Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 56-2 (2023) 4896–4903

Tabular Q-learning Based Reinforcement Learning Agent for Autonomous Vehicle Drift Initiation and Stabilization

Szilárd H. Tóth[1a, 2c], Ádám Bárdos[1b], Zsolt J. Viharos[2d, 3e]

[1]*Department of Automotive Technologies, Budapest University of Technology and Economics, Budapest, Hungary.*
[a]*szilardhunor.toth@edu.bme.hu*
[b]*bardos.adam@kjk.bme.hu*
[2] *Research Laboratory on Engineering and Management Intelligence, Intelligent Processes Research Group Institute of Computer Science and Control, Centre of Excellence in Production Informatics and Control, Eötvös Loránd Research Network (ELKH), Centre of Excellence of the Hungarian Academy of Sciences (MTA), Budapest, Hungary.*
[c]*toth.szilard.hunor@sztaki.hu*
[d]*viharos.zsolt@sztaki.hu*
[3] *Department of Management and Business Law, Faculty of Economics, John von Neumann University, Kecskemét, Hungary.*
[e]*viharos.zsolt@gtk.uni-neumann.hu*

Abstract: This paper aims to report on novel research results about developing a reinforcement learning agent for steady-state vehicle drift motion control. Based on the previous results of this research, the primary goal was to eliminate the problems causing learning instability experienced with the Soft Actor-Critic (SAC) algorithm applying Tabular Q-learning in this work. Trained in a MATLAB/Simulink-based simulation environment, the resulting agent succeeded in this task while being able to smoothly operate the vehicle to achieve and retain the desired target drift state, regardless of the discreet nature of the algorithm used for solving an inherently continuous task.

Keywords: Motion Control, Autonomous Vehicles, Vehicle Dynamics Control, Reinforcement Learning, Vehicle Drifting, Handling Limits

1. INTRODUCTION

Performing special driving techniques such as drifting can be challenging even for professional human drivers. However, such maneuvers can be essential to avoid certain accidents in critical road situations, such as losing traction on a slippery road surface, or executing a maneuver to avoid a sudden obstacle in front of the car (e.g., a deer) (Zhao et al., 2021).

According to Li et al. (2016), the relative frequency of control loss-related crashes in the US was around 8.3% and accidents connected with running over an animal or a pedestrian sum up to 6.69% based on GES (General Estimates System) crash records from 2013. These significant numbers indicate the importance of focusing on such driving assistance which can help to avoid these unfortunate situations.

With the rapid improvement of artificial intelligence (AI) technologies, autonomous vehicles (AV) are becoming more effective at solving the above-described realistic scenarios to improve road safety, but there is still a lot of work ahead to provide a reliable solution in practice. A reasonable first step is to focus on solving steady-state drift problems, which means initiating and holding a pre-defined target drift state. This requires the kind of precision essential to complete more complex, practical tasks, but in a less sophisticated environment.

Researching steady-state drift problems with linear control methods started at the beginning of the previous decade (Voser et al., 2010; Velenis et al., 2011), but not later than ten years later that a MIMO controller was successfully applied in a real-world environment (Bárdos, 2020). Also, Model Predictive Controls (MPCs) were considerably successful even in handling more complex, trajectory-following tasks and changing environmental conditions (Czibere et al., 2021; Domina & Tihanyi, 2022). Furthermore, supervised neural network-based hierarchical control solutions have also been applied and evaluated on RC cars (Acosta & Kanarachos, 2018; Yang et al., 2022).

Besides the methods mentioned above, reinforcement learning (RL) can have promisingly better performance and adaptability in the case of changing environmental conditions and on-line learning, making it a considerable approach for developing self-driving agents. In the case of the current state of the art applications, deep reinforcement learning (DRL) is by far the most accepted and widely used method so far (Kiran et al., 2021). The main reason behind this is the complex nature of automotive control in general, which requires an approximation method precise enough to operate the vehicle in the continuous environment. The works of Cutler & How (2016), Bhattacharjee et al. (2018), Cai et al. (2020) and Orgován et al. (2021) show that both model-based and model-free DRL can solve simple and more complex drift problems, even with added stochastic elements in the environment. Further enhancing these results, Domberg et al. (2022) introduced an agent which is viable to drift along arbitrary trajectories, showing the assumed generalization abilities of

10.1016/j.ifacol.2023.10.1261

RL. Previous works (Tóth et al., 2022a, b) lay further groundwork for steady-state drifting, although it was found later by the authors that the applied Soft Actor-Critic (SAC) algorithm – with its differential entropy-based exploration policy – is more susceptible to instability than anticipated before. This had a significantly negative impact on the robustness of the agent, like catastrophic forgetfulness, so a solution for this is introduced in this paper, using simple tabular Q-learning.

This paper contributes to the field of autonomous drifting by introducing a tabular Reinforcement Learning (RL) based, purely discrete self-operating agent with a novel, adaptive exploration policy to perform steady-state drifting in a MATLAB/Simulink simulation environment. Although the task is inherently continuous, the proposed agent learnt to achieve its goal while operating the vehicle smoothly enough to show promising performance for further development and future real-life applications.

2. THEORETICAL BACKGROUND

In the case of steady-state drifting, the main objective is to reach a pre-defined drift equilibrium point in the vehicle's state space, which usually contains the longitudinal (v_x) and lateral (v_y) velocities along with the yaw rate (r) (Fig. 1.). For a vehicle with three degrees of freedom, the Newtonian laws of motion (Jazar, 2019)

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

$$\dot{v_y} = \frac{1}{\mathrm{m}} F_y - r v_x \tag{2}$$

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

define that if a point is an equilibrium, then

$$\dot{v_x} = \dot{v_y} = \dot{r} = 0 \tag{4}$$

must apply. With added tire force saturation constraints, drift equilibrium points can be computed by solving this (4) system of algebraic equations (Hindiyeh & Gerdes, 2009).

Besides steady-state problems, drifting can be defined as other objectives, like encouraging the vehicle to achieve a side slip angle (β) as high as possible while following a desired trajectory (Cai at al., 2020; Orgován et al., 2021; Domberg et al., 2022). In these cases, the resulting agent is trained to perform spectacular stunts worth using for autonomous competitions (Yang, 2021), however, in critical traffic scenarios, precisely attaining the desired side slip angles along the trajectory is a more important factor, which is the main reason this research is focusing on steady-state drifting.

2.1 Vehicle Model

The model used for the MATLAB/Simulink simulation environment is a rear wheel drive (RWD) one-track planar vehicle model with a brush tire model on the front wheel (Jazar, 2019) and a combined-slip tire model on the rear wheel (Hindiyeh, 2009, 2013). The main describing equations for the vehicle model are the following:

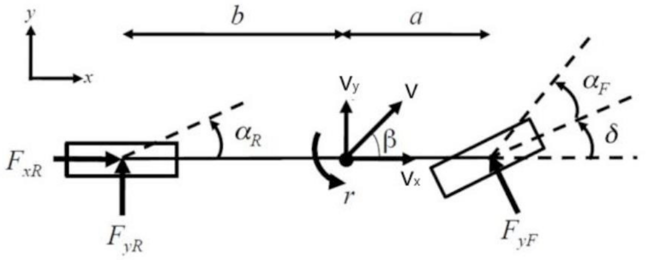

Fig. 1. The force components and tire slip angles of the presented one-track vehicle model. (Hindiyeh, 2009)

$$F_x = F_{x_r} - F_{y_f} \sin\delta \tag{5}$$

$$F_y = F_{y_f} \cos\delta + F_{y_r} \tag{6}$$

$$M_z = aF_{y_f} \cos\delta - bF_{y_r} \tag{7}$$

where the rear longitudinal wheel force (F_{x_r}) and the front wheel angle (δ) are the input parameters of the model and the lateral wheel forces for the front (8) and the rear (9) are

$$F_{y_f} = \begin{cases} -C_\alpha \tan\alpha_f + \frac{C_\alpha^2}{3\mu F_{z_f}} |\tan\alpha_f| \tan\alpha_f \\ \quad - \frac{C_\alpha^3}{27\mu^2 F_{z_f}^2} \tan^3\alpha_f & if\ |\alpha_f| \le \alpha_{sl_f} \\ -\mu F_{z_f} \operatorname{sgn}\alpha_f & if\ |\alpha_f| > \alpha_{sl_f} \end{cases} \tag{8}$$

$$F_{y_r} = \begin{cases} -C_\alpha \tan\alpha_r + \frac{C_\alpha^2}{3\xi\mu F_{z_r}} |\tan\alpha_r| \tan\alpha_r \\ \quad - \frac{C_\alpha^3}{27\xi^2\mu^2 F_{z_r}^2} \tan^3\alpha_r & if\ |\alpha_r| \le \alpha_{sl_r} \\ -\mu\xi F_{z_r} \operatorname{sgn}\alpha_r & if\ |\alpha_r| > \alpha_{sl_r} \end{cases} \tag{9}$$

respectively. The advantage of the proposed vehicle model against other representations is its computational simplicity, while it only ignores non-essential dynamics (for moderate-speed drifting), like roll and aerodynamics. For more details regarding the non-described components and variables of the model, please address the referenced works (Jazar, 2019; Hindiyeh, 2009; Tóth et al., 2022a).

2.2 The Proposed Reinforcement Learning Algorithm

The agent presented in this paper was trained with Tabular Q-learning, which is one of the first off-policy Temporal-Difference (TD) control algorithms (Watkins, 1989). This simple method has already proven its effectiveness many times in practice, e.g. in manufacturing (Viharos & Jakab, 2021) and even for autonomous driving (García Cuenca et al., 2019). Also, because of its simplicity, the method is very customizable with exploration policies.

The agent itself is an action-value function $Q: [\mathcal{S}_d, \mathcal{A}_d] \to \mathbb{R}$, which can be represented as a lookup table where every row

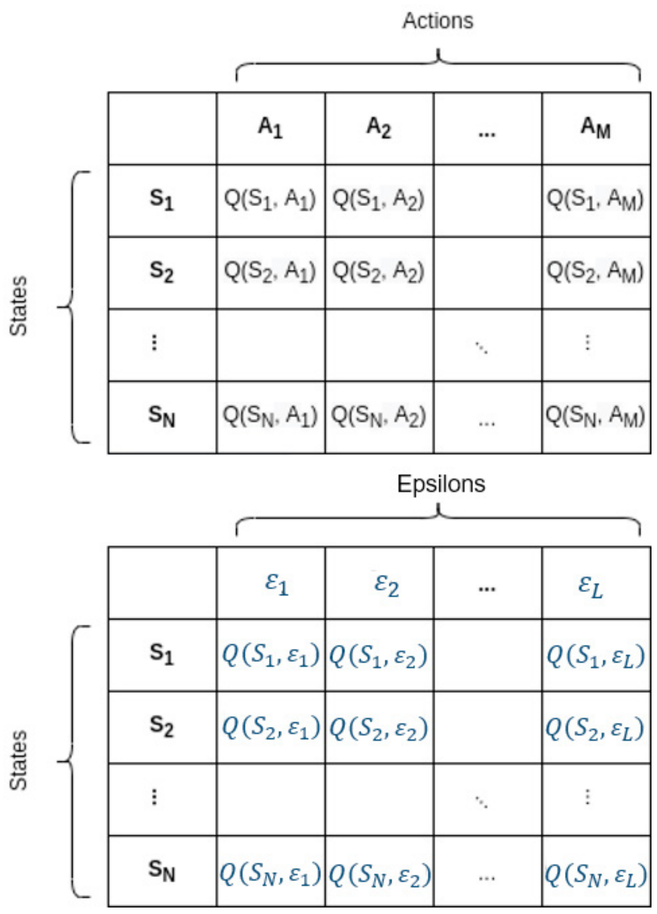

Fig. 2. The main structure of the adaptive exploration tabular Q-learning algorithm.

identifies as a (discrete) state and every column as an action. The learning is based on the TD update rule:

$$G_t = R(S_t) + \gamma R(S_{t+1}) + \cdots + \gamma^{n-1} R(S_{t+n-1}) + + \gamma^n \max_{a \in \mathcal{A}_d} Q\left(S_{(t+n)_d}, a\right) \tag{10}$$

$$Q\left(S_{t_d}, A_{t_d}\right) = Q\left(S_{t_d}, A_{t_d}\right) + \alpha\left(G_t - Q\left(S_{t_d}, A_{t_d}\right)\right) \tag{11}$$

where α is the learning rate, γ is the discount parameter, n is the parameter of foresight and $R: \mathcal{S} \to \mathbb{R}$ is the reward function of the defined reinforcement learning problem.

The algorithm was implemented and tested with two different exploration strategies. The first one is a decaying ε-greedy policy, which means that the exploration rate ε changes as

$$\varepsilon = \varepsilon\left(1 - \varepsilon_{decay}\right) \tag{12}$$

after every update step, where $0 < \varepsilon_{decay} < 1$ is a tuneable parameter. The initial value of ε is 1, and for each $Q(S, A)$ is 0 (the best possible solution). The second alternative is a custom, self-adaptive exploration policy based on the Q-learning update rule. In this case (Fig.2.), in addition to the action-value table, a separate exploration table is defined, which chooses an ε value (from a pre-defined, finite ε value set) for the agent before every action selection and is updated accordingly to the Q-learning update rule (11). A very important feature of this solution is that all states have different, own ε values that control itself dynamically during training. This novel applied self-adaptive method is already proven to be effective in manufacturing (Viharos & Jakab, 2021).

3. THE STRUCTURE OF THE PROBLEM

3.1 Properties of the Continuous Environment

The state space of the continuous environment contains the three major describing velocities:

$$\mathcal{S} := \left[v_x, v_y, r\right] \tag{13}$$

There is no need to include the side slip angle β because it can be computed directly from v_x and v_y:

$$\beta = \tan^{-1}\left(\frac{v_y}{v_x}\right) \tag{14}$$

As it was mentioned in the beginning of Section 2, it is needed to specify a target drift state, and by solving the equation system (1) with the pre-definition $v_x = 10\, {}^{m}/_{s}$ and $\delta = -10\, rad$:

$$S_{drift} = \left(v_{x_{drift}}, v_{y_{drift}}, r_{drift}\right) = \left(10\, {}^{m}/_{s}, -3.4812\, {}^{m}/_{s}, 0.8334\, {}^{rad}/_{s}\right) \tag{15}$$

Based on the vehicle model's inputs, the action space consists of the steering wheel angle $\delta_{steer} \in [-200°, 100°]$ (identical to δ) and the gas pedal position $ped_{acc} \in [0,1]$ (identical to F_{x_r}),so:

$$\mathcal{A} = [0,1] \times [-200°, 100°] \tag{16}$$

The domain of δ_{steer} is limited from $[-400°, 400°]$ to the above interval accordingly to the target drift state to reduce the number of actions for the discreet version of the action space, and to give some help to the agent in finding the optimal action values (Tóth et al., 2022a). The defined reward function is

$$R(S_t) = -\sqrt{\frac{1}{3}\sum_{i=1}^{3}\left(\frac{S_{t_i}}{S_{drift_i}} - 1\right)^2} \tag{17}$$

which is the relative mean squared distance of the current state from the target drift state. In other words, the agent's objective is to initiate and hold a drifting motion with minimal average error at a dedicated equilibrium point. In addition, an indicator metric of drift tolerance was defined as a primary metric to evaluate and monitor the agent's performance during training:

$$Isdrift(S_t) = \begin{cases} 1 & if \left|\frac{S_{t_i}}{S_{drift_i}} - 1\right| < 0.1\ \forall i \in \{1,2,3\} \\ 0 & otherwise \end{cases} \tag{18}$$

3.2 Discrete Representation of the Environment

For a discrete agent to operate in a continuous environment, a discretized representation was created for both the state and the action spaces. During operation, at each time step $t \geq 0$, the agent processes the continuous state signal coming from the vehicle model by rounding it accordingly to the finite state space $\mathcal{S}_d \subseteq \mathcal{S}$, such that

$$S_{t_d} = \underset{S_d \in \mathcal{S}_d}{\operatorname{argmin}}(|S_d - S_t|) \tag{19}$$

In other words, the R_{t+1} agent recognizes the S_t state as S_{t_d}, then chooses an action $A_{t_d} \in \mathcal{A}_d \subseteq \mathcal{A}$. An issue to address when creating $\mathcal{S}_d$ is the unboundedness of $\mathcal{S}$: to have finitely many elements, a maximum and a minimum value for each dimension must be defined. In the case of v_x, because the aim is to initiate moderate-speed drifting, $5\,{}^{m}/_{s} \leq v_x \leq 15\,{}^{m}/_{s}$ are reasonable upper and lower bounds, while $-5\,{}^{m}/_{s} \leq v_y \leq 0$ and $0 \leq r \leq 1\,{}^{rad}/_{s}$ are based on the definition of S_{drift}: too low negative side slip angles are considered equal and the positive direction is irrelevant in this case. After experimenting with several different ($\mathcal{S}_d$, $\mathcal{A}_d$) finite subset pairs, the following ones provided the best performance:

$$\mathcal{S}_d = V_{x_d} \times V_{y_d} \times Yaw_d \tag{20}$$

$$V_{x_d} = \{5\,{}^{m}/_{s}, 6\,{}^{m}/_{s}, \dots, 15\,{}^{m}/_{s}\} \tag{21}$$

$$V_{y_d} = \{-5\,{}^{m}/_{s}, -4.5\,{}^{m}/_{s}, \dots, 0\,{}^{m}/_{s}\} \tag{22}$$

$$Yaw_d = \left\{0\,{}^{rad}/_{s}, 0.1\,{}^{rad}/_{s}, \dots, 1\,{}^{rad}/_{s}\right\} \tag{23}$$

$$\mathcal{A}_d = P_d \times \Delta_d \tag{24}$$

$$P_d = \{0, 0.1, \dots, 0.9, 1\} \tag{25}$$

$$\Delta_d = \{-200, -170, \dots, -20, 0, 10, 40, 70, 100\} \tag{26}$$

Representations sparser than this resulted in an insufficient performance of the agent, and denser ones did not perform as much better considering the increase in training time. As for the reward, two options were implemented to calculate its value, either from the current vehicle state (continuous) or the current agent state (discrete).

3.3 Training Properties

The training of the agent is episodic with the initial state $S_0 = \left(9\,{}^{m}/_{s}, 0\,{}^{m}/_{s}, 0\,{}^{rad}/_{s}\right)$ and an episode last until a simulation reaches a defined simulation termination time T. The sample time of the controller is $T_\Delta = 0.1s$, which means that the agent intervenes $\frac{T}{0.1}$ times during each episode. With the help of this method the training of the agent is balanced between initiating and stabilizing the drift. As for the values of the optimal hyperparameters in each case, see Table 1.

Table 1. Optimal Hyperparameter Values

Name of parameter	ε-greedy exploration	Adaptive exploration
Learning Rate (α)	0.5	0.2
Discount factor (γ)	0.7	0.7
Parameter of foresight (n)	1	1
Episode simulation time (T)	$5s$	$8s$
Reward computation	continuous	discrete

4. RESULTS

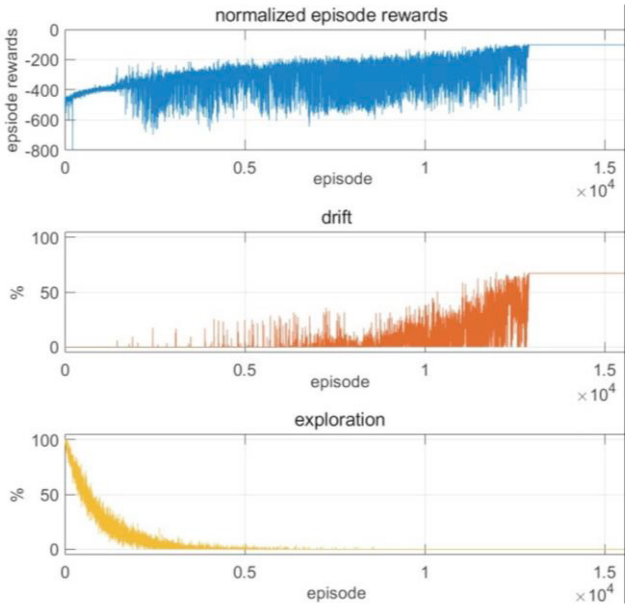

Fig. 3. Successful training with the ε-greedy strategy, with cumulative rewards earned during the episode in blue, 'isdrift' episode percentages in red, and exploration rates in yellow.

4.1 Greedy Exploration

The curves showing a successful training can be read from Fig. 3.: after 12,900 episodes, the attainable optimum was found, with which the agent is able to maintain the defined target drift through 67.26% of the 5-second episode, i.e., for 3,363 seconds. Achieving numbers higher than this in 5 seconds might be possible, although it's sure that there is an upper bound lower than 100%, because the starting position is not a drift state, and the vehicle physically needs a certain amount of time to create drifting. The exploration decay factor was $\varepsilon_{decay} = 7{*}10^{-5}$, which means that the ε exploration ratio reaches zero close to the 100.000th episode. It can be seen from the exploration rates that the interval between episode 6000 and 13000 mostly contains exploitation, though even during this phase, the agent's performance improves significantly. This is mainly due to the appropriate all-zeros initialization of the Q-table: since the global maximum of the reward function is 0 and the values of state-action pairs that have not yet been tried via exploration are all left as zeros, the algorithm will prioritize these in case of exploitation, thus providing a secondary, internal exploration feature.

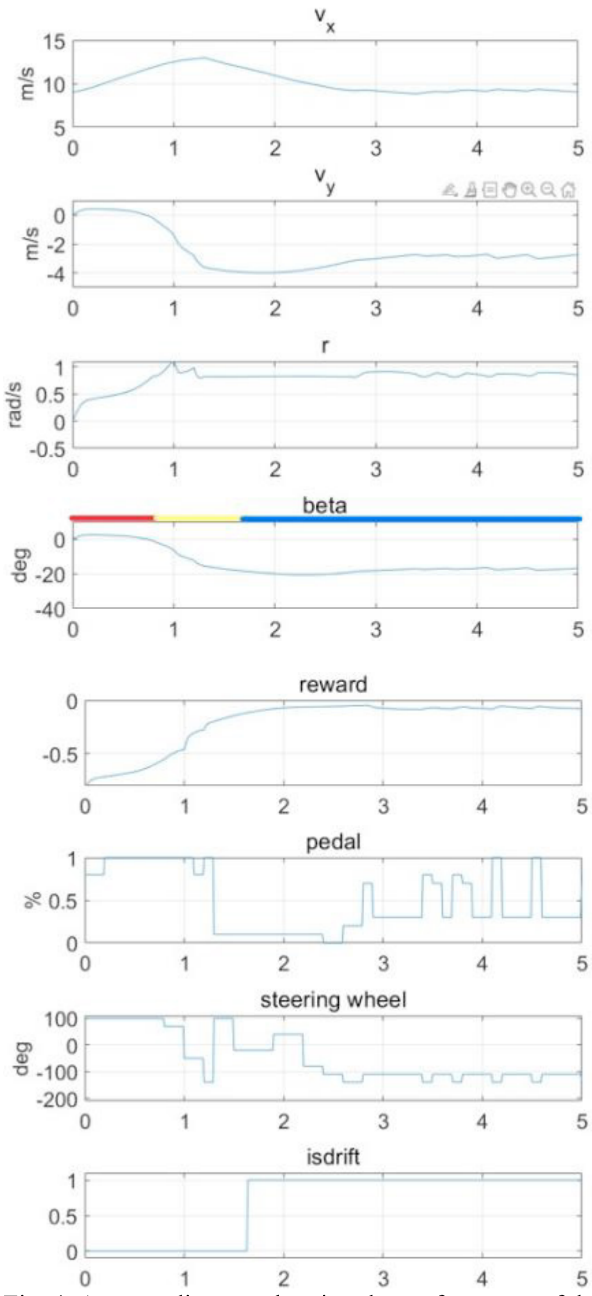

Fig. 4. A scope diagram showing the performance of the agent obtained as a result of a successful training. The 'x' axis shows the simulation time in seconds.

Fig. 4. shows a scope graph, which describes the evolution of the state variables, the reward, the indicator, and the actions delivered by the agent through an episode. It can be noticed that the drifting movement takes place long before the defined indicator function shows the target drift. This can be seen from the fact that shortly before 1 second the angle β is already in the negative range, while the value of r is still positive. Also, Fig. 5. shows the trajectory taken by the vehicle controlled by the agent in the case of a 30 second simulation. In the red part of the curve, no drifting has yet occurred, in the yellow part side slippage is already taking place, and in the blue part the indicator function is already indicating. This colour coding can also be seen on Fig. 4., above the sideslip angle curve. All this information means that the defined target state is located not only at the handling limit, but also well beyond it.

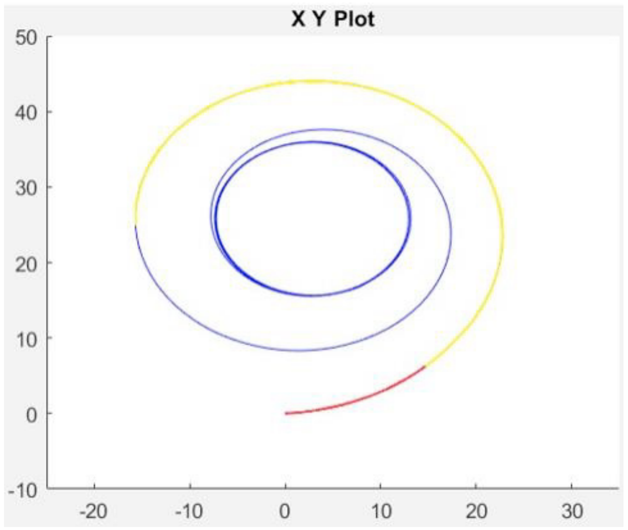

Fig. 5. The movement path of the car controlled by the agent for 30 seconds.

Furthermore, looking at the evolution of the actions, it can be concluded that the agent is able to choose them quite precisely, there is no significant "twitching", which is a surprisingly good result considering the discrete nature of the agent. Nevertheless, a more serious regulation of the dynamics of the actuators will be necessary for the purpose of solving more complex tasks and testing on a real vehicle.

In summary, the tabular agent operated with simple ε-greedy discovery successfully performed the set task, and the training instabilities experienced with the SAC algorithm were also eliminated, each training can be repeated with the same result, and the exploration strategy does not cause performance issues. However, it is important to note that finding the optimal values of the hyperparameters controlling the exploration took considerable time, which can cause difficulties in the case of a more complicated task (e.g. changing traction conditions).

4.2 Adaptive Exploration

During the experiments, the possible ε value set for the exploration agent was defined as follows:

$$E = \{0, 0.05, 0.15, 0.25, 0.5, 1\} \quad (19)$$

Initially, all values are -1's for both Q-tables. The first noticeable experience during training (Fig.6.) was that the rate of exploration oscillates between 20 and 40%, so around the average of 32.5%. Examining the evolution of the values of the exploratory Q-table during training, it can be seen in the vast majority of states that the Q-values of the epsilons are relatively close to each other throughout, and therefore their selection takes place with almost the same probability (Table 2.). After investigating, the hypothesis was formed that the main reason for this is that when choosing the probabilities, the distance of the rate values relative to 0 (best possible reward) is considered, however, in most cases, reaching this value is impossible even with optimal actions, since the reward is the distance from the working point,

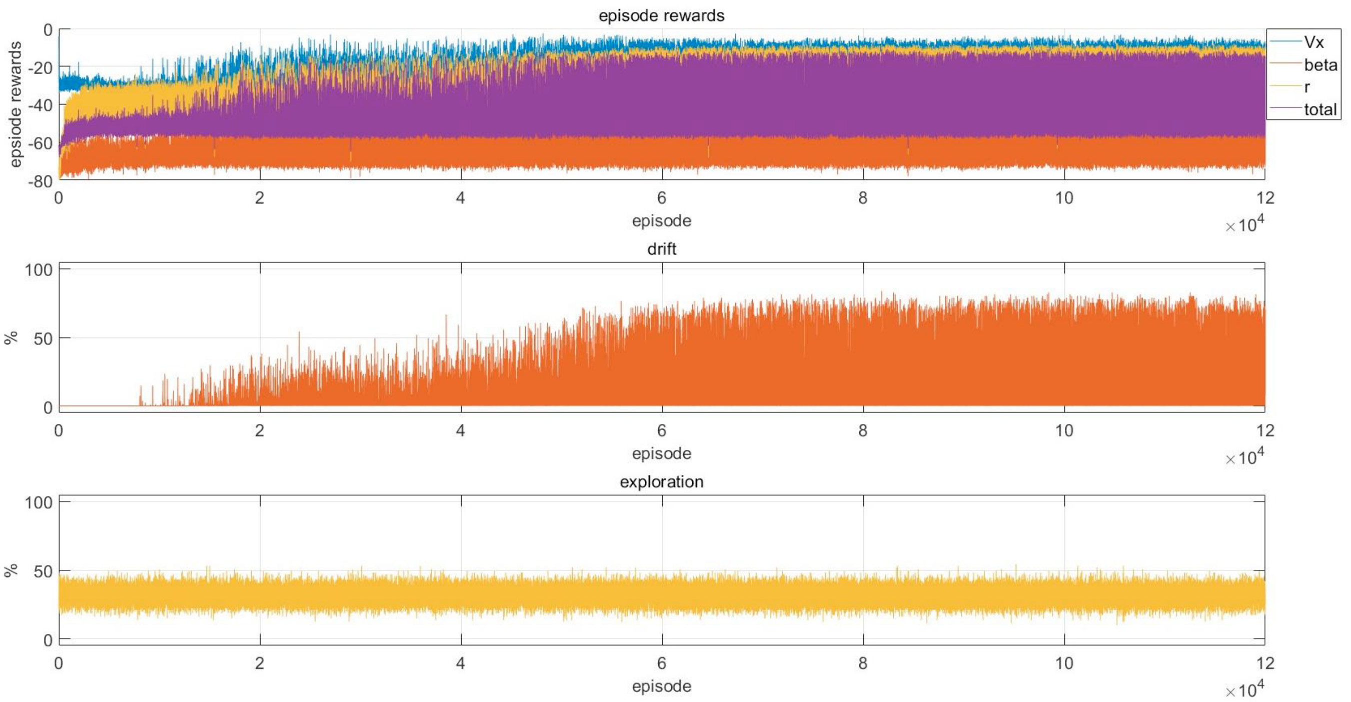

Fig. 6. Learning curves of the agent with the adaptive exploration strategy. The reward signal here has been split with respect to the state variable components in (17).

Table 2. Choosing an exploration rate

Choosing exploration:		
Epsilon value	Exploratory table value (in the given state)	Probability of selection
$\varepsilon = 0$	-0.2737	0.1704
$\varepsilon = 0.05$	-0.2857	0.1632
$\varepsilon = 0.15$	-0.2467	0.1891
$\varepsilon = 0.25$	-0.3301	0.1413
$\varepsilon = 0.5$	-0.2702	0.1726
$\varepsilon = 1$	-0.2853	0.1635
Random probability variable for selection: $p = 0.1279$		
$p < 0.1704 \rightarrow \varepsilon_{chosen} = 0$		

which is obviously nonzero for practically all states. Changing the reward calculation to continuous did not solve this issue. Despite all of this, the agent has an even better performance than with the ε-greedy exploration, with a 72.55% maximum drift ratio under the first 5 seconds (83.95% under 8 seconds) (Fig.7.). Moreover, this result has been achieved with stable, repeatable convergence, but due to the explorations that occur with a high probability on average, in a lot of cases it cannot maintain drifting for a long time. In addition, a significant "twitching" of the action values can also be experienced, which was not the case with the ε-greedy exploration. These issues will be addressed by adding regulator constraints for the actions (see Section 5.1).

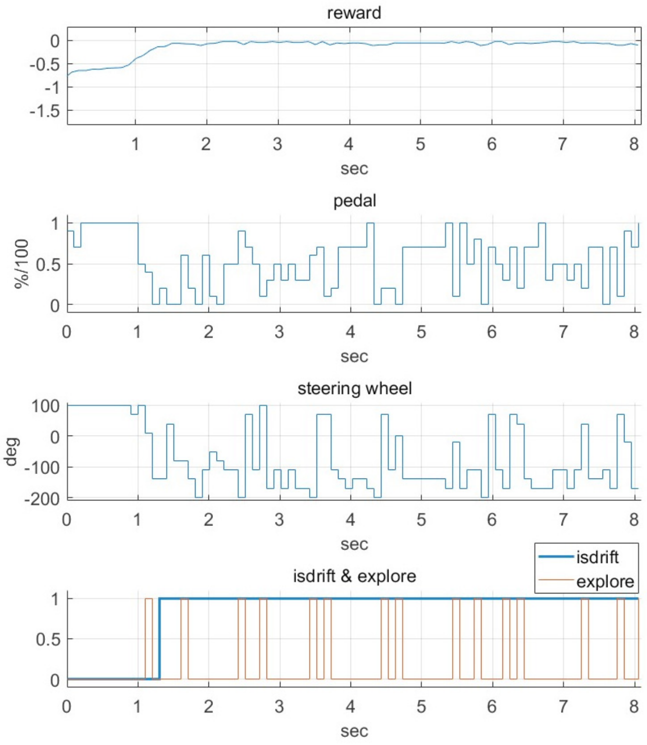

Fig. 7. Performance of the adaptively exploring agent on a scope diagram. The brown curve shows the exploration cases together with the "isdrift" values (blue).

5. CONCLUSIONS

In this paper, a control agent was developed to initiate and stabilize a steady-state drift motion in a simulation environment using tabular Q-learning with two different exploration strategies. The learned agents successfully approached the target state in both cases.

The greedy exploration also made it possible to learn the information needed to maintain the target state, though finding the correct exploration hyperparameters required a significant amount of time. Although the adaptive strategy resulted in too much exploration, the reason for this behaviour was found, and it is possible to correct it in the future. The learning instabilities experienced before with the SAC algorithm were also eliminated, every session can be repeated with the same result, and the exploration strategies do not cause critical performance issues.

The research will continue with correcting the difficulties of the adaptive algorithm and adding environmental noise and regulation dynamics for the actuators. After these trials, if the agent continues to show promising performance and robustness, testing on a real vehicle can take place on the ZalaZONE test track (https://www.zalazone.hu/). In the later stages, the goal is to develop an agent capable of performing continuous drifting through specified tracks and road sections. Another goal could be the development of an adaptive feature that successfully responds to stochastic road conditions while performing on the track.

5.1 Outlook: Applying Regulation for the Actuator Dynamics

The ongoing research works related to the regulation of actuator dynamics will be briefly presented here.

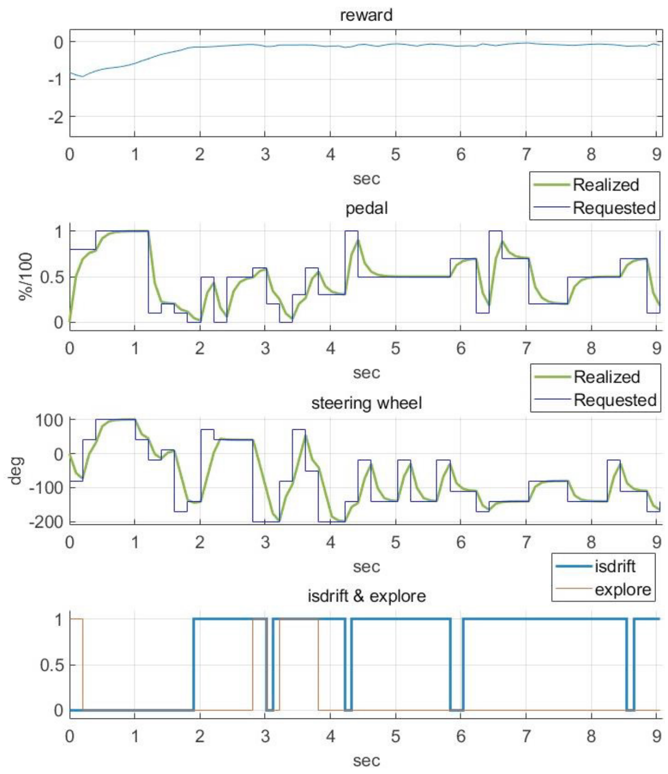

Fig. 8. The performance of the adaptive algorithm with the addition of actuator dynamics, in the case of n = 5 and a sample time of 0.2s. The green curves are the action values actually delivered for the vehicle.

Actuator control in this case means that a delaying, one-storage proportional member is added to the input signal of both actions, and a rate limiter is added to the steering wheel, thereby simulating the conditions of controlling a real vehicle. Experiments with the adaptive exploration so far show that the algorithm can adapt to conditions that make the task more difficult, but only by increasing the parameter of foresight. In some cases, it can also be observed that the limited actions only reach the target values desired by the agent if the sample time is increased accordingly (Fig. 8.). In the absence of this, it may happen that the agent plans too far in advance, thus intervening in the process prematurely, which sometimes nullifies the effect of the actions issued so far. On the other hand, increasing the sample time can have the disadvantage of losing the precision required to maintain the drift. This trade-off situation is an important part of future development to do further experiments on.

ACKNOWLEDGEMENTS

The research included in the publication, which was carried out by the Institute of Computer Science and Control and the Budapest University of Technology and Economics, was supported The research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems. (RRF-2.3.1-21-2022-00002). This research has been supported by the ED_18-2-2018-0006 grant on an "Research on prime exploitation of the potential provided by the industrial digitalisation". On behalf of Project “Comprehensive Testing of Machine Learning Algorithms” we thank for the usage of ELKH Cloud (https://science-cloud.hu/) that significantly helped us achieving the results published in this paper.

REFERENCES

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

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

Bhattacharjee, S., Kabara, K. D., Jain, R., & Kabara, K. (2018). Autonomous drifting RC car with reinforcement learning. *Dept. Comput. Sci., Univ. Hong Kong, Tech. Rep*.

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.

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.

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.

Domberg, F., Wembers, C. C., Patel, H., & Schildbach, G. (2022, May). Deep Drifting: Autonomous Drifting of Arbitrary Trajectories using Deep Reinforcement Learning. In *2022 International Conference on Robotics and Automation (ICRA)* (pp. 7753-7759). IEEE.

Domina, Á., & Tihanyi, V. (2022). LTV-MPC Approach for Automated Vehicle Path Following at the Limit of Handling. *Sensors*, *22*(15), 5807.

García Cuenca, L., Puertas, E., Fernandez Andrés, J., & Aliane, N. (2019). Autonomous driving in roundabout maneuvers using reinforcement learning with Q-learning. *Electronics*, *8*(12), 1536.

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

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).

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

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., & Pérez, P. (2021). Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*.

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).

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

Tóth, S. H., Viharos, Z. J., & Bárdos, Á. (2022, March). Autonomous Vehicle Drift With a Soft Actor-critic Reinforcement Learning Agent. In *2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI)* (pp. 000015-000020). IEEE.

Tóth, S. H., Bárdos, Á., & Viharos, Z. J. (2022). Initiation and Stabilization of Drifting Motion of a Self-driving Vehicle with a Reinforcement Learning Agent. In *The First Conference on ZalaZONE Related R&I Activities of Budapest University of Technology and Economics 2022* (pp. 53-57). Budapest University of Technology and Economics.

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.

Viharos Z. J.; Jakab, R. B. (2021). Reinforcement Learning for Statistical Process Control in Manufacturing. Measurement, Vol. 182.

Voser, C., Hindiyeh, R. Y., & Gerdes, J. C. (2010). Analysis and control of high sideslip manoeuvres. *Vehicle System Dynamics*, *48*(S1), 317-336.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, University of Cambridge.

Yang, B., Lu, Y., Yang, X., & Mo, Y. (2022, May). A hierarchical control framework for drift maneuvering of autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)* (pp. 1387-1393). IEEE.

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.

Zhao, T., Yurtsever, E., Chladny, R., & Rizzoni, G. (2021, September). Collision Avoidance with Transitional Drift Control. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)* (pp. 907-914). IEEE.