region where the vehicle has the above-described ability to maintain equilibrium. It can also be approached by analyzing tire slip angles and tire force saturation [5]. This can also be an important aspect of vehicle stability control, especially when considering the tire's radial, circumferential, and lateral stiffness. Based on related studies [6], these can also affect the ability to transfer the torque input of the vehicle into the ability to cause and prevent sliding and drifting.

Current state-of-the-art Advanced Driver-Assistance Systems (ADAS) follow the proven principle of avoiding such driving scenarios that might cause a road vehicle to behave in ways that are unknown to most human drivers. There are several well-known technical approaches to this, like ABS (Anti-lock Braking System), ASR/TCS (Anti-Slip Regulation/Traction Control System), and ESC (Electric Stability Control). While it is undeniable that these driving systems provide safer and more effective vehicle control [7], it is still not proven that these would be the most effective way to reduce the number of control loss-related accidents. A good example of this would be a collision-avoidance scenario where an object enters the vehicle's collision path far inside its braking zone. In such cases, there is a chance that performing an avoidant steering maneuver could result in losing stability and this could endanger the passengers or other participants. Another example would be aquaplaning or prolonged driving on a low-traction surface at a high speed. In [8], the authors address multiple scenarios where a maneuver at the handling limit significantly reduces fatality and collision risk. In addition to this reasoning, it is also an essential scope of AVC to enhance road vehicle transportation beyond human limits, especially considering the rapidly evolving field of overactuated vehicle control [9,10], which is an overly complex task for most human drivers. This makes this area of research not just important for safety-critical systems, but also for motorsports, space exploration, and military sciences.

Drifting can be considered one of the most basic maneuvers of unstable vehicle control. It is a cornering motion mostly characterized by a high sideslip angle, saturated tire forces, and counter-steering. As such, it is an excellent example of unstable vehicle motion, and its predictable control can improve autonomous vehicles' capabilities significanty. Defining drifting for a formal representation can be carried out in many different ways. The research presented in this article is focused on steady-state drifting, which sets a goal to reach and maintain a vehicle state that can be considered as an unstable drift equilibrium (for definition, see Section 2.2).

The current literature on maneuvering autonomous vehicles beyond the limit of handling presents many different approaches to the problem. First, it is important to highlight the applications of linear feedback controllers on two-wheeled vehicle models, which successfully utilized the model-based calculation of equilibrium points. Concerning the selected actuators, in [11], the longitudinal and lateral movement handling operations were decoupled using two independent controllers. In [12,13], the authors used a Linear Quadratic Multiple Input Multiple Output (LQ MIMO) controller, wherein this controller was also successful. Also, positive results were obtained using a four-wheel vehicle model [14]. In addition, some works utilize the added control of braking [14–16]. Trajectory following drift controllers are also present in the literature [17].

MPC (Model Predictive Control) has proven successful for both stabilization and track-following drift tasks [18,19]. It also performed well when simulating changing road conditions, thus indicating a good adaptive property. Robust control-based [20] and multi-layer methods [21] are also present in the literature. Among the results that consider path planning and following, it is worth mentioning the hybrid application of linear control and MPC for implementing a parking maneuver to be carried out with drift [22]. The former construction worked both in simulation and with a small RC car. To present a more advanced objective, drift-based collision avoidance has also been considered with different hierarchical controllers [23,24].

The application of reinforcement learning to solve autonomous drifting also shows a promising direction. Among the potential advantages of reinforcement learning is primarily the ability to generalize under constantly changing driving conditions (such as varying road surfaces) and the absence of the need to use previous data for training. It also provides an intuitive way to define objectives for the driving agent through its reward function without specifying low-level control demands or planning an exact trajectory if the use case in question does not require it.

In general, it can be stated that deep reinforcement learning (DRL) has been the most popular in autonomous vehicle control [25]. The main reason is that a discrete agent operating in a continuous environment performs the task insufficiently or in too piecemeal of a fashion in theory, usually due to the incorrect construction of the finite space representation. Despite this, this paper has in scope the application of a discrete agent along with a DRL-based contender. The results produced by these methods in the simulation have been promising so far [26–28].

In addition to the previous works of the authors, other works, like [29], used the Probabilistic Inference for Learning Control (PILCO) [30] model-based policy-search algorithm to achieve steady-state drifting in simulation and on a small-scale remote-controlled vehicle. Strengthening these results, ref. [31] performed similar work using PILCO and deep Q-learning (DQL). Also, ref. [32] defined drifting as a track-following task in the CARLA [33] simulator, where the exact goal was to achieve large sideslip angles at high speeds. The agent was trained on several predetermined courses, and then evaluated on a course unknown to it, with several different types of vehicles and road condition settings. In [34], the authors proposed a similar solution and approach. The training in these two cases was carried out with the Soft Actor–Critic (SAC) algorithm, and [35] also experimented with this task with the TD3 algorithm [36], which also proved to work well. Further developing the results mentioned so far, ref. [37] successfully developed an agent that can even drift on arbitrary trajectories in simulation. Also, as with control methods, drift parking has been attempted with RL too [38]. In addition to these works, an autonomous racing task, including control at handling limits, was also attempted with a model-based policy search [39].

Problem Statement

This paper presents novel results on applying reinforcement learning agents for autonomous drifting on a full-scale, real test vehicle. This is the same car that was used in [13,19]. To the authors' best knowledge, this is the first time such an agent was successfully deployed to perform autonomous drifting on a real vehicle.

The exact goal of the agent is to perform a steady-state drifting maneuver and maintain it for arbitrary durations, even with GNSS (Global Navigation Satellite System) and/or sensor noise/delay with a real test car. The car is sped up by a controller, which is independent of the agent, to $v_x = 28$ km/h, and, when this is attained, the control of the vehicle is entirely released to the RL agent. Then, its objective is to reach a defined medium-speed target drift state and maintain it with a minimal mean error while handling the actuators smoothly enough to not lose stability from sudden actuator handling, even with small disturbances. This goal was formulated intuitively without using prior data from expert drivers. This problem definition is possible because an RL agent is used for the solution. The considered traction conditions for this application are set to range between grip coefficients of $\mu = 0.6$ and $\mu = 0.95$, and the tests were performed in such conditions. Learning to drift for any traction coefficient in this range effectively requires an adaptive ability, which is one of the reasons to use an RL agent. Successfully performing this maneuver would prove that RL agents can learn to control vehicles beyond handling limits. This would indicate that it is possible to train such agents for collision avoidance where only such an aggressive maneuver could provide a possibility to ensure the safety of the passengers.

In the following section, the paper presents the vehicle model structure and drift equilibrium calculation background, which were implemented in the MATLAB/Simulink simulation environment for training the agents. In Section 3, the RL algorithm used for training (SAC) is described, along with the agent's specific structure and parameters. Section 4 discusses the applied sim-to-real methodology, the presentation of the test vehicle's characteristics, and the hardware used for the real tests. Section 5 uncovers the results achieved. A discussion of the analysis, acknowledgment, and references closes the paper.

2. Vehicle Dynamics

To have a proper understanding and a correct implementation for the problem of steady-state drifting, addressing the vehicle's dynamics is essential. In general, drifting requires a vehicle with sufficient rear torque and traction. Typically, most of these are rear-wheel drive (RWD) vehicles, so the chosen model is also RWD, but, at the same time, drift-like behavior is not impossible with front-wheel or four-wheel drive vehicles either. The vehicle model features presented throughout this section are validated based on previous works [13,19].

Because training a learning agent in a simulation environment requires significant computation resources, it is beneficial to have a simple but still realistic vehicle environment. In the case of drifting, ignoring pitch and roll dynamics does not seem to prevent engineering agents that can perform drifting (for example, [13] proves this), and, at the same time, involving them would significantly increase the dimensionality of the problem. Therefore, the vehicle models presented in this paper only consider three degrees of freedom: translation in the x and y directions and rotation around the z axis (yaw).

The Newton–Euler equations of motion of the vehicle in the body frame's coordinate system can be described as follows [40]:

$$F_x = m\left(\dot{v_x} - rv_y\right) \tag{1}$$

$$F_y = m\left(\dot{v_y} + rv_x\right) \tag{2}$$

$$M_z = I_z\dot{r} \tag{3}$$

where v_x and v_y are the velocities in the x and y directions, respectively, r is the speed of vehicle rotation of the vehicle around the z-axis, m is the vehicle's total mass, and I_z is the inertia constant. Based on these, the derivatives can be expressed as

$$\dot{v}_x = \frac{1}{m}(F_x - F_A) + rv_y \tag{4}$$

$$\dot{v}_y = \frac{1}{m}F_y - rv_x \tag{5}$$

$$\dot{r} = \frac{1}{I_z}M_z \tag{6}$$

The force components in (1)–(3) can also be derived as the total effect of the forces acting on the wheels:

$$F_x = F_{x_f}\cos\delta + F_{x_r} - F_{y_f}\sin\delta \tag{7}$$

$$F_y = F_{y_f}\cos\delta + F_{y_r} - F_{x_f}\sin\delta \tag{8}$$

$$M_z = aF_{y_f}\cos\delta + aF_{x_f}\sin\delta - bF_{y_r} \tag{9}$$

where a and b are the front and rear wheelbase ($l = a + b$), respectively, and δ is the front (steered) wheel angle.

Since the chosen vehicle has rear-wheel drive, the longitudinal force acting on the steered front wheel is zero: $F_{x_f} = 0$. The rear longitudinal force F_{x_r} and the front wheel angle δ are the input parameters of the vehicle. The specific constant values (based on the properties of the test vehicle) for the vehicle model's equations are described in Table A1. Figure 1 illustrates the vehicle model.

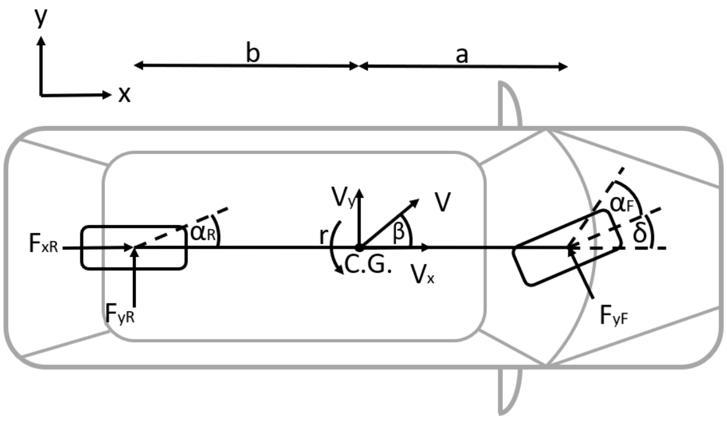

Figure 1. The implemented two-wheel dynamic vehicle model in its coordinate frame.

2.1. Tire Modeling

The tire model is one of the most important parts of the simulation model's structure. Without a proper tire model, performing accurate simulations for drifting is not possible. The most critical expectation from the tire model is that the saturation of the lateral forces acting on the tires can be well described, which is an essential feature of drifting. For example, in [11–13,26–28], the implementation of a lateral slip brush tire model, proposed by [41], worked well for designing controllers for drifting. However, modeling longitudinal slip on the rear tire in addition to lateral slip supports an RL agent greatly, especially in the initiation phase of a drift maneuver. On this note, a combined slip brush tire model was implemented, based on the work of Pajecka [42]. Also, there may be other state-of-the-art tire models which would be worth considering, like the TMeasy model [43], the elliptical tire model [44], and the Mooney–Rivlin material model [45,46].

The general form of the Pacejka model stands as the following:

$$\begin{aligned} y(x) &= D \cdot \sin\left(C \cdot \tan^{-1}\left(Bx - E\left(Bx - \tan^{-1}(Bx)\right)\right)\right) \\ Y(x) &= y(x) + Sv \\ x &= X + Sh \end{aligned} \tag{10}$$

where $y(x)$ (the output) is the longitudinal/lateral force component, x (the input) is the slip angle or slip ratio, D is the peak factor, C is the shape factor, B is the stiffness factor, E is the curvature factor, and Sv and Sh are the vertical and horizontal shifts, respectively. The shifts consider the camber thrust, conicity, and rolling resistance. Given the S_x slip ratio, S_{xp} peak slip ratio, α slip angle, and α_p peak slip angle parameters, the normalized slip parameters are determined:

$$\begin{aligned} S_x^* &= \frac{S_x}{S_{xp}} \\ \alpha^* &= \frac{\alpha}{\alpha_p} \end{aligned} \tag{11}$$

Afterward, the resultant slip of the tire and the modified slip ratio and slip angle are determined:

$$\begin{aligned} S^* &= \sqrt{(S_x^*)^2 + (\alpha^*)^2} \\ S_{xm} &= S^* \cdot S_{xp} \\ \alpha_m &= S^* \cdot \alpha_p \end{aligned} \tag{12}$$

From these, the longitudinal and lateral force components are given:

$$F_x = F_{x_0} \cdot \frac{S_x^*}{S^*} \tag{13}$$

$$F_y = F_{y_0} \cdot \frac{\alpha^*}{S^*} \tag{14}$$

where F_{x_0}/F_{y_0} are the longitudinal/lateral force components, calculated using the modified slip ratio/angle and Equation (10).

In the model, the implemented wheel model is based on [47] and summarized as

$$J\omega_w = (T_w - T_b) - F_x r_w - F_{rr} r_w \tag{15}$$

where J is the wheel inertia, T_w is the wheel torque, T_b is the break torque, F_{rr} is the roll resistance, r_w is the wheel radius, and ω_w is the wheel speed. Given these, the rear longitudinal slip κ can be calculated as

$$\kappa = \frac{\omega r_w - v_x}{v_x} \tag{16}$$

while the lateral slip angles α_f, α_r and the sideslip angle β are formulated as

$$\begin{aligned} \alpha_f &= \tan^{-1}\left(\frac{v_y + ar}{v_x}\right) - \delta \\ \alpha_r &= \tan^{-1}\left(\frac{v_y - br}{v_x}\right) \end{aligned} \tag{17}$$

$$\beta = \tan^{-1}\left(\frac{v_y}{v_x}\right) \tag{18}$$

All the above equations conclude the implemented vehicle model, whose defined parameters are collected in Table A1.

2.2. Drift Equilibrium Calculation

As mentioned, this paper focuses on autonomous drifting and has an objective defined as reaching a desired drift equilibrium state. Based on the work in [48], drift can be described as vehicle equilibrium states that consider rear lateral slip saturation, which is how reaching handling limits is described, based on [41]. As such, writing

$$\begin{aligned} &\dot{v}_x = \dot{v}_y = \dot{r} = 0 \\ &\alpha_r = \alpha_{p_r} \\ &\alpha_f < \alpha_{p_f} \end{aligned} \tag{19}$$

and combining with Equations (4)–(18) formulates an algebraic equation system with five variables $\left(v_x, v_y, r, F_{x_r}, \delta\right)$.

The system can be solved by assigning a fixed constant value to any of these five parameters. In this presented research, given that $\delta = -10^\circ$ and $v_x = 10$ m/s, the following drift equilibrium was used for the objective :

$$S_{drift} = \left(v_{x_{drift}}, v_{y_{drift}}, r_{drift}\right) = (10\,\text{m/s}, -3.3728\,\text{m/s}, 0.8335\,\text{rad/s}) \tag{20}$$

This leads to a left-directional circular drift motion with a sideslip angle of $\beta = -18.6382^\circ$.

3. The Reinforcement Learning Model

The fundamental concept of reinforcement learning revolves around the continuous cycle of interactions between an agent and the environment. In this learning paradigm, the agent's primary objective is to maximize the cumulative reward it receives from the environment. This is achieved by the agent making decisions (performing actions) based on the current state of the environment, which then responds with feedback in the form of rewards. The agent discovers the optimal actions (its optimal policy) by exploring the action space.

For the previously introduced vehicle model, the following state and action spaces are defined:

$$S = \left(v_x, v_y, r, \dot{v}_x, \dot{v}_y, \dot{r}\right) \in \mathbb{R}^6 \tag{21}$$

$$A = \left(acc_{ped}, \delta_{steer}\right) \in \{[0\,\%, 100\,\%] \times [-420°, 420°]\} \tag{22}$$

The state space $S \in \mathcal{S}$ contains the model's describing velocities and their derivatives. The action space $A \in \mathcal{A}$ represents the model's input variables through the form of the gas pedal acc_{ped} and the steering wheel angle δ_{steer}, limited to a physically achievable range. The conversion of these variables into an (F_{x_r}, δ) input pair is discussed in Section 4.3.

The state transition function $P : [\mathcal{S}, \mathcal{A}] \rightarrow \mathcal{S}$ maps accordingly to the underlying vehicle dynamics. It can be proven that the environment satisfies the Markov property:

$$\begin{aligned} &S_{t+1} = P(S_t, A_t) \\ &\mathbb{P}[S_{t+1} = S | S_t, A_t, S_{t-1}, A_{t-1}, \ldots] = \mathbb{P}[S_{t+1} = S | S_t, A_t] \end{aligned} \tag{23}$$

The reward function is the RMSE of the current state from the target drift state, with an added term that punishes the agent relative to the difference between its current and previous actuator request (action):

$$R(S_t, A_t) = \sqrt{\frac{1}{3}\sum_{i=1}^{3}\left(\frac{S_t^{(i)}}{S_{drift}^{(i)}} - 1\right)^2} + \sqrt{\frac{1}{2}\left(\left(\frac{A_t^{(1)} - A_{t-1}^{(1)}}{50} - 1\right)^2 + \left(\frac{A_t^{(2)} - A_{t-1}^{(2)}}{420}\right)^2\right)} \tag{24}$$

The first term in (24) ensures that the agent reaches and maintains the target drift state with a minimal mean error, where $S_t^{(i)}$ is the value of the ith position in the current state vector and $S_{drift}^{(i)}$ is the same for the target state vector. The second term encourages the agent to smooth its driving policy during the drift's initiation and stabilization phases, where $A_t^{(i)}$ is the value of the ith position in the issued action vector and $A_{t-1}^{(i)}$ is the same but for the previous action vector.

The agents were trained in MATLAB while connected to the Simulink environment that contained the vehicle model introduced in Section 2. The training algorithm used was the state-of-the-art Soft Actor–Critic (SAC) algorithm [49], specifically designed for continuous state and action spaces.

The agent is separated into two parts (see Figure 2). The actor represents the agent's policy, a mapping from the state space to the action space. This is a stochastic function, meaning, technically, that the action is generated from a normal distribution $A \sim \mathcal{N}(\mu_S, \sigma_S)$, where the distribution's parameters are given by the policy function $(\mu_S, \sigma_S) = \pi(S)$. This stochasticity helps the agent to explore the action space. The measure of this is the distribution's differential entropy. The algorithm tunes the weight (temperature) of the entropy factor adaptive to the received rewards: less reward means more exploration is needed, while higher observed rewards in the long run cause reduced entropy.

The critic is the value function, whose purpose is to approximate the expected cumulative reward for a given state–action pair. It calculates the TD (Temporal Difference) error that contributes to the actor's training. For more information regarding the algorithm, please refer to [49].

Both the actor and the critic use neural networks to approximate their respective functions. For the neural network structures used for the agent and the exact training parameters, see Appendix B.

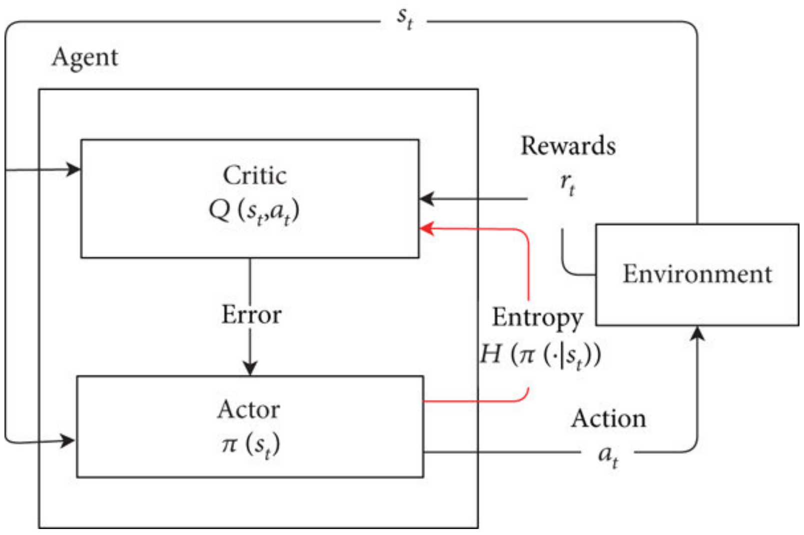

Figure 2. The SAC reinforcement learning framework [50]. The black arrows represent the general actor-critic framework. The red arrows introduce the entropy term.

4. Sim-To-Real Structure & Methodology

4.1. Test Vehicle Setup

As a vehicle platform for the agent's sim-to-real application and testing, a series production sports car (Figure 3) was used after suitable modification to enable self-driving. It was powered by a 3.0-liter twin-turbocharged straight-six engine that produced 302 kW (411 LE) between 5230 and 7000 RPM and 550 Nm torque between 2350 and 5230 RPM. The car had rear-wheel drive with an electronically controlled differential lock, which is essential for drifting [51]. The vehicle had a 7-speed dual-clutch transmission, and the 0–100 km/h acceleration time was 4.2 s. Performance sport tires were installed in sizes 245/35 ZR19 at the front and 265/35 ZR19 at the rear axle.

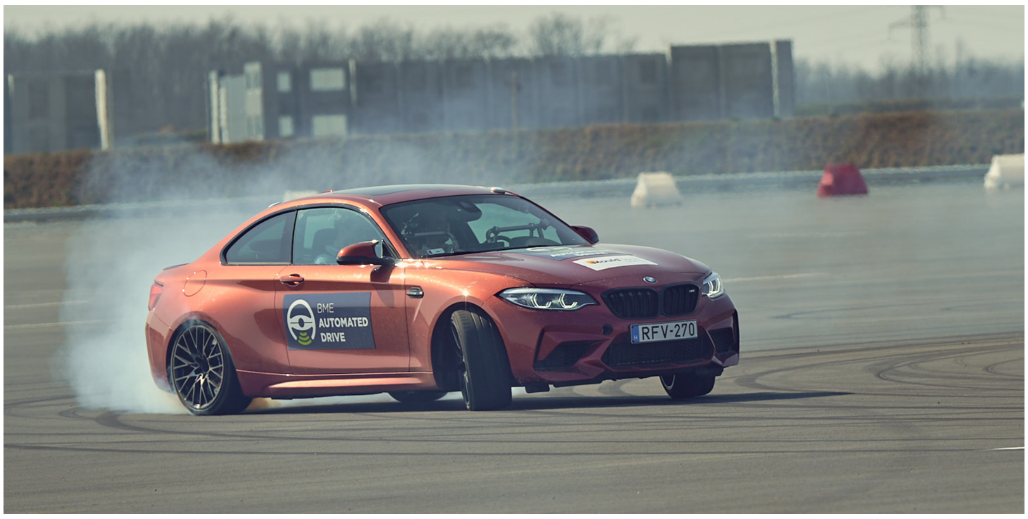

Figure 3. The author's test vehicle while performing the drift maneuvers presented in this paper.

The longitudinal position of the vehicle body Center of Gravity (CoG) was specified by measuring the individual axle weights with all the measurement equipment and two operators, which are needed currently to ensure the safety and handling of the measurement system. This results in 925 kg on the front and 895 kg on the rear axle, which gives a 1.32 m CoG longitudinal distance behind the front axle with the 2.69 m long wheelbase.

For the agent implementation, data acquisition, and controlling the actuators, dSpace microAutoBoxII real-time hardware was used, connected to a Raspberry PI 4 Model B

(Figure 4). The latter was needed because the real-time hardware does not support neural network inference in the form of a Simulink model block; thus, an outside control unit was required. The neural network inputs (state) and outputs (action) were transferred through a CAN (Controller Area Network) bus with a 50 ms sample time between the two pieces of hardware. This is identical to the sample time of the agent during training (see Table A2). A control model, developed in a MATLAB/Simulink environment, received the action signal from the Raspberry unit, and then sent it forward to the vehicle's actuator units [52]. After C code generation, it ran with a 1 ms sample time on the rapid prototyping system.

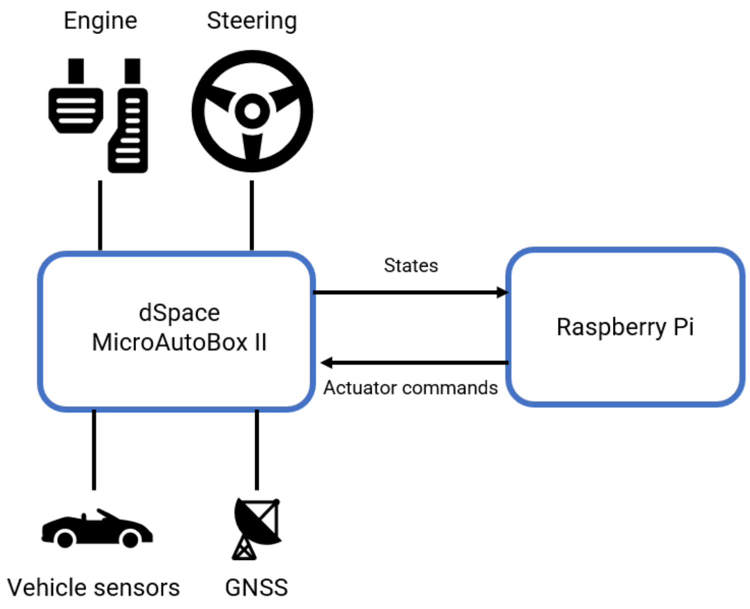

Figure 4. The test vehicle's setup with the connected hardware frame.

To enable steer-by-wire, a steering robot was installed in place of the original steering wheel. It was used in angle-control mode and communicated via CAN bus with the microAutoBoxII.

For the longitudinal motion control of the car, the accelerator pedal was removed, and its signal was emulated by the microAutoBoxII. The engine torque—i.e., the traction force—can be controlled in this way.

For the self-localization of the vehicle, a GNSS system was used, with GSM RTK correction and dual antennas. All the relevant states (including sideslip angle, yaw rate, longitudinal velocity, etc.) of the vehicle can be calculated from the measured high-frequency position and heading signals.

Due to safety reasons, signals from the vehicle CAN were received by a non-intrusive contactless sensor and were converted to FMS-Standard messages by a special CAN Gateway (FMS Gateway). These CAN signals were used during model identification and real-time control as well.

4.2. Model Parameter Identification

The vehicle model parameters described in Table A1 were identified with measurements to reproduce the car's behavior as accurately as possible. The above-described tire models need the front and rear tires' cornering stiffness and friction factors. The values were specified with a ramp steer maneuver (ISO 13674-2) [53]. After selecting the neutral gear, the steering wheel was subjected to a slow and constant velocity ramp input.

A homogeneous, flat, dry asphalt surface, the 300 m diameter Dynamic Platform of the ZalaZONE Automotive Proving Ground [54], was used for parameter identification and agent testing. The tire forces were calculated from the lateral dynamics equations of the one-track model with the assumption $\dot{r} = \dot{v}_y = 0$. The fitting of the tire model on the measurement data gives a friction coefficient of 1 for both tires and 300,000 N/rad cornering stiffness for the front and 500,000 N/rad for the rear tires. As it reveals, the positive understeer gradient results in the understeer behavior of the car.

One of the control inputs from the agent is the accelerator pedal position acc_{ped}, which can be emulated in the control model into an analog signal for the vehicle. However, the vehicle model used for training requires the longitudinal force F_{x_r} as the actuator parameter. Therefore, the relationship between the two quantities must be analyzed to implement it in the simulation environment. The traction force was estimated during a test maneuver from the longitudinal vehicle acceleration (while estimating the air drag and rolling resistance) and the actual engine torque signal received via CAN from the engine ECU. The results were collected in the second gear, which was used during drifting. The reason for the gear selection and limitation was to provide the agent with the best possible transmission characteristics for performing the targeted medium-speed ($v_x = 10$ m/s) drift maneuver without making the action selection more complex. The build-up of the engine torque follows the demand of the accelerator pedal with a considerable lag, which adds a challenge to the control task. Moreover, the engine torque signal gives relatively accurate feedback from the traction force. The steering robot can realize a given steering wheel position, but, from the control point of view, the roadwheel position has a meaning. Therefore, the steering ratio was measured for the whole steering range. Additionally, the equivalent roadwheel angle of the bicycle model was also calculated.

4.3. The Applied Sim-to-Real Methodology

One of the biggest challenges in today's sim-to-real RL advancement is the application of reinforcement learning for tasks where the target (application) environment cannot be modeled in simulation with high precision, like between a vehicle model and a real test vehicle. However, there are some methods in the current state of the art [55] for robotics applications where the above issue is present.

In this paper, the applied sim-to-real method is domain randomization, which means that some critical parameters of the vehicle are randomized in the simulation during training to increase the robustness of the agent for these specific criteria. These are usually parameters that could change depending on unmeasurable outside circumstances (meaning they could be seen as stochastic), or their measurements are noisy or include an unknown offset. During repeated testing and taking measurements of the performance of trained agents on the test vehicle, the following domain randomization solution was constructed.

The agent was trained on a range of traction coefficients between $\mu = 0.6$ and $\mu = 0.95$ to ensure good performance under different weather, varying location conditions, and tire wear conditions. Even though the drift target (see Equation (20)) was calculated for $\mu = 0.95$, the agent was able to learn to stabilize the vehicle in a relatively close equilibrium to the target state for differing traction coefficients within the above range.

High-frequency white noise was added to the received state variables to model the measurement noise from the GNSS signal (mirroring the measurement uncertainties and value fluctuation/delay coming from the sensor). Also, to model CAN communication, sensor, and actuator delays, the input and output signals were further augmented with a random time delay between 0.5 ms and 20 ms.

The engine and powertrain dynamics were modeled with a 1-D lookup table with six breakpoints between 0% and 100% to convert the accelerator pedal value to engine torque between 0 Nm and 550 Nm. The values of the breakpoints were randomized between training episodes in a range based on measurement data from the test vehicle (for substituting the complex but not completely modeled characteristics of the engine). Also, linear transfer functions were added to the input signals to imitate the real transfusion between the agent's demand and the current state of the actuators.

5. Results

The following results show the trained agent's performance in simulation and with the real test vehicle. In Figure 5 on the left side, a scope of a simulation (blue) and a measurement (yellow) instance is shown from the point of the agent's inference until a 10 s termination mark. For the simulation results shown, the randomized values (e.g., noise, engine characteristics) were set to their mean values, except for the traction coefficient, which was $\mu = 0.95$, appropriate to the definition of the target state. The real test results shown (named as *measurement*) were taken on a dry asphalt track. The simulation is started with the vehicle accelerating to $v_x = 28$ km/h. In reality, the vehicle was accelerated to $v_x = 28$ km/h using a PID (Proportional Integral Derivative) controller before turning on the RL agent. The left side shows the vehicle's state variables (v_x, v_y, r) along with the sideslip angle β. The right side consists of the calculated reward signal, the actuator signals, and a secondary performance measure called $Isdrift$. The definition of this measure is

$$Isdrift(r, \beta) = \begin{cases} 1 & \text{if } r > 0 \text{ and } -35° \leq \beta \leq -10° \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

This measure indicates if the agent successfully performs a left directional drift with a sideslip angle between 10° and 35°.

On the graphs of the pedal and the steering wheel, the blue and yellow curves show the agent's demand during the simulation and the test, respectively, and the red and purple curves show the current state of the actuators in the same manner. The actual state of the pedal is determined from the current engine torque percentage, measured by the vehicle's sensors during measurement and calculated during simulation. Figure 5 clearly shows that the largest differences are in the pedal diagram, mirroring that the very complex behavior of the engine is not completely mapped in the simulation, while, at the steering wheel, the signals are close to each other, mirroring accurate mapping.

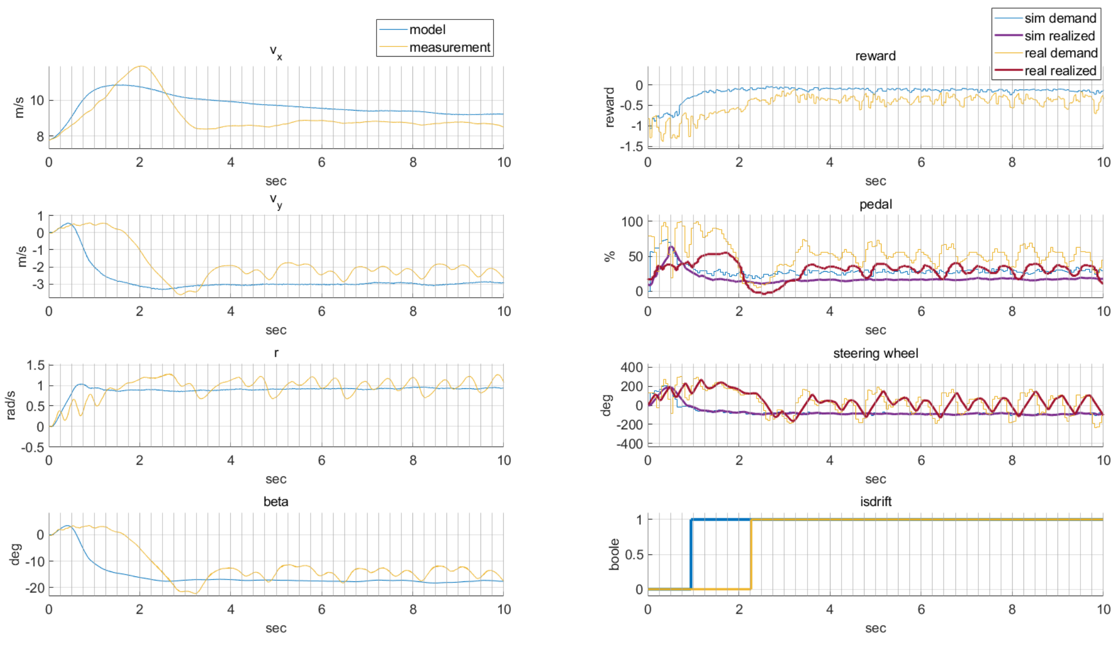

Figure 5. Comparison of the agent's performance between the simulation and the real vehicle (measurement).

Figure 6 shows the motion trajectory of the simulation results from Figure 5 (blue) and Figure 7 shows the motion trajectory of the exact measurement from Figure 5. The red part of the curve represents the (X, Y) coordinates of the vehicle (recorded from the

GNSS signals) when $Isdrift = 0$, and the green part indicates $Isdrift = 1$. The blue arrows point the direction of the vehicle's heading angle along the motion trajectory. The black dots mark every 0.25 s of time since the start of the measurement, identically to the vertical grid lines in Figure 5.

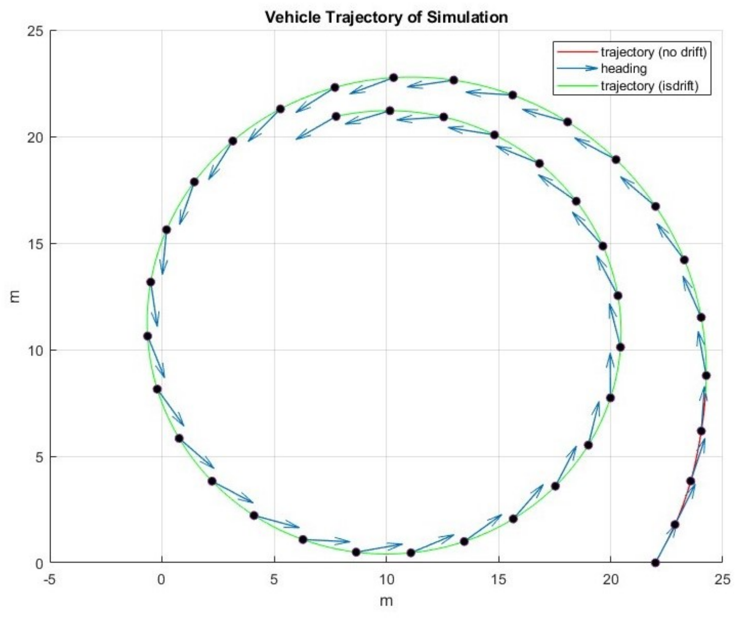

Figure 6. The motion trajectory and the direction of the vehicle's heading when the agent is performing in simulation. This is the desired outcome of the defined drift objective in (24).

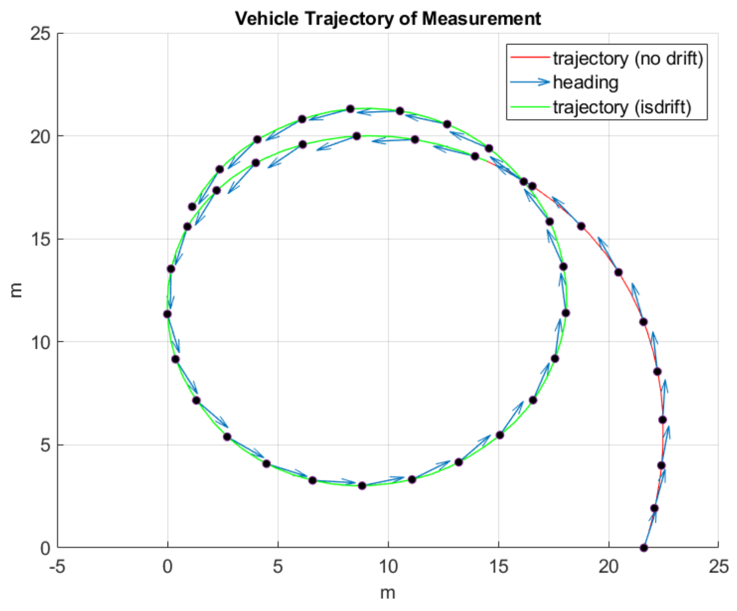

Figure 7. The measured motion trajectory and the direction of the vehicle's heading produced by the agent performing on the real test vehicle (Figure 1).

6. Discussion

As a reminder for the discussion of the results, the goal of the RL agent was to perform an approximate $\beta = -20^\circ$ sideslip angle drift at around a speed of $v_x = 10$ m/s. This goal was formulated with a drift equilibrium target that was defined in Equation (20).

The simulation results show that the agent was successfully trained in the virtual environment. It performed the desired drift smoothly and efficiently, initiating and stabilizing the maneuver within 3 s. In addition, the applied domain randomization technique helped the agent to learn enough information to perform drifting during the real test as well.

Figure 7 represents well the evolution of this drift on the real vehicle. On the right bottom side, the car starts with a normal left turn without drifting because the blue arrows are directed to the right from the red trajectory line. At first, at (x = 12.5 m; y = 19 m), the arrow changes its direction to the left side of the trajectory curve, demonstrating that the car started the drift. The same moment is evident in Figure 5 where the yellow $Isdrift$ line changes its value from 0 to 1 at 2.2 s, similarly to the color change of the trajectory in Figure 7. This proves that the proposed reinforcement learning-based agent could initiate the drift state on a real vehicle. After this critical, drift-starting point, the arrows in Figure 7 are continuously on the left side of the remaining trajectory (in Figure 5, the yellow (real) $Isdrift$ indicator is continuously 1 until the end of the complete test episode), proving that the reinforcement learning-based agent could maintain continuous (stabilized) drifting on a real vehicle.

In Figure 5, some discrepancies between the simulation and the real data can be seen. These are due to the limitations of the current sim-to-real domain randomization and vehicle model, which do not represent the real conditions completely. The lateral velocity and the yaw moment reach the target values more slowly in the real environment, indicating that the engine dynamics could be implemented more accurately to move the simulated and the real environment closer to each other. Also, this affects the consistency of the sideslip angle and the time required to reach the drift domain. Between Figures 6 and 7, it can also be seen that this results in a smaller circle of drifting.

Considering expert human driver performance, based on [14], the agent handles the actuators and produces a similar motion trajectory compared to the human driver. Furthermore, the agent achieves a more consistent sideslip angle than the expert driver, which is essential to apply such techniques to collision avoidance.

7. Conclusions

The paper's goal was to show that it is possible to train RL agents to perform steady-state drifting, a challenging maneuver, on a real vehicle. The reason is to provide the groundwork for applying these agents for more complex tasks involving extreme maneuvers performed at the limits of handling, like collision avoidance. The presented methodology and results prove that RL agents can be trained in simulation only to perform drifting on real vehicles. This indicates that these agents can be trained to perform driving tasks involving collision avoidance and stability control with further improvements.

Future Research

Considering future work, there may be an improvement in performance with a remodeling of the complex engine dynamics, so that the characteristics of the pedal actuator match the real world even more. Extending the domain randomization further on the selected elements (and involving new ones, like the tire characteristics) would make the agent even more robust.

Another idea would be to take further measurements with the current agents, and then use the recorded data to further train the agent with mixed, "supervised-reinforcement" learning. This might be the more beneficial approach for the current use case, although it is unknown how much measurement data it would require to maximize the performance. Next, reinforcement learning could be based directly on these hybrid real–simulated measurements or only on real measured data beyond the currently proposed sim-to-real case.

Experimenting in low-traction ($\mu < 0.6$) environments is also an important factor of future research, especially considering the possible robustness of these RL agents in such cases. Designing an agent that can control stability and perform extreme maneuvers in broad-scale changing grip conditions will be essential to improve road safety. To assist these capabilities, road traction coefficient estimation can also be considered [56].

Venturing even further, the main vision is to extend the agent's capabilities to more complex drift tasks, like trajectory following and collision avoidance. Overactuated vehicle control (like active suspension and decoupled drive control) is also in the scope of future research [16].

Author Contributions: Conceptualization, S.H.T., Z.J.V. and Á.B. and Z.S.; methodology, S.H.T., Z.J.V. and Á.B.; software, S.H.T. and Á.B.; validation, S.H.T., Z.J.V. and Á.B.; formal analysis, S.H.T., Z.J.V. and Á.B.; investigation, S.H.T., Z.J.V. and Á.B.; resources, Á.B. and Z.S.; data curation, S.H.T., Z.J.V. and Á.B.; writing—original draft preparation, S.H.T.; writing—review and editing, S.H.T., Z.J.V., Á.B. and Z.S.; visualization, S.H.T., Z.J.V., Á.B. and Z.S.; supervision, Á.B. and Z.J.V.; project administration S.H.T., Z.J.V., Á.B. and Z.S.; funding acquisition, S.H.T., Z.J.V., Á.B. and Z.S. All authors have read and agreed to the published version of the manuscript.

Funding: The research included in the publication, which was carried out by the HUN-REN Institute of Computer Science and Control, John von Neumann University, and the Budapest University of Technology and Economics, was supported by the Ministry of Innovation and Technology and the National Research, Development, and Innovation Office within the framework of the National Laboratory of Autonomous Systems (RRF-2.3.1-21-2022-00002). The project supported by the Doctoral Excellence Fellowship Programme (DCEP), which is funded by the National Research Development and Innovation Fund of the Ministry of Culture and Innovation and the Budapest University of Technology and Economics, under a grant agreement with the National Research, Development, and Innovation Office (2020-2.1.1-ED-2023-00239). Supported by the ÚNKP-23-3-I-BME-339 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development, and Innovation Fund.

Data Availability Statement: The datasets presented in this article are not readily available because the data are part of an ongoing study.

Acknowledgments: The preparation of the publication was realized with the support of the Board of Trustees of the Széchenyi István University Foundation; we thank them for allowing us to use the Dynamic Platform of ZalaZONE Automotive Proving Ground, which provided us with an excellent environment for performing these dynamically extreme and dangerous tests safely. We want to thank our colleague Ádám Domina for his help in implementing and validating the vehicle model, and colleagues Tamás Kazár and Zsombor Pethő for their support in preparing the real-time hardware setup.

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

Appendix A. Vehicle Model Parameters

Table A1. Constant parameter values of the vehicle model used in the simulation.

Parameter	Name	Value
m	Vehicle Weight	1810 [kg]
I_z	Inertia Constant	2500 [kg· m^2]
a	Front Wheelbase	1.35 [m]
b	Rear Wheelbase	1.37 [m]
r_w	Wheel Radius	0.32705 [m]
J	Wheel Inertia	10 [kg· m^2]
E, E_x	Lateral/Longitudinal Curvature Factor	$-1.6/-0.4$ [-]
C, C_x	Lateral/Longitudinal Shape Factor	1.2/1.15 [-]
B, B_x	Lateral/Longitudinal Stiffness Factor	0.27/25 [-]
D	Lateral & Longitudinal Peak Factor	9000·μ [-]
μ	Traction Coefficient (default) [1]	0.95 [-]
Sv, Sh	Vertical/Horizontal Shift	0 [-]
S_{xp_f}, α_{p_f}	Front Peak Slip Ratio/Angle	10.8 [-/°]
S_{xp_r}, α_{p_r}	Lateral Rear Peak Slip Ratio/Angle	7.1 [-/°]
$S_{xp_{xr}}$, $\alpha_{p_{xr}}$	Longitudinal Rear Peak Slip Ratio/Angle	0.09 [-/°]

[1] During training, the traction coefficient was an episodically randomized parameter to increase the robustness of the agents (see Section 4.3).

Appendix B. RL Agent Specifications

Appendix B.1. Neural Network Architectures

As was mentioned in Section 3, both the actor and the critic use a neural network to approximate their respective functions (the policy or the value function). The architectures of these NNs are shown in Figure A1. In the actor, the pedal and the steering wheel demands are calculated in one common and two separate fully connected layers to implement dual decision-making for the two actuators. It was tested that this architecture works better than using only common layers. The critic separates the state and action inputs of the value function and then uses a concatenation layer to return a single number.

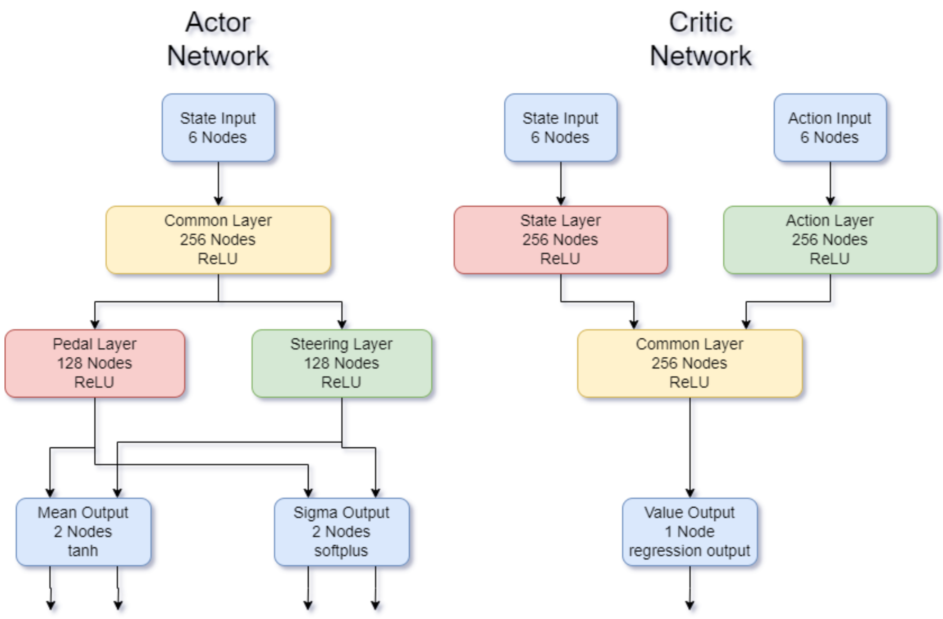
Actor Network
Critic Network
State Input 6 Nodes
Common Layer 256 Nodes ReLU
Pedal Layer 128 Nodes ReLU
Steering Layer 128 Nodes ReLU
Mean Output 2 Nodes tanh
Sigma Output 2 Nodes softplus
State Input 6 Nodes
Action Input 6 Nodes
State Layer 256 Nodes ReLU
Action Layer 256 Nodes ReLU
Common Layer 256 Nodes ReLU
Value Output 1 Node regression output

Figure A1. The trained agent's actor–critic neural network architecture.

Appendix B.2. Training Parameters

Table A2. Hyperparameter values of the RL agents used in inference.

Parameter	Name	Value
γ	Discount Factor	0.95 [1]
α_l	Neural Network Learning Rate	0.001
T_{sample}	Agent Sample Time	0.05 [s]
n_{TD}	Temporal-Difference Parameter of Foresight	18 [1]
H'	Target Entropy	−2
α_ε	Entropy Weight Learning Rate	0.003
E_{size}	Experience Buffer Size	10,000
M_{size}	Mini-Batch Size	64

[1] Parameter tuned with Bayesian optimization [57].

References

1. *SAE J3016;* Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. SAE International: Warrendale, PA, USA , 2018.
2. Ahmed, H.U.; Huang, Y.; Lu, P.; Bridgelall, R. Technology developments and impacts of connected and autonomous vehicles: An overview. *Smart Cities* **2022**, *5*, 382–404. [CrossRef]

3. Wang, G.G.; Shan, S. Review of metamodeling techniques in support of engineering design optimization. In Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Philadelphia, PA, USA, 10–13 September 2006; American Society of Mechanical: New York, NY, USA, 2006; Volume 4255, pp. 415–426.
4. Zhu, Z.; Tang, X.; Qin, Y.; Huang, Y.; Hashemi, E.A survey of lateral stability criterion and control application for autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 10382–10399. [CrossRef]
5. Tang, Y.; Tao, L.; Li, Y.; Zhang, D.; Zhang, X. Estimation of Tire Side-Slip Angles Based on the Frequency Domain Lateral Acceleration Characteristics Inside Tires. *Machines* **2024**, *12*, 229. [CrossRef]
6. Krmela, J.; Krmelova, V. Tire casings and their material characteristics for computational modeling of tires. In Engineering for Rural Development: 16th International Scientific Conference, Jelgava, Latvia, 24–26 May 2017; University of Agriculture: Jelgava, Latvia, 2017; pp. 230–235.
7. Furlan, A.D.; Kajaks, T.; Tiong, M.; Lavallière, M.; Campos, J.L.; Babineau, J.; Haghzare, S.; Ma, T.; Vrkljan, B. Advanced vehicle technologies and road safety: A scoping review of the evidence. *Accid. Anal. Prev.* **2020**,*147*, 105741. [CrossRef] [PubMed]
8. Zhao, T.; Yurtsever, E.; Rizzoni, G. Justifying emergency drift control for automated vehicles. *IFAC-PapersOnLine* **2022**, *55*, 141–148. [CrossRef]
9. Heydrich, M.; Ricciardi, V.; Ivanov, V.; Mazzoni, M.; Rossi, A.; Buh, J.; Augsburg, K. Integrated braking control for electric vehicles with in-wheel propulsion and fully decoupled brake-by-wire system. *Vehicles* **2021**, *3*, 145–161. [CrossRef]
10. Medina, A.; Bistue, G.; Rubio, A. Comparison of typical controllers for direct yaw moment control applied on an electric race car. *Vehicles* **2021**, *3*, 127–144. [CrossRef]
11. Voser, C.; Hindiyeh, R.Y.; Gerdes, J.C. Analysis and control of high sideslip manoeuvres. *Veh. Syst. Dyn.* **2010**, *48* (Suppl. S1), 317–336. [CrossRef]
12. Bárdos, Á; Domina, Á; Szalay, Z.; Tihanyi, V.; Palkovics, L. MIMO controller design for stabilizing vehicle drifting. In Proceedings of the 2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo), Szeged, Hungary, 14–16 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 000187–000192.
13. Bárdos, Á; Domina, Á; Tihanyi, V.; Szalay, Z.; Palkovics, L. Implementation and experimental evaluation of a MIMO drifting controller on a test vehicle. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1472–1478.
14. Velenis, E.; Katzourakis, D.; Frazzoli, E.; Tsiotras, P.; Happee, R. Steady-state drifting stabilization of RWD vehicles. *Control. Eng. Pract.* **2011**, *19*, 1363–1376. [CrossRef]
15. Peterson, M.T.; Goel, T.; Gerdes, J.C. Exploiting linear structure for precision control of highly nonlinear vehicle dynamics. *IEEE Trans. Intell. Veh.* **2022**, *18*, 1852–1862. [CrossRef]
16. Goel, T.; Goh, J.Y.; Gerdes, J.C. Opening new dimensions: Vehicle motion planning and control using brakes while drifting. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 560–565.
17. Baars, M.; Hellendoorn, H.; Alirezaei, M. Control of a scaled vehicle in and beyond stable limit handling. *IEEE Trans. Veh. Technol.* **2021**, *70*, 6427–6437. [CrossRef]
18. Czibere, S.; Domina, Á.; Bárdos, Á.; Szalay, Z. Model predictive controller design for vehicle motion control at handling limits in multiple equilibria on varying road surfaces. *Energies* **2021**, *14*, 6667. [CrossRef]
19. Domina, Á.; Tihanyi, V. LTV-MPC approach for automated vehicle path following at the limit of handling. *Sensors* **2022**, *22*, 5807. [CrossRef]
20. Xu, D.; Wang, G.; Qu, L.; Ge, C. Robust control with uncertain disturbances for vehicle drift motions. *Appl. Sci.* **2011**, *11*, 4917. [CrossRef]
21. Tian, X.; Yang, S.; Yang, Y.; Song, W.; Fu, M. A Multi-Layer Drifting Controller for All-Wheel Drive Vehicles Beyond Driving Limits. *IEEE/ASME Trans. Mechatron.* **2023**, *29*, 1229–1239. [CrossRef]
22. Jelavic, E.; Gonzales, J.; Borrelli, F. Autonomous drift parking using a switched control strategy with onboard sensors. *IFAC-PapersOnLine* **2017**, *50*, 3714–3719. [CrossRef]
23. Zhao, T.; Yurtsever, E.; Chladny, R.; Rizzoni, G. Collision avoidance with transitional drift control. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 19–22 September 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 907–914.
24. Li, D.; Zhang, J.; Lin, S. Planning and control of drifting-based collision avoidance strategy under emergency driving conditions. *Control. Eng. Pract.* **2023**, *139*, 105625. [CrossRef]
25. Kiran, B.R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A.A.; Yogamani, S.; Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 4909–4926. [CrossRef]
26. Tóth, S.H.; Viharos, Z.J.; Bárdos, Á. Autonomous Vehicle Drift with a Soft Actor-critic Reinforcement Learning Agent. In Proceedings of the 2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI), Poprad, Slovakia, 2–5 March 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 000015–000020.

27. Tóth, S.H.; Bárdos, Á.; Viharos, Z.J. Initiation and Stabilization of Drifting Motion of a Self-driving Vehicle with a Reinforcement Learning Agent. In Proceedings of the First Conference on ZalaZONE Related R&I Activities of Budapest University of Technology and Economics 2022, Budapest, Hungary, 31 March 2022; Budapest University of Technology and Economics: Budapest, Hungary, 2022; pp. 53–57.
28. Tóth, S.H.; Bárdos, Á.; Viharos, Z.J. Tabular Q-learning Based Reinforcement Learning Agent for Autonomous Vehicle Drift Initiation and Stabilization. *IFAC-PapersOnLines* **2023**, *56*, 4896–4903. [CrossRef]
29. Cutler, M.; How, J.P. Autonomous drifting using simulation-aided reinforcement learning. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 5442–5448.
30. Deisenroth, M.P.; Fox, D.; Rasmussen, C.E. Gaussian processes for data-efficient learning in robotics and control. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *37*, 408–423. [CrossRef]
31. Bhattacharjee, S.; Kabara, K.D.; Jain, R.; Kabara, K. *Autonomous Drifting RC Car with Reinforcement Learning*; The University of Hong Kong: Hongkong, China, 2018.
32. Cai, P.; Mei, X.; Tai, L.; Sun, Y.; Liu, M. High-speed autonomous drifting with deep reinforcement learning. *IEEE Robot. Autom. Lett.* **2020**, *5* 1247–1254. [CrossRef]
33. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. Carla: An open urban driving simulator. In Proceedings of the 2017 Conference on Robot Learning, PMLR, Mountain View, CA, USA, 13–15 November 2017; pp. 1–16.
34. Samsani, S.; Bollapragada, P.S. Rapid Autonomous Vehicle Drifting with Deep Reinforcement Learning. *Int. J. Res. Appl. Sci. Eng. Technol.* **2022**, *6*, 3901–3909. [CrossRef]
35. Orgován, L.; Bécsi, T.; Aradi, S. Autonomous drifting using reinforcement learning. *Period. Polytech. Transp. Eng.* **2021**, *49*, 292–300. [CrossRef]
36. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the 2018 International Conference on Machine Learning, PMLR, Vienna, Austria, 10–15 July 2018; pp. 1587–1596.
37. Domberg, F.; Wembers, C.C.; Patel, H.; Schildbach, G. Deep drifting: Autonomous drifting of arbitrary trajectories using deep reinforcement learning. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 7753–7759.
38. Leng, B.; Yu, Y.; Liu, M.; Cao, L.; Yang, X.; Xiong, L. Deep reinforcement learning-based drift parking control of automated vehicles. *Sci. China Technol. Sci.* **2023**, *66*, 1152–1165. [CrossRef]
39. Spielberg, N.A.; Templer, M.; Subosits, J.; Gerdes, J.C. Learning policies for automated racing using vehicle model gradients. *IEEE Open J. Intell. Transp. Syst.* **2023**, *4*, 130–142. [CrossRef]
40. Jazar, R.N. *Advanced Vehicle Dynamics*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 143–147.
41. Hindiyeh, R.Y. Dynamics and Control of Drifting in Automobiles. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2013.
42. Pacejka, H.B.; Bakker, E. The magic formula tyre model. *Veh. Syst. Dyn.* **1992**, *21* (Suppl. S1), 1–18. [CrossRef]
43. Hirschberg, W.; Rill, G.; Weinfurter, H. Tire model tmeasy. *Veh. Syst. Dyn.* **2007**, *45* (Suppl. S1), 101–119. [CrossRef]
44. Gim, G.; Nikravesh, P.E. A three dimensional tire model for steady-state simulations of vehicles. *SAE Trans.* **1993**, *102*, 150–159.
45. Kumar, N.; Rao, V.V. Hyperelastic Mooney-Rivlin model Determination and physical interpretation of material constants. *Parameters* **2016**, *2*, 1.
46. Fathi, H.; El-Sayegh, Z.; Ren, J.; El-Gindy, M. Modeling and Validation of a Passenger Car Tire Using Finite Element Analysis. *Vehicles* **2024**, *6*, 384–402. [CrossRef]
47. Singh, K.B.; Taheri, S. Integrated state and parameter estimation for vehicle dynamics control. *Int. J. Veh. Perform.* **2019**, *5*, 329–376. [CrossRef]
48. Hindiyeh, R.Y.; Gerdes, J.C. Equilibrium analysis of drifting vehicles for control design. In Proceedings of the 2009 Dynamic Systems and Control Conference, Hollywood, CA, USA, 12–14 October 2009; Volume 48920, pp. 181–188.
49. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, H.; Zhou, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
50. Ali, H.; Majeed, H.; Usman, I.; Almejalli, K.A. Reducing entropy overestimation in soft actor-critic using dual policy network. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 9920591. [CrossRef]
51. Edelmann, J.; Plöchl, M. Controllability of the powerslide motion of vehicles with different drive concepts. *Procedia Eng.* **2017**, *199*, 3266–3271. [CrossRef]
52. Pethö , Z.; Kazár, T. M.; Kraudy, R.; Szalay, Z.; Török, Á. Considering PKI safety impact on network performance during V2X-based AD/ADAS function development processes. In Proceedings of the 2022 IEEE 1st International Conference on Cognitive Mobility (CogMob) Budapest, Hungary, 12–13 October 2022; pp. 000135–000140. [CrossRef]
53. *ISO 13674-2:2016*; Information and Documentation—Bibliographic References. International Organization for Standardization: Geneva, Switzerland, 2016.
54. Szalay, Z.; Hamar, Z.; Nyerges, L. Novel design concept for an automotive proving ground supporting multilevel CAV development. *Int. J. Veh. Des.* **2019**, *80*, 1–22. [CrossRef]
55. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 1–4 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 737–744.

56. Xu, Z.; Lu, Y.; Chen, N.; Han, Y. Integrated adhesion coefficient estimation of 3D road surfaces based on dimensionless data-driven tire model. *Machines* **2023**, *11*, 189. [CrossRef]
57. Snoek, J.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. *Adv. Neural Inf. Process. Syst.* **2012**, *25*. Available online: https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html (accessed on 21 April 2024).