PID vs. Reinforcement Learning: A Comparative Study on Autonomous Driving in the Gymnasium Car Racing Environment

Ali Roshandelzade[1*] - Behrooz Rezaie[2]
[1] MSc student at Babol Noshirvani University of Technology, Shariati Av., Babol, Mazandaran, Iran
[2] Department of Electrical and Computer Engineering, Babol Noshirvani University of Technology, Shariati Av., Babol, Mazandaran, Iran

ABSTRACT

In this paper, we investigate two distinct control strategies for autonomous vehicles navigating tracks: Proportional-Integral-Derivative (PID) control and Proximal Policy Optimization (PPO). We compare their feasibility and computational efficiency and introducing a novel approach for longitudinal and lateral control within the CarRacing environment of OpenAI's Gymnasium. While deep reinforcement learning methods, such as PPO, have demonstrated significant potential in the control domain, they often require substantial computational resources and time due to the inherent exploration-exploitation trade-off. Our findings suggest that, in certain scenarios, classical control techniques like PID offer greater reliability and ease of implementation.
Keywords: *reinforcement learning, control systems, pid, ppo, proximal policy optimization*

1. *INTRODUCTION*

In recent years, there has been a significant increase in attention to autonomous vehicles. The growing need for energy conservation, road safety, and ease of transportation has led to increased scientific discussions and research on autonomous vehicles. One current method of vehicle control is path tracking, where an acceptable path is determined, and the autonomous vehicle is controlled to follow that path. Raffone et al. [1]designed a controller using a camera to detect lanes and maintain the vehicle's path. Kim et al. [2] presented a control strategy for high-speed autonomous vehicles as an active safety measure, using inverse vehicle dynamics for longitudinal control and a linear quadratic regulator (LQR) for lateral control. Goldner et al. [3] applied sliding mode control for lateral vehicle control on highways. Similarly, Tagn et al. [4] utilized a higher-order sliding mode controller for lateral vehicle control, applying the super twisting algorithm to minimize vehicle displacement. Sliding mode control allows for robust performance against noise and disturbances. A major challenge in autonomous vehicle control is managing disturbances and nonlinearities. Korno et al. [5]developed a linear controller with variable parameters based on robust control (H∞) to control the vehicle's angle amid nonlinearities. The controller architecture, by programming a control weight function, addresses steering nonlinearities. Controlling the vehicle's lateral position and path planning is another critical challenge. Norouzi et al. [6] used a fifth-order polynomial to define a path for lane changing and employed a sliding mode controller for vehicle position control and lane changing. Among the latest and most widely used control methods for autonomous vehicles are machine learning and deep learning techniques, which have seen significant growth due to advancements in hardware processing speed and machine learning algorithms. A popular approach in autonomous vehicle control is reinforcement learning, where trial and error teaches the agent to perform specific actions, such as following a designated path. Recent studies on autonomous vehicle control, path tracking, and vehicle angle planning using reinforcement learning demonstrate substantial progress [7], [8], [9]. One advantage of reinforcement learning is the elimination of the need for modeling and error calculation. However, employing reinforcement learning with deep neural networks, while highly effective, demands powerful processing

hardware and considerable time for environmental learning. This research aims to examine the differences in controlling autonomous vehicles for path tracking using a simple controller (PID) alongside deep learning in a simplified simulation environment.

2. OVERVIEW OF CONTROL METHODS

In this section, we outline the control strategies implemented in this study, focusing on their theoretical basis and practical application.

2.1 PID Control

The Proportional-Integral-Derivative (PID) controller is a well-established and widely adopted technique for regulating diverse dynamic systems. This method calculates an error signal by subtracting the measured system output from the desired setpoint. The control input is then determined as a linear combination of three terms: the proportional term, which addresses the current error; the integral term, which corrects accumulated past errors to eliminate steady-state discrepancies; and the derivative term, which anticipates future errors by evaluating the error's rate of change. Each term is weighted by a specific gain, allowing the PID controller to be tuned for optimal performance. Its enduring use is attributed to its straightforward implementation and robust effectiveness across a broad range of applications.

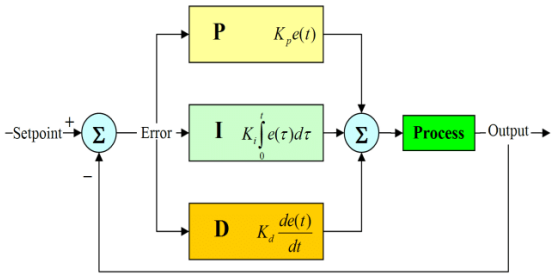

Fig. 1. *PID Control loop diagram*

The overall PID control function is:

$$u(t) = k_p e(t) + k_i \int_0^t e(\mathrm{T})d\mathrm{T} + k_d \frac{de(t)}{dt} \qquad (1)$$

Where k_p,k_i and k_d are all non-negative parameters denote the coefficents of proportional, integral and derivative terms respectively. E(t) is the error signal calculated by subtracting the output from the input.

2.2. Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make sequential decisions by interacting with an environment to maximize a cumulative reward. Unlike supervised learning, which relies on labeled data, or unsupervised learning, which identifies patterns in unlabeled data, RL operates through trial and error. The agent observes the environment's state, selects an action based on a policy, and receives feedback in the form of a reward and a new state. Over time, the agent refines its policy to favor actions that yield higher long-term rewards. This process is formalized using a Markov Decision Process (MDP), comprising states, actions, transition probabilities, rewards, and a discount factor to balance immediate and future rewards. RL algorithms, such as Q-learning, Deep Q-Networks (DQN), and Proximal Policy Optimization (PPO), differ in how they update the policy. For instance, value-based methods estimate the expected reward for state-action pairs, while policy-based methods directly optimize

the policy. RL is particularly suited for control tasks, such as autonomous vehicle navigation, where explicit instructions are unavailable, and the agent must adapt to dynamic conditions. However, RL often requires significant computational resources and careful tuning due to challenges like exploration-exploitation trade-offs and reward function design.
Basic reinforcement learning is modeled as a markov decision process:

$$P_a(S, \acute{S}) = \Pr(S_{t+1} = \acute{S} | S_t = s, A_t = a) \quad (2)$$

Which S is set of environment and agent states (State Space), A is set of action (Action Space), S' is the next state and the function will calculate the probability of next state based on the current action.
$R_a(S,S')$ is the immediate reward after transition from S to S' under action a. The goal of a reinforcement learning agent is to learn a policy:

$$\pi: S \times A \to [0,1], \pi(S, a) = pr(A_t = a | S_t = S) \quad (3)$$

Value function is a function that estimates how good is to be in current state:

$$V_\pi(S) = E[\sum_{t=0}^{\infty} \Upsilon^t R_{t+1} | S_0 = S] \quad (4)$$

Which $0<\gamma<1$ is discount factor.
proximal policy optimization is a new state of the art reinforcement learning method which was introduces by schulman et al[10] in 2017 and since then it is default RL algorithm in OpenAI . PPO has since become a widely adopted standard in continuous control tasks due to its balance between sample efficiency and implementation simplicity.
The PPO algorithm is based on optimizing a surrogate objective function while ensuring that policy updates do not deviate excessively from the current policy, which helps maintain stable learning. It modifies the objective function of traditional policy gradient methods by clipping the probability ratio between the new and old policies. The clipped surrogate objective is defined as follows:

$$L^{CLIP}(\theta) = E_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_T] \quad (5)$$

Where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$ is the probability ratio, $\hat{A}_T$is the estimated advantage at time step t,ε is a hyperparameter controlling the clip range, π_θ and $\pi_{\theta old}$ are new and old policy networks respectively.
By clipping the objective function, PPO discourages large policy updates, which improves training stability and performance over time.

here is a proximal policy optimization algorithm that uses fixed length trajectory segment:

Algorithm PPO, Actor-Critic Style
For iteration=1,2, ... do
For actor=1,2,....N do
Run Policy $\pi_{\theta old}$ in environment for T timesteps
Compute advantage estimates $\hat{A}_1, \ldots \ldots, \hat{A}_T$
End for
Optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT
$\theta_{old} \leftarrow \theta$
End for

each iteration, each of N actors collect T timesteps of data. Then we construct the surrogate loss on these NT timesteps of data and optimize it with minibatch SGD or any other optimization algorithm for K epochs.[10]

schulman et al.[10] *also did a comparison between the PPO algorithm and other reinforcement algorithms in varous OpenAI's gymnasium environments and in almost all of them PPO have done a better job than others.*

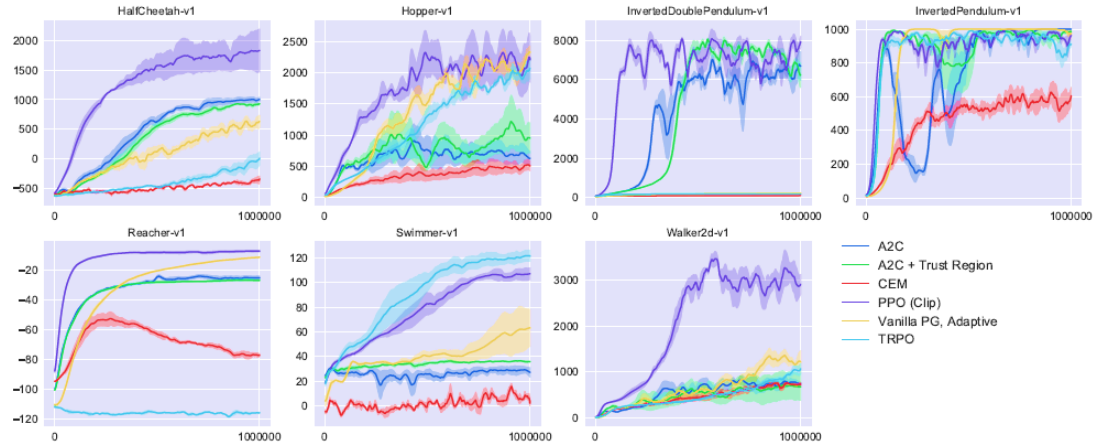

Fig. 2. *comparison between PPO and some other RL algorithms*

2.3 Car Racing Environment

For simulation and implementing the ideas we used an specific OpenAI's gymnasium environment called Car Racing, gymnasium is an application programming interface standard for reinforcement learning with a diverse collection of reference environments [11]. *Each environment in gymnasium created with a space type which important types are Box, Discrete, Multibinary and Multidiscrete. This types are shape of action space and observation space for agent for making decision and observing the environment. Car Racing environment uses Box2D action space which consist of steering, gas and braking action with both discrete and continuous actions. This environment has a top-down view from racetrack and its observation space contains a 96×96 pixel image from the current state of the environment.*

3. IMPLEMENTATION

we compared the car behavior in the environment with both PID controlled and Reinforcement learning controlled agent, for implementing reinforcement learning we use a proximal policy optimization method combined with a neural network.
In our implementation, the agent interacts with the CarRacing-v3 environment through trial and error. The reward structure is as follows:
A positive reward of +1000/N is given each time the agent successfully drives over a new track tile, where N is the total number of tiles visited during the episode. A negative reward of −0.1 is applied at every frame to encourage faster and more efficient navigation. A deep neural network with six hidden layers serves as the policy model and another one with 5 hidden layers implemented for value function, model is trained using PPO to maximize cumulative rewards. Through continuous interaction with the environment, the agent learns optimal driving strategies that balance speed, accuracy, and control.
We implemented the PPO algorithm for car racing environment in 4 different ways, 20,000, 50,000, 100,000 and 150,000 timesteps, the learning rate for every model is 0.0003 as default and batch size of 64, the discount factor or γ is 0.99 for making the agent more farsighted and promotes strategies that yield higher long term rewards and set the clip range equals to 0.2 as its default.

we implement a dual-PID control framework to manage the steering and speed of an autonomous vehicle in the CarRacing-v3 simulation environment. The controller leverages image processing techniques to extract key features from the environment. Specifically, optical flow is employed to estimate the vehicle's speed

based on successive image frames, while a combination of color filtering, edge detection, and region-of-interest extraction is used to determine the lateral error relative to the center of the road.
Two PID controllers are deployed:
Steering Controller: Computes the steering angle using the error derived from the processed image slice. The controller's output is a weighted sum of the proportional, integral, and derivative terms, which corrects the vehicle's trajectory to align with the road's center.
Speed Controller: Adjusts the throttle and brake inputs by comparing the desired speed (which is a function of the steering angle) against the speed estimated by optical flow. Similar PID control principles are applied to modulate the gas and brake commands.
This integrated approach not only provides real-time control adjustments based on visual feedback but also facilitates data collection for subsequent analysis by recording simulation frames for offline review.

4.RESULTS

For testing our ideas we used google colab service, google Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs.[12] We used a Nvidia tesla T4 GPU for training our models and table 1 shows training time for each of our models.

Table 1: *comparison of training time consumed for training models*

Model	*Number of Epochs*	*Training Time(s)*
PPO 20K	*20,000*	*255.95*
PPO 50k	*50,000*	*637.90*
PPO 100k	*100,000*	*1232.16*
PPO 150K	*150,000*	*1878.98*

One of the methods for comparing the accuracy of prediction in PPO models is kullback-leibler divergence. Kullback-leibler divergence is a type of statistical distance and it measures how much model probability distribution (Q) differs from real probability distribution (P).

$$D_{KL}(P||Q) = \sum_{x \epsilon X} P_x \log\left(\frac{P(x)}{Q(x)}\right) \quad (6)$$

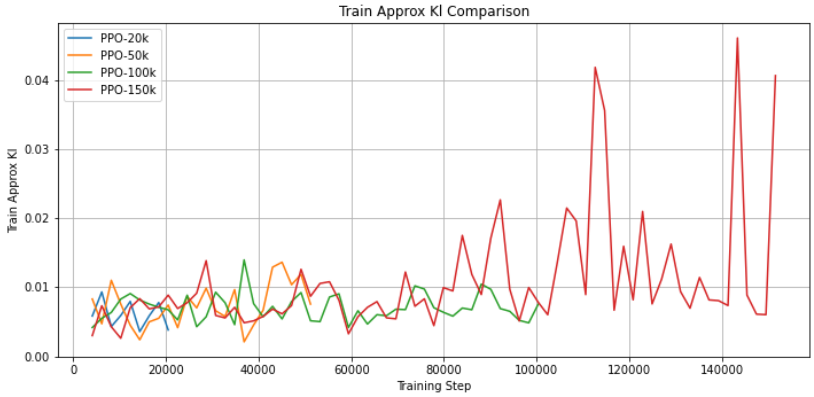

Fig. 3. *train approximate kullback-leibler per training step for models.*

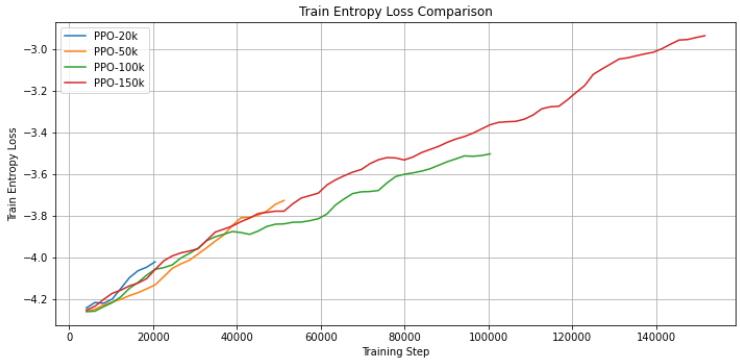

Fig. 4. *comparison between train entropy loss for models.*

Entropy loss, which measures the randomness of the policy's actions. Higher entropy means higher exploration, lower entropy means higher exploitation, as in PPO entropy used to encourage exploration by adding an entropy bonus to the objective function, the increasing in 150k model indicates that model reduces exploring due to its better actions.

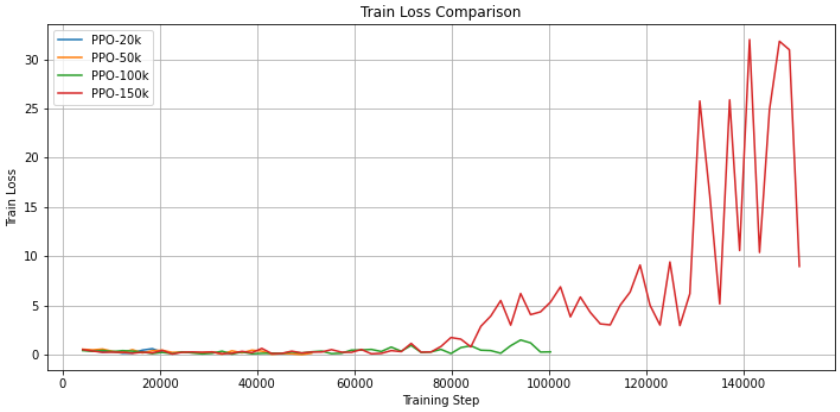

Fig. 5. *Overall train loss comparison between models.*

Figure 5 shows overall train loss for different models, in PPO train loss combines policy loss, value loss and entropy loss. PPO-150K's large spikes suggest significant updates to the policy or value function, possibly due to encountering new states or larger policy shifts (as seen in the KL divergence graph).

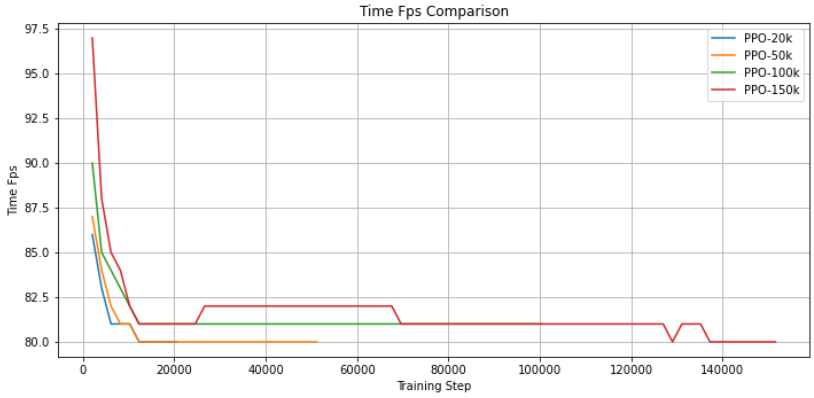

Fig. 6. *comparison of time fps in models.*

Frames Per Second (FPS) over training steps, indicating the computational efficiency of training. Training speed is fairly consistent across timesteps after the initial drop, with minor variations. PPO-150K's dip suggests it might be handling more complex updates due to longer training.

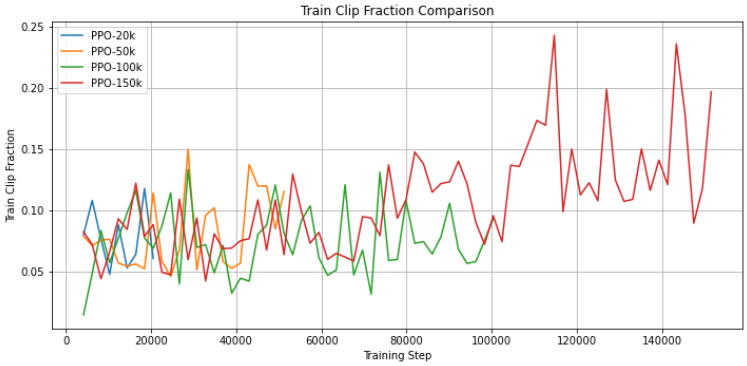

Fig. 7. *train clip fraction comparison.*
The fraction of policy updates that are clipped in PPO, a mechanism to prevent large policy changes, a higher clip fraction in PPO-150k means more updates are being clipped, indicating larger attempted policy changes that are being considered, this aligns with its higher KL divergence and loss spikes, showing it's pushing the boundaries of policy updates more aggressively.

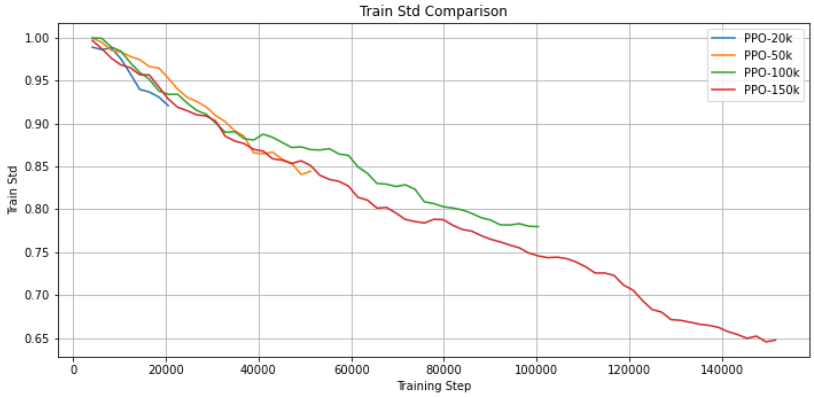

Fig. 8. *standard deviation for training comparison between models.*
Standard deviation of the policy's action distribution, another measure of exploration which higher std stands for higher exploration. It clear that more timesteps maintain more exploration.

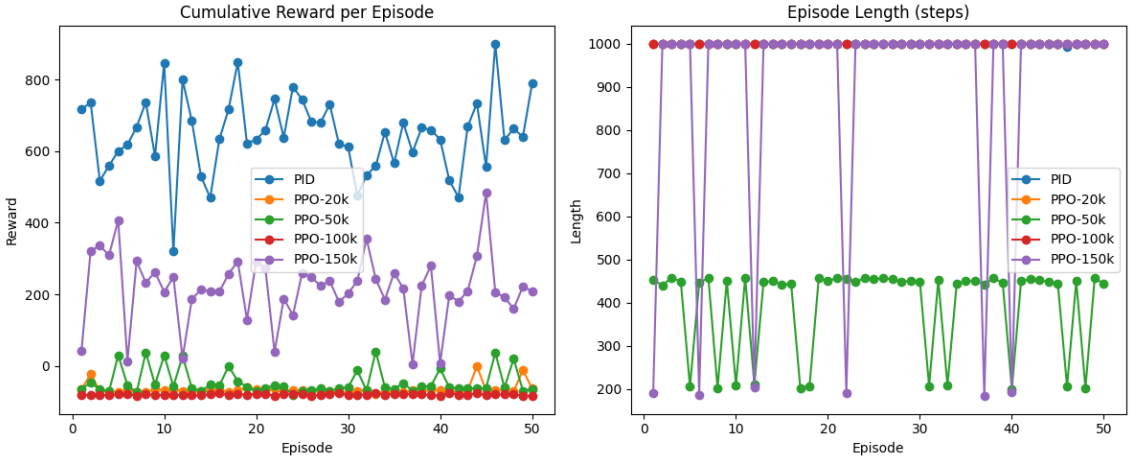

Fig. 9. *comparison between PID control and RL trained models with different timesteps.*

We implementing the PID controller and RL models for controlling the car in car racing environment for 1000 frames each, Fig 9 shows cumulative reward per episode and episode length for every try, we have done the experiment for 50 times with each of models and with PID approach, as showed in previous results the PPO-150k model is the best model trained and best performance between RL models, but in almost every try PID controlled car done a better job and maintain a better result. For training the ppo-150k model we spent 31 minutes of training which is not much but for a 2d simple environment its sufficent, more complex environments need much more computational power and exploration time for agent, and for better results bigger neural networks implementation needed, PID control doesn't need any training and its more simple to implement, and it shows that for simple tasks with not much complexity classic control approaches can still do a significant job with less expense.

5.CONCLUSION

we compared PID control and model based Reinforcement learning control for the car racing environment of OpenAI's gymnasium. This research shows that deep reinforcement learning, although has shown promising results in control field but also for learning properly needs a significant amount of time and computational resource, in other hand PID control can be easily applied to many control problems and it doesn't need much processing resource, so for doing simpler tasks we can use it with reassurance for good outcome. And for more complicated problems hybrid control of PID or any classic methods combined with reinforcement learning can be effective.

REFERENCES

[1] E. Raffone, C. Rei, and M. Rossi, "Optimal look-ahead vehicle lane centering control design and application for mid-high speed and curved roads," in *2019 18th European Control Conference (ECC)*, 2019, pp. 2024–2029. doi: 10.23919/ECC.2019.8796031.
[2] D. Kim, J. Kang, and K. Yi, "Control strategy for high-speed autonomous driving in structured road," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 186–191. doi: 10.1109/ITSC.2011.6082856.
[3] J. Guldner, V. I. Utkin, and J. Ackermann, "A sliding mode control approach to automatic car steering," in *Proceedings of 1994 American Control Conference - ACC '94*, 1994, pp. 1969–1973 vol.2. doi: 10.1109/ACC.1994.752420.
[4] G. Tagne, R. Talj, and A. Charara, "Higher-order sliding mode control for lateral dynamics of autonomous vehicles, with experimental validation," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 678–683. doi: 10.1109/IVS.2013.6629545.
[5] M. Corno, G. Panzani, F. Roselli, M. Giorelli, D. Azzolini, and S. M. Savaresi, "An LPV Approach to Autonomous Vehicle Path Tracking in the Presence of Steering Actuation Nonlinearities," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 4, pp. 1766–1774, 2021, doi: 10.1109/TCST.2020.3006123.
[6] A. Norouzi, M. Masoumi, A. Barari, and S. Farrokhpour Sani, "Lateral control of an autonomous vehicle using integrated backstepping and sliding mode controller," *Proc. Inst. Mech. Eng. Part K J. Multi-Body Dyn.*, vol. 233, no. 1, pp. 141–151, 2019.
[7] E. Meyer, H. Robinson, A. Rasheed, and O. San, "Taming an Autonomous Surface Vehicle for Path Following and Collision Avoidance Using Deep Reinforcement Learning," *IEEE Access*, vol. 8, pp. 41466–41481, 2020, doi: 10.1109/ACCESS.2020.2976586.
[8] K. B. Naveed, Z. Qiao, and J. M. Dolan, "Trajectory Planning for Autonomous Vehicles Using Hierarchical Reinforcement Learning." 2020. [Online]. Available: https://arxiv.org/abs/2011.04752
[9] J. Ma, H. Xie, K. Song, and H. Liu, "Self-optimizing path tracking controller for intelligent vehicles based on reinforcement learning," *Symmetry*, vol. 14, no. 1, p. 31, 2021.
[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms." 2017. [Online]. Available: https://arxiv.org/abs/1707.06347
[11] "Gymnasium Documentation." [Online]. Available: https://gymnasium.farama.org/
[12] "Google Colaboratory." [Online]. Available: colab.google