CONTROL-INFORMED NEURAL NETWORK FOR CONTROLLER SELECTION

Dániel Fényes[1], Tamás Hegedűs[1], Péter Gáspár[1,2]

***Abstract*— This paper proposes a reinforcement learning (RL)-based approach for dynamically selecting the most suitable control method according to changing operating conditions. Using the nominal model of the actual system, several feedback controllers are developed, each offering different levels of performance depending on the scenario. The RL algorithm is employed to determine and apply the optimal control strategy within a given operational range. Four control methods are investigated: Linear Parameter Varying (LPV), Ultra-local Model-based (ULM), Linear Quadratic Regulator (LQR), and a kinematic model-based controller. The performance and effectiveness of the proposed approach are assessed through three test scenarios using the high-fidelity vehicle simulation platform, CarMaker.**

I. Introduction and motivation

The control design process of arbitrary dynamical systems has been dominated by classical, linear approaches such as PID (Propotional-Integral-Derivative) [1] or Linear-Quadratic Regulator (LQR) [2], due to their well-defined stability properties. However, their performance can degrade in the presence of unmodeled dynamics and nonlinear effects, forcing researchers to develop adaptive strategies that overcome these limitations. A new method has been developed that relies on multiple nominal models, adapting to varying system dynamics and operating conditions rather than relying on just one linear model with fixed, constant parameters. The main idea was to create a set of nominal models that accurately describe the dynamics of the real system, with the control algorithm switching between these linear models during operation [3].

In the field of switching controllers several methods can be found such as the fuzzy logic controllers (FLC) [4]. However, most of the existing switching controller methods can provide satisfactory results only to specific scenarios, which highly limits their general applicability. Moreover, these methods can provide solutions with high complexity, which can make the implementation process more challenging. The main challenges within these control frameworks are: 1) to find the balance between the complexity and generalization ability, 2) smooth transition must be guaranteed between the controllers to avoid oscillations and undesired transients, which can compromise system stability [5]. Although switching times that satisfy stability requirements can be determined using methods such as the Lyapunov approach [6], identifying the actual operational point can still involve difficulties.

Neural networks have shown their capability to enhance the adaptability and performance of control systems. Moreover, neural networks often outperform classical methods in handling nonlinearities, uncertainties, and neglected or unknown dynamics. This provides the main motivation behind the approach, which is called a combined or hierarchical control design [7]. In this control structure, a neural network is integrated with a classical controller to exploit the strengths of both approaches: high-performance level and stability. Within the combined solutions, two main methods can be separated in the context of the neural network. In the first type of method, the neural network addresses the nonlinearities and uncertainties of the system during the modeling process [8], while in the second type, the neural network generates the control input, which is supervised by a classical approach [9]. These methods have been successfully applied to various control problems over the years, however similarly to the previous case, the supervisory or the classical part can have a high complexity. Moreover, in many cases, the main limitation of these methods is the difficulty in properly balancing the neural network-based part and the classical controller. Additionally, the goal of the supervisory algorithm is to guarantee stability, which indicates that in some cases, the performance level can decrease significantly.

The limitations of combined solutions motivated this paper, which aims to integrate classical control approaches with a neural network-based method at the switching mechanism level. This means, that using the classical control methods and the nominal model of the systems, different feedback controllers are designed. Despite the classical switching control method, the nominal model of the system is not varied only the design method of the controller. The main idea is to use a Reinforcement Learning (RL) technique [10], which provides the switching signal between the different controllers. The main advantage of this method is that the performances (e.g., tracking accuracy) can be formulated at a higher level while the neural network accurately determines the given operational point of the system. Moreover, the oscillations, which come from the inappropriate switching, can be eliminated through penalization of this behavior during

D. Fényes and P. Gáspár are with [1]Institute for Computer Science and Control (SZTAKI), Hungarian Research Network (HUN-REN), Kende u. 13-17, H-1111 Budapest, Hungary. E-mail: [daniel.fenyes;peter.gaspar]@sztaki.hun-ren.hu

P. Gáspár is with [2]Department for Control of Transportation and Vehicle Systems, Budapest University of Technology and Economics, Stoczek u 2., H-1111 Budapest, Hungary. E-mail: [balazs.nemeth;peter.gaspar]@kjk.bme.hu

The paper was funded by the National Research, Development and Innovation Office under OTKA Grant Agreement No. K 143599. The work of Daniel Fényes was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. The research was partially supported by the European Union within the framework of the National Laboratory for Autonomous Systems (RRF-2.3.1-21-2022-00002).

the training process. The results of this paper contribute to the field of switching control methods, with the integration of neural network-based solutions. The improved performance level of the proposed method is demonstrated through a vehicle-oriented control problem.

In the context of lateral trajectory tracking for automated vehicles, several control approaches have been developed in recent years. The main difference between these methods lies in the model with which the controller is designed. Kinematic model-based controllers rely on simplified vehicle motion equations that ignore forces and dynamics [11], making them suitable for low-speed or path-planning applications such as the Stanley controller [12]. These models often do not need vehicle-specific parameters such as cornering stiffnesses, however, the vehicle can become unstable within higher velocity ranges and may produce oscillations in sharp turns. On the other hand, dynamic model-based controllers incorporate vehicle-specific parameters such as mass, inertia, and tire forces, providing a more accurate model at higher velocities. Using these models, a Linear Quadratic Regulator (LQR) can be designed, which provides high performances for trajectory tracking [13]. The drawback of this approach is that the vehicle model may change during operation, reducing the accuracy of the nominal model, which results in a degraded tracking accuracy. Using dynamical vehicle models, Linear Time-Varying (LTV) models can be developed, which serve as a basis for designing Linear Parameter-Varying (LPV) controllers [14]. The main advantage of these methods is that this type of controller can adapt to changing parameters through a scheduling variable, with which enhanced performance level can be reached. However, in practice, accurately determining this scheduling parameter can be a challenging task. The Model-Free Control (MFC) method has been developed to overcome the limitations of the mentioned methods, which come from the inaccurate modeling and uncertainties [15]. The main idea is to estimate the so-called ultra-local model, which captures effects not considered during the modeling process. This computed ultra-local model is involved in the control loop as a control input, with which the tracking performance can be significantly enhanced. Although this modification gives an improved performance level, for the computation of the additional control input the derivatives of the output are needed, which is hardly computable due to the noises. MFC-based control structures are sensitive to noises and time delays, which limits the range of applications of this approach. In summary, it can be seen that the presented methods have advantages and also disadvantages. This suggests that different feedback controllers provide higher accuracy in different operational ranges. This motivates the main goal of the paper: to use a reinforcement learning-based algorithm to select the feedback controller that achieves the highest possible performance for a given operational range of the vehicle.

The paper is structured as follows: The applied methods and techniques are detailed in Section II. The training of the RL agent is presented in Section III. The operation and the effectiveness of the proposed method are presented through simulation in Section IV. Finally, the contribution of the paper is concluded in Section V.

II. Applied methods and techniques

In this section, the applied methods are details including the control algorithms and the reinforcement learning technique. Four control methods: Linear Parameter Varying (LPV), Linear Quadratic Regulator (LQR) feedback controller, Error-based ultra-local model-based control method, and kinematic model-based controller: Stanley.

A. Stanley

Stanley controller is a vehicle-oriented controller, designed specifically for trajectory tracking. It is based on the kinematic bicycle model, see [12].
In the first step, the heading error must be eliminated:

$$\delta(t) = \psi_e(t) \tag{1}$$

where δ is the steering angle, ψ_e is the yaw error.
In the second step, the tracking error is eliminated:

$$\delta = tan^{-1}\frac{ke(t)}{v_x(t)} \tag{2}$$

where $e(t)$ is the tracking error, k is the curvature, v_x is the longitudinal velocity.
Finally, the control signal is computed as:

$$\delta = \psi_e(t) + tan^{-1}\frac{ke(t)}{v_x(t)} \tag{3}$$

The Stanley is an effective controller for trajectory tracking especially at low longitudinal velocity.

B. Linear Quadratic Regulator (LQR)

LQR is a well-known and widely used technique in control engineering, It requires an LTI (Linear Time Invariant) state-space representation of the considered system:

$$\dot{x} = Ax + Bu \tag{4}$$

$$y = C^T x \tag{5}$$

Using weighting matrices Q and R the desired performances can be guaranteed such as tracking, or limitation of a specific state. The optimization problem can be formulated as:

$$J(x(t)) = \int_0^\infty (x_k^T Q x_k + u_k^T R u_k) \tag{6}$$

where x is the state-vector, A, B, C are the state-matrices. The optimization can be solved using the Ricatti equation, see [13].

C. Error-based ultra-local model

In the Model Free Control (MFC) structure, the ultra-local model is used to compute an additional control input, which aims to deal with the unmodelled, nonlinear behavior of the considered system, see [16]. The ultra-local model (F) consists of two signals: previous input $u(t-1)$, derivative of the measured signal $(y^{(\nu)})$ and a free tuning parameter α. An improved version of the ultra-local model is the error-based ultra-local model, see [17]. Basically, the error-based ultra-local model consists of two ultra-local models: The first one is computed from the measured signals, while the second one is from a nominal model ($y_{ref}^{(\nu)}$,F_{nom},u_{nom}):

$$y^{(\nu)} = F + \alpha u, \tag{7a}$$

$$y_{ref}^{(\nu)} = F_{nom} + \alpha u_{nom}, \tag{7b}$$

$$\underbrace{y^{(\nu)} - y_{ref}^{(\nu)}}_{e^{(\nu)}} = \underbrace{F - F_{nom}}_{\Delta} + \underbrace{\alpha u - \alpha u_{nom}}_{\alpha\tilde{u}}, \tag{7c}$$

$$e^{(\nu)} = \Delta + \alpha\tilde{u}. \tag{7d}$$

The error-based ultra-local model (Δ) is the error signal between the two models. Although the ultra-local models can help to eliminate the effect of the nonlinearities, they cannot guarantee any tracking performance. Thus, additional control is needed $(\mathcal{K}(e,x))$, whose structure is not limited to a specific control type e.g. LQR, see [18].

$$u = -\frac{\Delta}{\alpha} - \mathcal{K}(e,x). \tag{8}$$

D. Linear Parameter Varying (LPV)

The Linear Parameter Varying model allows engineers to cope with nonlinear dynamics by applying scheduling parameters, see [19]:

$$\begin{aligned} \dot{x} &= A(\rho)x + B(\rho)u \\ y &= C^T(\rho)x + D(\rho)u \end{aligned} \tag{9}$$

The scheduling vector (ρ) determines the actual operating point of the system. Similarly to the robust control design technique, the performances of the controller can be defined, such as:

- *Tracking pefromance* As a frequent control goal, the tracking of a specific state can be formulated as:

$$z_1 = x_{1,ref} - y, \;\; |z_1| \to min, \tag{10}$$

- *Intervention* Another goal could be to minimize the energy consumption of the system:

$$z_2 = u, \qquad |z_2| \to min. \tag{11}$$

The presented performances can be guaranteed by appropriately chosen weighting functions. Including the performances, the following augmented state-space representation can be written:

$$\dot{x}_e = A_e(\rho)x_e + B_e(\rho)u_e + B_{e,w}(\rho)w_e, \tag{12a}$$

$$z_e = C_{e,1}(\rho)x_e + D_e(\rho)u_e. \tag{12b}$$

The design of an LPV controller leads to a quadratic optimization problem, which can be solved by selecting an adequate controller $(K(\rho))$. This controller must guarantee the quadratic stability of the closed-loop system. Moreover, the induced $\mathcal{L}_2$ norm from the disturbances to the performances must be smaller than a given value γ.

$$\inf_{K(\rho)} \sup_{\rho\in F_\rho} \sup_{\|w\|_2\neq 0, w\in\mathcal{L}_2} \frac{\|z\|_2}{\|w\|_2}, \tag{13}$$

E. DDPG

Deep Deterministic Policy Gradient (DDPG) is a type of reinforcement learning algorithm. It simultaneously learns a Q function and a policy based on the Bellman equation.
If the optimal action-value function $(Q^*(s,a))$ is known, the optimal action $(a^*(s))$ is computed as:

$$a^*(s) = arg\ \underset{a}{max}\ Q^*(s,a) \tag{14}$$

As mentioned, the DDPG has two sides: Q-learning, and Policy learning.

a) Q-learning: The Bellman equation provides the optimal action function:

$$Q^*(s,a) = \underset{s'\approx P}{E}\big[r(s,a) + \gamma \underset{a'}{max} Q^*(s',a')\big] \tag{15}$$

where $s' \approx P$ denotes the next state, s' is from the environment while γ is the learning rate.
With a neural network-based approximator $Q_\phi(s,a)$ with parameter ϕ and a set of transitions (s,a,r,s',d), the mean-squared Bellman error (MSBE) function can be computed

$$L(\phi,\mathcal{D}) = \tag{16}$$

$$\underset{(s,a,r,a',d)\approx\mathcal{D}}{E}\left[\Big(Q_\phi(s,a) - \big(r + \gamma(1-d)\underset{a'}{max} Q_\phi(s',a')\big)\Big)^2\right]$$

The target network is defined as:

$$r + \gamma(1-d)\underset{a'}{max} Q_\phi(s',a') \tag{17}$$

The Q-learning function tries to reach the target network by minimizing MSBE.
The MSBE is augmented with a target term $\mu\theta_{trag}$, giving the final MSBE loss function:

$$L(\phi,\mathcal{D}) = \tag{18}$$

$$\underset{(s,a,r,a',d)\approx\mathcal{D}}{E}\left[\Big(Q_\phi(s,a) - \big(r + \gamma(1-d)Q_{\phi_{targ}}(s',\mu\theta_{targ})\big)\Big)^2\right] \tag{19}$$

b) Policy Learning: The aim of this layer is to learn a deterministic policy $(mu_\theta(s))$ that maximizes $Q_\phi(s,a)$:

$$\underset{\theta}{max}\ \underset{s\approx\mathcal{D}}{E}[Q_\phi(s,\mu_\theta(s))] \tag{20}$$

A more detailed description can be found in [20].

TABLE I

TRAINING SCENARIOS

Type	Aggresivity	v_x	μ
DLC	H	C:5,10,...,30m/s	1
DLC	M	C:5,10,...,30m/s	0.8...,1
DLC	E	C:5,10,...,30m/s	0.5...,1
DLC	E	V:$v_{max} = 30 \downarrow$	1
DLC	M	V:$v_{max} = 30 \downarrow$	0.8,0.9
DLC	E	V:$v_{max} = 30 \downarrow$	0.5,...,0.7
DLC	E	V:$v_{min} = 10 \uparrow$	1
DLC	M	V:$v_{min} = 10 \uparrow$	0.8,0.9
DLC	E	V:$v_{min} = 10 \uparrow$	0.5,...,0.7
SINS	H	C:5,10,...,30m/s	1
SINS	M	C:5,10,...,30m/s	0.8,0.9
SINS	E	C:5,10,...,30m/s	0.5,...,0.7
CHIRPS	H	V:$v_{min} = 10 \uparrow$	1
CHIRPS	M	V:$v_{min} = 10 \uparrow$	0.8,0.9
CHIRPS	E	V:$v_{min} = 10 \uparrow$	0.5,...,0.7

III. TRAINING AND CONTROL DESIGN

In this section, the training of the control-aided neural network is presented. Firstly, the dynamical bicycle model is detailed, which describes the lateral motion of the vehicle, and is used in the control design. Then, training scenarios are described. Finally, the training process of the reinforcement learning-based agent is presented.

A. Lateral vehicle model

The lateral model, which is used in the control design, is based on the single-track bicycle model, see [21] It consists of two main equations: yaw-motion and lateral acceleration:

$$I_z\ddot{\psi} = \mathcal{F}_f(\alpha_f)l_f - \mathcal{F}_r(\alpha_r)l_r, \tag{21a}$$

$$ma_y = mv_x(\dot{\psi} + \dot{\beta}) = \mathcal{F}_f(\alpha_f) + \mathcal{F}_r(\alpha_r), \tag{21b}$$

where: a_y denotes the lateral acceleration, v_x is the longitudinal velocity, β is the side-slip, I_z denotes the yaw-inertia, m gives the mass of the vehicle, l_f, l_r are geometric parameters, $\mathcal{F}_f$,$\mathcal{F}_r$ are the lateral forces on the tires, $\alpha_f = \delta - \beta - \frac{\dot{\psi} l_f}{v_x}, \alpha_r = -\beta + \frac{\dot{\psi} l_r}{v_x}$ present the slip angles of the front and rear axles.
This model is the basis for the control design for ULM, LPV, and LQR techniques

B. Training scenarios

The goal of the training scenarios is to cover the whole dynamic range of the vehicle. Thus, it consists of several different tests listed in the table below. In the table, DLC means Double Lane Change, SINS: Sinuse-signal steering, and CHRIPS: Chirp-signal steering. E: Easy, M: Medium, H: Hard with regard to the lateral acceleration. C: constant, V: Varying longitudinal speed. μ is the adhesion coefficient, arrow $\uparrow$ means an increasing speed profile, while arrow $\downarrow$ presents a decreasing speed profile.

C. Inputs and structure of agent

One of the most crucial points of RL training is the selection of the appropriate input signals, which contain the information from the actual state and dynamic of the considered system. In case of the lateral dynamics, the measured and used signals are listed below:

- $v_y(t), v_y(t-1), \dot{\psi}(t), \dot{\psi}(t-1), v_x(t), v_x(t-1), \delta(t), \delta(t-1), a_x(t), a_x(t-1), a_y(t), a_y(t-1)$: are the states of the vehicle, two steps horizon helps to identify the local operational point of the system.
- e_y, ψ_e: are the error signals (lateral and yaw-angle), which are the inputs of the classical controllers.
- $\kappa(t+l), v_x(t+l)$: is the reference signals using a look-ahead distance-time (l), it allows the agent to select the appropriate control ahead.
- $\delta_{MPC}, \delta_{ULM}, \delta_{LQR}, \delta_{Stan}$: are the control inputs provided by the presented classical control structures.

a) Structure of the agent: The agent consists of a critic and an actor network, both have the same structure. They have one input layer with 21 inputs, 3 hidden layers, and one output layer with 4 outputs. The hidden layers consist of 48 neurons with ReLU activation functions. The agent has 4 outputs, which are a number between $p_i \in [0, 1]$ describing the predicted goodness of the four controllers on the look-ahead distance.

b) Reward function: It is a crucial point of the training of the agent. It must reflect on the stability and the performance of the vehicle as well:

$$R = -\dot{\delta}^2 a_1 - \delta^2 a_2 + T^2 a_3 - y_e^2 a_4 \quad \text{s.t } if\ |\psi_e| > \pi/2 \text{ then stop} \tag{22}$$

$a_1, ..., a_4$ are parameters weighting the intervention and its derivative (δ), the length of the simulation (T), and the tracking error (y_e). The simulation is also subject to a constraint: if the orientation error (ψ_e) is bigger than $\pi/2$ the simulation stops avoiding turning back of the vehicle.

D. Control signal selection

Finally, the applied control signal is selected based on the prediction values (p_i):

$$u = \mathbb{F}(max(p_i), u_{ULM,}, u_{stan}, u_{LPV}, u_{LQR}) \tag{23}$$

$\mathbb{F}$ represents the selection function. In this case, the control signal with the highest prediction value will be chosen.

IV. SIMULATION

In this section, the operation and the effectiveness of the proposed RL-based control algorithm are presented through 3 different scenarios. The control algorithm is implemented in a Simulink environment with a connection to the high-fidelity vehicle dynamics simulation software, CarMaker. In the simulation, a D-class passenger car is used. In the first test, the vehicle is driven along the Hungarian Formula 1 track at low speed. The second simulation includes the same track with a high-speed profile. In the last scenario, the vehicle is driven on a low μ surface.

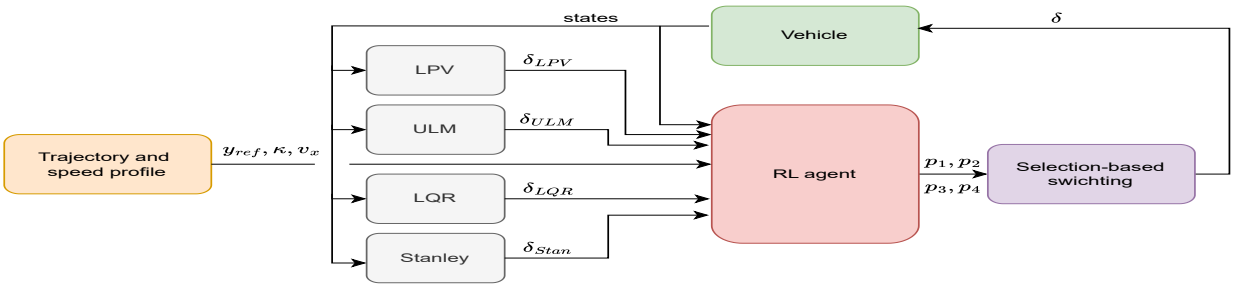
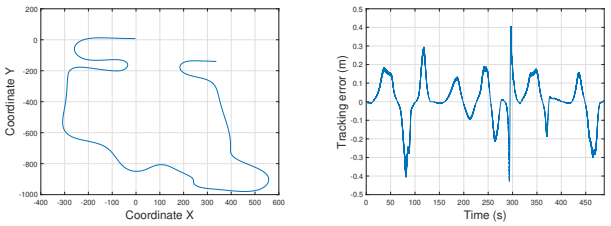


Fig. 1. Structure of the proposed algorithm

A. First test scenario

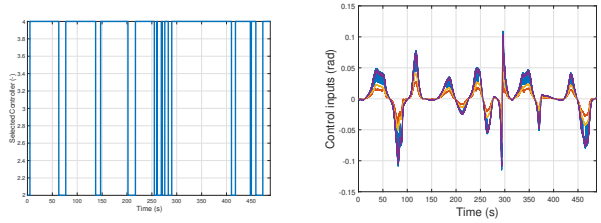
In the first scenario, the proposed algorithm is tested at low longitudinal velocity. Figure 2(a) presents the test track, while Fig 3 (b) the lateral error during the simulation. As the figure shows, the peak value of the lateral error is below $< 0.4m$, which means the vehicle is able to follow the test track with high accuracy. Fig 3 presents the selected



(a) Reference trajectory (b) Lateral error

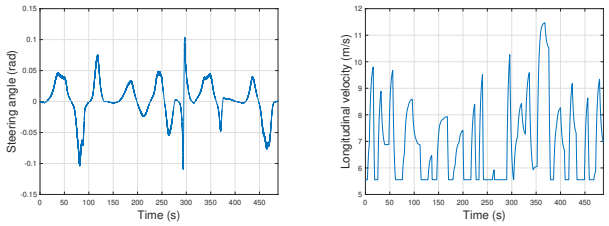
Fig. 2. Reference trajectory and the computed lateral error during the simulation

signal by the neural network and all control signals. In the selection "1" is the ULM, "2" is the Stanley controller, "3" is LQR, and "4" denotes the LPV controller. As the figure shows, in this test case, the neural network switches between the Stanley and the LPV controller. The switching is linked to the longitudinal velocity, shown in Fig 4 (b). When the velocity is low (around $5m/s$) and there is a bend within the lookahead distance, the RL chooses the Stanley controller and, in other cases, the LPV-based steering angle. In Fig 3(b) the blue line represents the ULM, the yellow is the Stanley, orange means the LQR, and purple depicts the LPV control signals. The applied steering angle is shown in Fig 4(a).



(a) Selected signal (b) Controllers

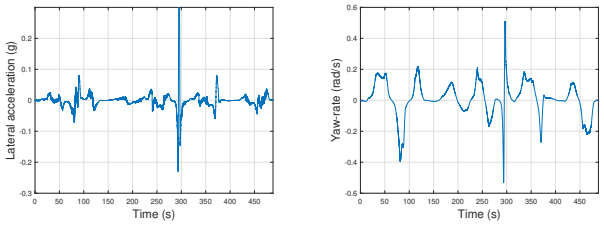
Fig. 3. Selected control signal and all controllers



(a) Steering angle (b) Longitudinal velocity

Fig. 4. Steering angle and velocity profile

The Fig 5 presents the lateral acceleration and the yaw-rate signal. Around $t = 300s$ there is a peak value in the lateral acceleration and the yaw rate. It is caused by a sharp bend and a relatively high longitudinal velocity.



(a) Lateral acceleration (b) Yaw-rate

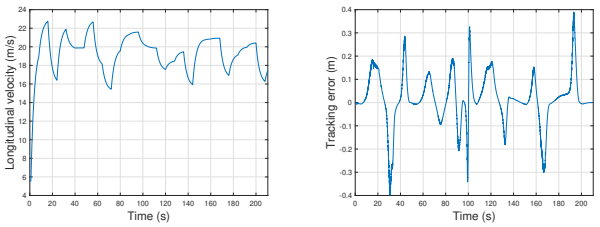
Fig. 5. Lateral acceleration and yaw-rate

B. Second test scenario

In the second test case, the vehicle is driven with the same conditions except for the longitudinal velocity profile. Fig 6 (a) presents the speed profile for this case. It varies between $16-23m/s$. The lateral error (Fig 6(b)) remains in the same range as in the previous case. Fig 7 demonstrates the control selection and the steering angle. In this case, the RL switches between the first (ULM) and the fourth (LPV) controllers. It is reasonable since the Stanley controller works well only at low longitudinal velocity, while the LQR is at a specific v_x.

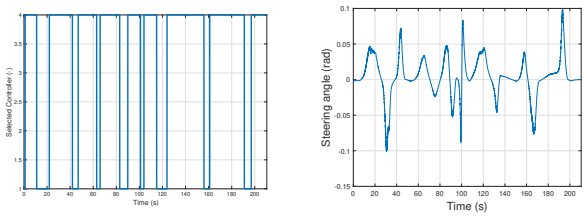
C. Low μ test scenario

In the last scenario, the proposed algorithm is tested under extreme circumstances, meaning surface with $\mu = 0.6$. The longitudinal velocity and the lateral error are shown in Fig 8.



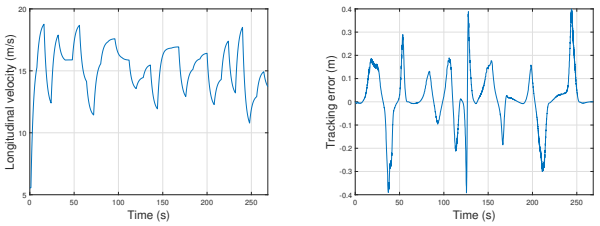
(a) Longitudinal velocity (b) Lateral error

Fig. 6. High-speed profile and the lateral error



(a) Selected signal (b) Steering angle

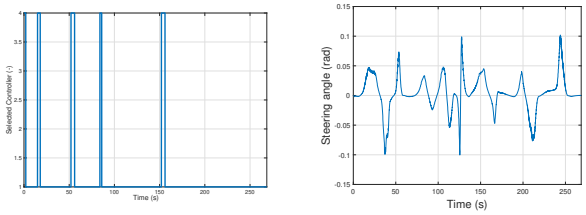
Fig. 7. Selected control signal and all controllers at high speed



(a) Longitudinal velocity (b) Lateral error

Fig. 8. Mid-ranged speed profile and the lateral error at low μ

The lateral error is still in an acceptable range. Finally, Fig 9 presents the selected controller and the steering angle. In this case, the RL agent switches between the LPV and ULM controllers. ULM is used in more than 80% of the simulation. The LPV controller has no information on the changed μ surface, however, the ULM can identify the change in the dynamics.



(a) Selected signal (b) Steering angle

Fig. 9. Selected control signal and all controllers at low μ

V. Conclusion

In the paper, a reinforcement learning-based switching control has been proposed. Four different controller techniques have been involved: LPV, ULM, LQR, and Stanley. The control algorithm has been tested through three different test scenarios: low and high-velocity profiles and with low adhesion coefficients. The tests have shown that the proposed agent is able to detect the changes and select the best control algorithm for the given circumstances. In this way, the stability of the controller can be guaranteed while maintaining a high performance level.

References

[1] A. Visioli, *Practical PID Control*. Springer, 2006.

[2] A. Sinha, *Linear Systems: Optimal and Robust Control*. CRC Press, 2007.

[3] D. Liberzon, *Switching in Systems and Control*. Springer, 2003.

[4] W. Zeng, Q. Jiang, Y. Liu, J. Xie, and T. Yu, "A multi-level fuzzy switching control method based on fuzzy multi-model and its application for pwr core power control," *Progress in Nuclear Energy*, vol. 138, p. 103743, 2021.

[5] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: A survey of recent results," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 308–322, 2009.

[6] E. Moulay, R. Bourdais, and W. Perruquetti, "Stabilization of nonlinear switched systems using control lyapunov functions," *Nonlinear Analysis: Hybrid Systems*, vol. 1, no. 4, pp. 482–490, 2007, proceedings of the International Conference on Hybrid Systems and Applications, Lafayette, LA, USA, May 2006: Part I.

[7] S. Cerf and Éric Rutten, "Combining neural networks and control: potentialities, patterns and perspectives," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 9036–9049, 2023, 22nd IFAC World Congress.

[8] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, p. 2397?2404, Apr. 2023.

[9] A. Lelkó and B. Németh, "Optimal motion design for autonomous vehicles with learning aided robust control," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 9, pp. 12 638–12 651, 2024.

[10] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.

[11] A. AbdElmoniem, A. Osama, M. Abdelaziz, and S. A. Maged, "A path-tracking algorithm using predictive stanley lateral controller," *International Journal of Advanced Robotic Systems*, vol. 17, no. 6, p. 1729881420974852, 2020.

[12] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *2007 American Control Conference*, 2007, pp. 2296–2301.

[13] C. Piao, X. Liu, and C. Lu, "Lateral control using parameter self-tuning lqr on autonomous vehicle," in *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, 2019, pp. 913–917.

[14] D. Fenyes, B. Nemeth, and P. Gaspar, "Lpv-based autonomous vehicle control using the results of big data analysis on lateral dynamics," in *2020 American Control Conference (ACC)*, 2020, pp. 2250–2255.

[15] M. Fliess and C. Join, "Stability margins and model-free control: A first look," *Proc. 13th European Control Conference*, pp. 454–459, 2014.

[16] M. Fliess and H. Sira-Ramirez, "An algebraic framework for linear identification," *ESAIM: Control, Optimisation and Calculus of Variation*, vol. 9, pp. 151–168, 2003.

[17] T. Hegedűs, B. Németh, and P. Gáspár, "Identification of tire characteristics using physics-informed neural network for road vehicles," in *2024 32nd Mediterranean Conference on Control and Automation (MED)*, 2024, pp. 706–711.

[18] T. Hegedus, D. Fenyes, Z. Szabo, B. Nemeth, L. Lukacs, R. CSikja, and P. Gaspar, "Implementation and design of ultra-local modelbased control strategy for autonomous vehicles," *Vehicle System Dynamics*, pp. 1–25, 2023.

[19] R. TĂłth, *Modeling and Identification of Linear Parameter-Varying Systems*, ser. Lecture Notes in Control and Information Sciences. Berlin, Heidelberg: Springer, 2010, vol. 403.

[20] "Deep deterministic policy gradient, howpublished = https://spinningup.openai.com/en/latest/algorithms/ddpg.html, note = Accessed: 2015-04-01."

[21] J. Hahn, R. Rajamani, S. You, and K. Lee, "Real-time identification of road-bank angle using differential GPS," *IEEE Transactions on Control Systems Technology*, vol. 12, pp. 589–599, 2004.