2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI) | 978-1-6654-9704-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/SAMI54271.2022.9780803

Autonomous Vehicle Drift With a Soft Actor-critic Reinforcement Learning Agent

1st Szilárd Hunor Tóth
Department of Automotive Technologies
Budapest University of Technology and Economics
szilardhunor.toth@edu.bme.hu
Research Laboratory on Engineering & Management Intelligence
Intelligent Processes Research Group
Institute of Computer Science and Control
tothszilardhunor@sztaki.hu
Budapest, Hungary

2nd Zsolt János Viharos
Research Laboratory on Engineering & Management Intelligence
Intelligent Processes Research Group
Institute of Computer Science and Control
Budapest, Hungary
viharos.zsolt@sztaki.hu
Faculty of Economics
John von Neumann University
Kecskemét, Hungary
viharos.zsolt@gtk.uni-neumann.hu

3rd Ádám Bárdos
Department of Automotive Technologies
Budapest University of Technology and Economics
Budapest, Hungary
bardos.adam@kjk.bme.hu

***Abstract*—Self-driving vehicles have become a more and more important field in recent years. Supported by the techniques of Artificial Intelligence (AI), the current tendency of positive results in applications is making it a promising area to focus further research. Additionally, Reinforcement Learning (RL) is already proved to be an efficient approach for complex problems, e.g. robots, industrial systems and also games (like chess, Go), etc.**

Drifting is a driving technique at handling limits where the driver intentionally oversteers, with loss of traction, while driving the vehicle through the entirety of a corner. It is a very challenging control task and often results in an accident when it occurs on public roads, consequently, the efficient control of this motion is especially important in the safety of autonomous vehicles.

The paper reports novel research results whose main goal is to develop a self-driving agent for drift motion control based on vehicle simulation by Matlab/Simulink. Longitudinal and lateral velocity together with the yaw rate formed the state representation of the vehicle. The agent action space consists of two continuous actuator values: pedal ratio and roadwheel angle. The goal of the agent is twofold: first, it has to jump into a drifting state, second, it has to keep the vehicle in drift.

The simulation results show that the proposed Soft Actor-Critic (SAC) RL agent is capable of learning to approach a predetermined drift equilibrium from cornering and staying in this drift situation as well. For the training, the solution excluded using any kind of prior data, it only works with information gained from the simulation model, which is a remarkable difference from the actual state-of-the-art RL-based solutions.

***Index Terms*—vehicle drifting, vehicle motion control, reinforcement learning, soft actor-critic**

I. Introduction

The research and development of autonomous vehicles are in a strong focus today. [1] Different aspects of this field are going to establish a solid basis, including environment recognition, motion planning, and vehicle control. In the field of control, drifting with rear-wheel drive vehicles represents a significant challenge and importance, given its potential regarding possible applications.

Drifting means a cornering motion, where the driver is constantly counter-steering to maintain a high side-slip angle, usually at high speeds. It is an inherently unstable movement that most ordinary drivers are unable to control, and usually leads to the vehicle spinning uncontrollably. This maneuver is mostly seen in motorsports (rally, Formula-D, etc.), but the application of this motion has notable potential on public roads as well, mainly in having the ability to allow maneuvering in special critical situations, for avoiding accidents. According to a study from 2015 [2], in the U.S., the relative frequency of control loss-related accidents was 8.32%, making up a total number of 458,000 crashes based on 2013's GES (General Estimates System) crash reports. This is a significant number, that could be decreased by using self-control frameworks, which can maintain and adjust the unstable motions at handling limits. In addition, the increased value of automated vehicles in motorsports [3] also grants an important field for research.

Some previous examples and results for the implementation of self-drifting mirror it's worthwhile to invest research on this topic. For a steady-state problem, classic control methods using linear controllers seem to work successfully in simulation [4] [5] [6]. In each of the cases, a car with a high rear traction force was considered to be essential for achieving good performance, based on real-life observations and measurements data. Also, in addition to the simulation results, these control methods have been successful on RC Cars [7] and also on a real test car [1].

The idea of using reinforcement learning methods to solve self-drifting tasks also presents a promising direction, for example, because of a better possible generalization ability in continuously changing, various driving conditions (like road surfaces). Cutler and How [8] used Probabilistic Inference for Learning Control (PILCO) [9], a model-based policy search RL algorithm to achieve steady-state drifting in simulation and on a RC (radio controlled) vehicle. In paper [10], the drifting was specified also as a trajectory following task in the simulator CARLA, where the exact goal was to achieve high side-slip angles at high speeds. The training of the agent was done using the Soft Actor-Critic algorithm, combined with the TD3 algorithm [11] in a different work [12].

All of the above research achievements - while their results are significant - incorporated some kind of prior knowledge when training the agent: for the PILCO, the authors generated initial policies from simple models using traditional control techniques to initialize the learning, and the trajectories in CARLA were recorded by a human driver using the simulator. These indicate the challenge of how to apply reinforcement learning without any preliminary setup or knowledge to control and initiate the complex drift status of the given vehicle.

In this paper, new results on reinforcement learning aided autonomous drift are introduced without using any kind of prior knowledge. The task is the initiation and stabilization of a steady-state drifting cornering, where the target drift state was calculated by solving a system of equilibrium equations for the vehicle model. A one-track dynamic vehicle model was implemented in MATLAB/Simulink, and a Soft Actor-Critic (SAC) [13] algorithm for training was designed. The next section describes the drift state of vehicles established by equation-based solutions and supervised learning techniques followed by a paragraph describing the applied vehicle dynamic simulation model and the coupled reinforcement learning. The section about the experiments and results differentiates between ensuring the continuous drifting and the generation of drift from a simple turning maneuver. Conclusions and references close the paper.

II. Drift for Autonomous Vehicles

Basic drift definition and its state-of-the-art solutions based on theoretical models (equations) and supervised learning models are described in the following three paragraphs.

A. What is drifting?

To enter a drift state, the driver needs to give a high enough torque input for the rear wheels to increase the rear tire slip angle, so the rear end of the car can "drift" off the arc of the corner. This is why it's important to use rear-wheel-drive vehicles because they can directly increase the longitudinal forces applied to the rear wheels and saturate them. To control the resulting unstable state of the car, the driver needs to counter-steer to compensate for the high rear-slip angle, so the car can maintain the cornering motion, otherwise, it would spin out.

B. Equation-based Solutions

There are different methods to identify drifting operation points. Equation-based (or model-based) solutions involve using the vehicle model to describe the drift equilibrium points. This is done by solving a system of equilibrium equations, which come from the vehicle model.

Previous works [14] [15] [16] show methods and results for drift equilibria calculations for control methods, which are applied in the reported research as well. Also, there is a previous application of these methods using a MIMO (Multiple-Input Multiple-Output) controller [5].

C. Supervised Learning for Drift

Unlike equation-based solutions, there are examples of data-based models, which use neural networks to predict drift equilibrium points on given environmental inputs instead of calculating them from the model. Hence, these do not require a tire model for simulation.

In the paper of Acosta and Kanarachos [17], they proposed a hybrid controller consisting of an MPC and of a feedforward neural network to solve drifting in different scenarios. They used two neural network components. The first is a drift input predictor, which provides the drift references and tire parameters to the MPC for a given target body slip angle and road curvature input. The other one is a road terrain classifier, which adapts the controller references and the tire parameters for different road surfaces. The networks were trained by a human driver during driving in the simulator.

While these solutions have better generalization properties than the equation-based methods, they have a disadvantage in the performance in specific situations. In order for having the best performance, an equation-based solution was chosen.

III. Model Structure

The structure of the proposed, integrated simulation-based vehicle model with a soft actor-critic type reinforcement learning is described here.

A. Vehicle Model and Matlab/Simulink Implementation

The model used for the simulations presented in this paper is a one-track dynamics model with two different tire models. The advantage of this model is its simplicity: it ignores the roll dynamics and aerodynamics, which are far less important than the tire model [5], so the focus is on the latter's accuracy.

First, we define the equations for the longitudinal(1), lateral(2), and yaw (3) motions of the vehicle's body frame, based on the Newtonian laws [18]:

$$\dot{v}_x = \frac{1}{m} F_x + r v_y \tag{1}$$

$$\dot{v}_y = \frac{1}{m} F_y - r v_x \tag{2}$$

$$\dot{r} = \frac{1}{I_z} M_z \tag{3}$$

where v_x is the longitudinal velocity, v_y is the lateral velocity, r is the yaw rate, m is the mass of the vehicle, and I_z is the inertia coefficient. For the (1)-(3) equations, it's needed to calculate the forces applied on the wheels. In our case, we are implementing a rear-wheel drive vehicle, so the longitudinal force applied on the front wheel is zero: $F_{x_f} = 0$. The longitudinal force applied on the rear wheel (F_{x_r}) will be controlled as an input parameter by the agent as the angle of the acceleration pedal.

The main force components ("Fig. 1") applied on the vehicle can be described with the (4)-(6) equations:

$$F_x = F_{x_f} \cos\delta + F_{x_r} - F_{y_f} \sin\delta \tag{4}$$

$$F_y = F_{y_f} \cos\delta + F_{y_r} - F_{x_f} \sin\delta \tag{5}$$

$$M_z = aF_{y_f} \cos\delta + aF_{x_f} \sin\delta - bF_{y_r} \tag{6}$$

where δ is the wheel angle, and a,b are the front and rear wheelbases.

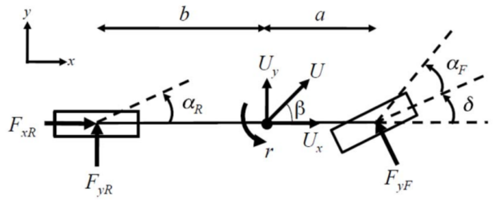

Fig. 1. The force components and slip angles of the bicycle model [15]

The lateral force components come from the tire model, for which we need the tire slip angles ((7),(8)) ("Fig. 1"):

$$\alpha_f = \arctan\left(\frac{v_y + ar}{v_x}\right) - \delta \tag{7}$$

$$\alpha_r = \arctan\left(\frac{v_y - br}{v_x}\right) \tag{8}$$

$$\beta = \arctan\left(\frac{v_y}{v_x}\right) \tag{9}$$

Also, (9) is the body frame's side slip angle.

The model of the front tires is an analytic hybrid model based on the brush tire model, which is purely a lateral slip model [16]:

$$F_{z_f} = \frac{m \cdot g \cdot b}{a + b} \tag{10}$$

$$\alpha_{sl_f} = \arctan \frac{3\mu F_{z_f}}{C_\alpha} \tag{11}$$

$$F_{y_f} = \begin{cases} -C_\alpha \tan\alpha_f + \frac{C_\alpha^2}{3\mu F_{z_f}} |\tan\alpha_f| \tan\alpha_f - \\ \frac{C_\alpha^3}{27\mu^2 F_{z_f}^2} \tan^3\alpha_f & |\alpha_f| \leq \alpha_{sl_f} \\ -\mu F_{z_f} sgn\, \alpha_f & |\alpha_f| > \alpha_{sl_f} \end{cases} \tag{12}$$

where g is the g-force, μ is the grip coefficient, and C_α is the cornering stiffness. In the case of the rear wheels, we also have a longitudinal force F_{x_r}, which would also generate longitudinal slip so a combined slip tire model is needed. This means that consideration of wheel speed dynamics is also needed if the brush tire model is being used. However, because F_{x_r} is handled here as an input, a simpler approach can be used based on the friction circle approximation as proposed in [16]:

$$F_{z_r} = \frac{m \cdot g \cdot a}{a + b} \tag{13}$$

$$\xi = \frac{\sqrt{(\mu F_{z_r})^2 - F_{x_r}^2}}{\mu F_{z_r}} \tag{14}$$

$$\alpha_{sl_r} = \arctan \frac{3\xi\mu F_{z_r}}{C_\alpha} \tag{15}$$

$$F_{y_r} = \begin{cases} -C_\alpha \tan\alpha_r + \frac{C_\alpha^2}{3\xi\mu F_{z_r}} |\tan\alpha_r| \tan\alpha_r - \\ \frac{C_\alpha^3}{27\xi^2\mu^2 F_{z_r}^2} \tan^3\alpha_r & |\alpha_r| \leq \alpha_{sl_r} \\ -\xi\mu F_{z_r} sgn\, \alpha_r & |\alpha_r| > \alpha_{sl_r} \end{cases} \tag{16}$$

Here, ξ ((14)) is a derating factor that describes the maximum achievable lateral force for the given input F_{x_r}. By including this factor into the smallest magnitude rear slip angle α_{sl_r} ((15)), the saturation is ensured similarly to the previous case. The constant values used in the vehicle model are described in "Table I".

TABLE I
CONSTANT VALUES FOR THE VEHICLE MODEL [5].

Variable	Value	Measure	Variable	Value	Measure
g	9.81	$\frac{m}{s^2}$	I_z	2500	$kg \cdot m^2$
a	1.35	m	C_α	300000	-
b	1.37	m	μ	0.95	-
m	1810	kg			

This vehicle model was implemented in Simulink with a simple linear drivetrain model, which returns engine torque between -15 Nm and 500 Nm for the pedal input, which is between 0 and 1. The gearbox is always set to be in 2nd gear. The freedom of rotation for the front wheel is 35 degrees in both directions.

B. Reinforcement Learning

Reinforcement learning (RL) algorithms use a framework where the learning is based on the interaction between an agent and an environment. The agent takes an action based on the current state of the environment and receives a reward signal based on the effectiveness of the selected action. The agent's mission is to maximize this reward signal while exploring the action space and exploring which actions give the maximum cumulative reward in the various states [19].

There are several state-of-the-art RL algorithms, which are applicable to different problems. It was needed to choose from these based on the nature of the drift problem, which has continuous state and action spaces. The Soft Actor-Critic (SAC) [13] algorithm looked to be promising in the case of performance and stability. The algorithm can handle the continuous spaces using neural networks, and it's using a stochastic actor, which grants an embedded exploration feature, said to be insensitive to hyperparameters, and has an adaptive exploration feature as well [20].

Actor-Critic methods ("Fig. 2") use neural networks (or some kind of other function approximators) to operate within a continuous environment. The actor is responsible for taking actions while the critic estimates the values of the states and produces a critique (TD-error) to update both networks, so they can maximize the cumulative expected reward. The most basic actor-critic algorithm is the DDPG (Deep Deterministic Policy Gradient) [21], which has good performance on baseline problems but has issues with value approximation.

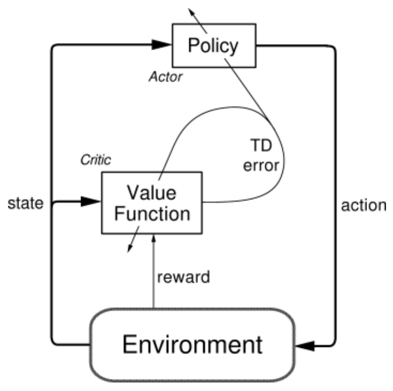

Fig. 2. Actor-Critic framework [19]

The SAC operates with two seperate Critic networks and target networks to prevent value overapproximation, like the TD3 algorithm [11], but instead of a noise model, it uses a stochastic actor to ensure the exploration. The stochastic actor estimates the μ mean and σ standard deviation parameters for a Gaussian distribution to generate an action randomly. The differental entropy measures the (partly random) exploration level of the applied agent. The goal of the agent is to maximize both the reward and the entropy [22], more precisely a weighted sum of these factors is to be maximized with self-controlled weight determination.

C. RL for Simulation-based Drift

In this section, the definitions of the required parameters for the RL problem are presented. The continuous state space was chosen to be

$$\boldsymbol{S} = (v_x, v_y, r) \tag{17}$$

which are the longitudinal velocity and the lateral velocity of the vehicle body frame and the yaw rate.

To define the target drift state for the model presented in this paper, it was needed to solve the algebraic equation system $\dot{v_x} = \dot{v_y} = \dot{r} = 0$ constructed from the equations (1) - (3) and using the first part of equation (12) and the second part of equation (16) as constraints. The last two are needed in this form because these ensure the forces on the rear tire to saturate for the drift. The system is underdefined: there are 3 equations and 5 parameters (δ, F_{x_r}, v_x, v_y, r), so two must be specified to be able to solve the system. By defining $v_x = 10\mathrm{m/s}$ and $\delta = -10°$ for this work (which means a given drift speed and a fixed steering wheel position), the following solution was received as the target drift state:

$$s_{drift} = (10\mathrm{m/s}, -3.4812\mathrm{m/s}, 0.8334\mathrm{rad/s}) \tag{18}$$

For the agent's actions, drift is achievable by just adjusting F_{x_r} through the pedal input ped_{acc} and the roadwheel angle δ by changing the steering wheel angle δ_{steer}. So the continuous action space is

$$\boldsymbol{A} = (ped_{acc}, \delta_{steer}) \tag{19}$$

These values are allowed to be in the intervalls $ped_{acc} \in [0, 1]$ and $\delta_{steer} \in [100°, -200°]$, meaning $ped_{acc} = 1$ full throthle and $ped_{acc} = 0$ full release. Also, $\delta_{steer} > 0$ is the left-hand side domain.

The defined reward function (20) is the negative of the relative eucledian distance of the state vector from the target drift state (18), so

$$r(s_t, a_t) = -\frac{1}{3}\sqrt{\sum_{i=1}^{3}(\frac{s_{t_i}}{s_{drift_i}} - 1)^2} \tag{20}$$

The neural networks of the actor and the critics have two hidden layers with 256 neurons. The hyperparameters used ensuring the best results are shown in "Table II".

TABLE II
HYPERPARAMETERS UNDER BEST TRAINING

Parameter	Value	Parameter	Value
Agent sample time	0.1	Nonlinearity	ReLU
Actor/Critic learning rate	0.001	Minibatch size	256
Entropy learning rate	$3e-4$	Optimizer	Adam
Experience buffer size	100000	Target Entropy	2500

IV. EXPERIMENTS AND RESULTS

A. Staying In The Drift Equilibrium

In the first approach, the aim was to solve a simpler version of the problem: if starting the simulation from the target drift state, whether the agent can learn to control the drift, so it can stay in very narrow proximity of the drift equilibrium. To know if the agent satisfies this condition, an indicator variable $Isdrift$ was created, which returns 1 if each state variable is in a close relative distance to the target variable at the same time, and 0 otherwise. Episodes for the learning were defined, where a simulation starts from the target state and lets the agent operate until some simulation termination time T.

The results showed that for just simply setting a relatively high T value, the agent was unable to achieve the goal, but with dynamically increasing T through consecutive training sessions, the training was successful. First, training the agent with $T = 1s$ termination time on 300 episodes, and it was able to stay in drift for around 1.5 seconds, but lost control right after. Then, the process was repeated using the weights of the last neural networks as initial weights with $T = 2s$ for 200 episodes, but the improvement was still not satisfying. After one more repetition for $T = 3s$, the agent learned enough information to control the drifting for infinite time ("Fig. 3"). The success of the agent was identified by the $Isdrift$ indicator variable and the state variables, which were close to constant. Because the agent reached the goal, no further training has been applied.

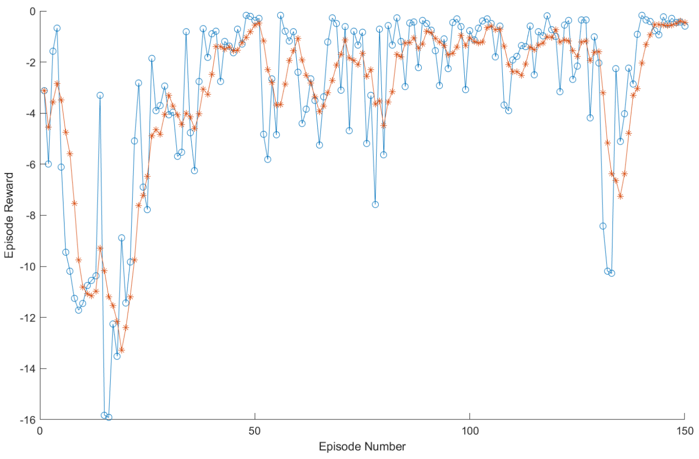

Fig. 3. Learning curve of the last training session ($T = 3$). The orange line is just the averaging of the blue episode rewards.

After these results, experiments were made on the trained agent with different starting states which are close to the target state to see how well the agent can adjust for sudden changes in the environment ("Fig. 4"). The agent performs well on a homogenous set of starting states but fails outside of it. Most of these failures above the upper bound of the yellow set are due to the physical capabilities of vehicle control, for example, if the car is already heavily spinning out, it's impossible to reverse the process.

B. *Approaching the Target Drift State from Simple Cornering*

With that in mind that the agent can learn to sustain the drift, experiments were made to see if it can learn how to enter the target drift state then stay close to it successfully. In this case the initial state was set to be $s_0 = (v_{x_0}, v_{y_0}, r_0) = (9\,\mathrm{m/s}, 0.825\,\mathrm{m/s}, 0.8334\,\mathrm{rad/s})$, which is a medium-speed simple left cornering. This is an equilibrium point which was calculated using the same method as in Section II-B, with the difference here the first part of equation (16) were used instead of the second, so the rear tire forces don't saturate.

Here a fixed termination time $T = 5s$ was set because the car needs a certain amount of time to enter drifting even if the actions applied to achieve this are perfect. After a few

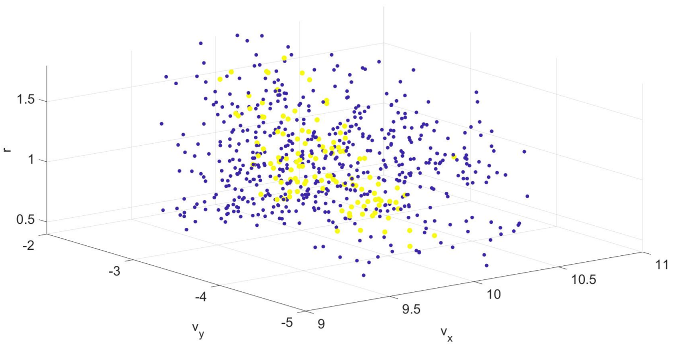

Fig. 4. A 3D scatter plot showing the initial states. Yellow means the agent adjusted the drift succesfully, and blue means the opposite. The figure represents a visible range of parameters where the agent can keep the drift state.

trials with different hyperparameters, a successful training was accomplished ("Fig. 5"), where the final resulting agent was able to reach the drift state in every case and to stay in it until the end of the episode.

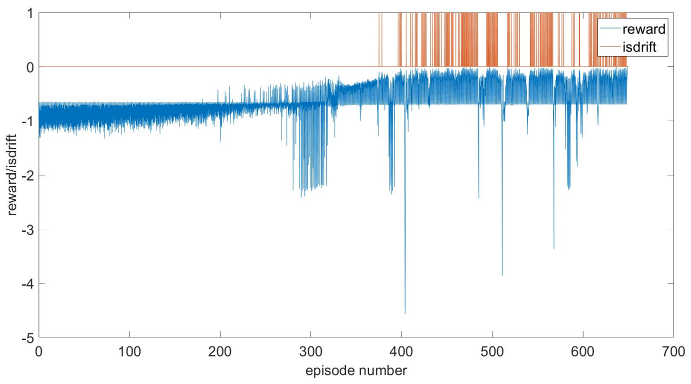

Fig. 5. The distribution of the reward signals and the indicator variable during a succesful training. We can see the frequency of drifting ($Isdrift = 1$) to increase after around episode 400.

However, when the episode length T was set to a higher value, it was found the agent loses control over the drift after a while. So the same technique was applied as in the previous case and did further training with the agent using longer episodes ($T > 10s$). The agent improved significantly, which was tested for much longer episodes, and the agent completed the task for even very high T values ("Fig. 6").

To validate the agent's performance with a detailed vehicle dynamics simulation environment, a simulation was ran in iPG CarMaker using the agent. The CarMaker provides much more realistic vehicle models and environmental physics than the presented Simulink implementation, so testing the agent in such kind of environment is critical when thinking about real-life applications. During the simulation ("Fig. 7"), after accelerating from a standing position, the agent properly approached the target drift equilibrium and successfully stabilized the motion.

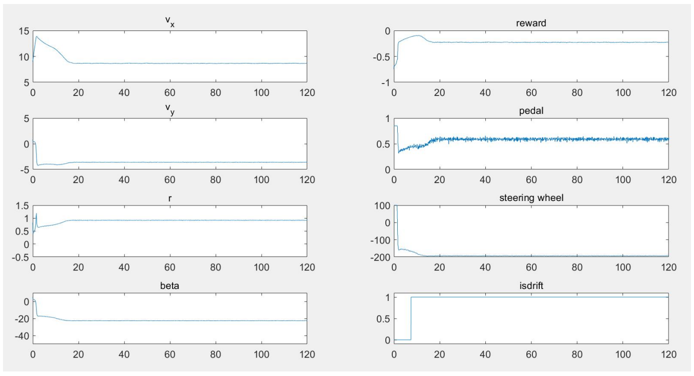

Fig. 6. The performance of the best agent under 120 seconds of simulation on a scope. It can be seen on the indicator that it jumps from 0 to 1 (the drifting starts) after around the 8th second and it stays 1 until the end of the simulation.

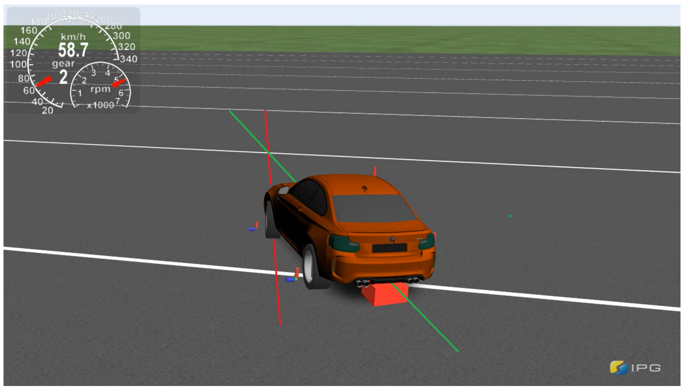

Fig. 7. The operating agent in CarMaker. The red line shows the direction of the steered wheel while the green line shows the line of the longitudinal axis of the car in its own coordinate frame. These lines are crossing ahead of the vehicle, meaning counter-steering is being applied.

V. Conclusions

The paper presented a Soft Actor-Critic agent was trained with a one-track vehicle model to approach and maintain a target drift equilibrium in a MATLAB/Simulink simulation environment. The results of this research mirror that the agent is capable of learning to control a pre-established drift situation and can also approach a target drift equilibrium from left cornering and even from a standing state. In future work, infinitely many drift equilibrium points will be tested with different road conditions. In middle term the goal is to test the approach in real conditions on a commercial car in the ZalaZONE proving ground preceded by a validation on the freely available ZalaZONE simulated environment [23].

VI. Acknowledgements

The research in this paper was partly supported by the European Commission through the H2020 project EPIC (https://www.centre-epic.eu/). In addition, the research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Autonomous Systems National Laboratory Program.

References

[1] Bárdos, Á., Domina, Á., Tihanyi, V., Szalay, Z., & Palkovics, L. (2020). Implementation and experimental evaluation of a MIMO drifting controller on a test vehicle. In 2020 IEEE Intelligent Vehicles Symposium (IV) (pp. 1472-1478). IEEE.

[2] Li, T., & Kockelman, K. M. (2016, January). Valuing the safety benefits of connected and automated vehicle technologies. In Transportation Research Board 95th Annual Meeting (Vol. 1).

[3] Yang, F. (2021, June). Research on the course, form and strategy of autonomous driving competition. In Journal of Physics: Conference Series (Vol. 1948, No. 1, p. 012093). IOP Publishing.

[4] Velenis, E., Katzourakis, D., Frazzoli, E., Tsiotras, P., & Happee, R. (2011). Steady-state drifting stabilization of RWD vehicles. Control Engineering Practice, 19(11), 1363-1376.

[5] Bárdos, Á., Domina, Á., Szalay, Z., Tihanyi, V., & Palkovics, L. (2019, November). MIMO controller design for stabilizing vehicle drifting. In 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) (pp. 000187-000192). IEEE.

[6] Czibere, S., Domina, Á., Bárdos, Á., & Szalay, Z. (2021). Model Predictive Controller Design for Vehicle Motion Control at Handling Limits in Multiple Equilibria on Varying Road Surfaces. Energies, 14(20), 6667.

[7] Baars, M., Hellendoorn, H., & Alirezaei, M. (2021). Control of a scaled vehicle in and beyond stable limit handling. IEEE Transactions on Vehicular Technology.

[8] Cutler, M., & How, J. P. (2016, May). Autonomous drifting using simulation-aided reinforcement learning. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 5442-5448). IEEE.

[9] Deisenroth, M. P., Fox, D., & Rasmussen, C. E. (2013). Gaussian processes for data-efficient learning in robotics and control. IEEE transactions on pattern analysis and machine intelligence, 37(2), 408-423.

[10] Cai, P., Mei, X., Tai, L., Sun, Y., & Liu, M. (2020). High-speed autonomous drifting with deep reinforcement learning. IEEE Robotics and Automation Letters, 5(2), 1247-1254.

[11] Fujimoto, S., Hoof, H., & Meger, D. (2018, July). Addressing function approximation error in actor-critic methods. In International Conference on Machine Learning (pp. 1587-1596). PMLR.

[12] Orgován, L., Bécsi, T., & Aradi, S. (2021). Autonomous Drifting Using Reinforcement Learning. Periodica Polytechnica Transportation Engineering, 49(3), 292-300.

[13] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018, July). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning (pp. 1861-1870). PMLR.

[14] Hindiyeh, R. Y. (2013). Dynamics and control of drifting in automobiles (Doctoral dissertation, Stanford University).

[15] Hindiyeh, R. Y., & Gerdes, J. C. (2009, January). Equilibrium analysis of drifting vehicles for control design. In Dynamic Systems and Control Conference (Vol. 48920, pp. 181-188).

[16] Hindiyeh, R. Y. (2013). Dynamics and control of drifting in automobiles. Doctor degree thesis Stanford University, 2(1), 2-1.

[17] Acosta, M., & Kanarachos, S. (2018). Teaching a vehicle to autonomously drift: A data-based approach using neural networks. Knowledge-Based Systems, 153, 12-28.

[18] Jazar, R. N. (2019). Advanced vehicle dynamics. Springer.

[19] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[20] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., ... & Levine, S. (2018). Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

[21] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[22] Ziebart, B. D., Maas, A. L., Bagnell, J. A., & Dey, A. K. (2008, July). Maximum entropy inverse reinforcement learning. In Aaai (Vol. 8, pp. 1433-1438).

[23] Gangel, K., Hamar, Z., Háry, A., Horváth, Á., Jandó, G., Könyves, B., ... & Viharos, Z. J. (2021). Modelling the ZalaZONE Proving Ground: a benchmark of State-of-the-art Automotive Simulators PreScan, IPG CarMaker, and VTD Vires. Acta Technica Jaurinensis, 14(4), 488-507.