A comparative analysis of machine learning and PID controllers for implementing adaptive cruise control in autonomous vehicles

A Comparative Analysis of Machine Learning and PID Controllers for Implementing Adaptive Cruise Control in Autonomous Vehicles

Federico Bravetti[1], Luigi Borriello[1], Alex Testa[1], Manuel Andruccioli[1], Kelvin Olaiya[1], and Roberto Girau[1]

[1]Dept. of Computer Science and Engineering, University of Bologna, Italy
Email: {federico.bravetti2, luigi.borriello2}@studio.unibo.it
{alex.testa, manuel.andruccioli, kelvin.olaiya, roberto.girau}@unibo.it

***Abstract*—Adaptive Cruise Control (ACC) is a widely adopted Advanced Driver Assistance System (ADAS) technology that enhances driving comfort and safety. As part of the broader effort to advance autonomous driving, this paper investigates the implementation of ACC systems using two approaches: a standard PID controller, valued for its simplicity, reliability, and widespread use, and a machine learning-based model, designed to offer similarly straightforward usability. The experiments were carried out within the CARLA simulator, a platform that provides a controlled environment for testing and evaluation. By comparing the performance of these two methods, the study aims to evaluate whether machine learning can achieve improved results while retaining the ease of implementation associated with traditional PID controllers. This comparative study highlights the strengths and limitations of both approaches, offering valuable insights into how machine learning can enhance ACC functionality and contribute to safer autonomous driving systems.**

***Index Terms*—Adaptive Cruise Control, ADAS, PID, Reinforcement Learning, Autonomous Driving, CARLA simulator.**

I. Introduction

Advanced Driver Assistance Systems (ADAS) are transforming the automotive industry by introducing innovative solutions that enhance driver safety, comfort, and overall vehicle efficiency [1]. Among these, Adaptive Cruise Control (ACC) stands out as one of the most widely adopted and mature technologies [2]. ACC systems maintain a set speed and regulate the distance from the vehicle ahead, reducing the driver's cognitive load and contributing to accident prevention. These systems are considered a cornerstone in the progression toward autonomous driving, where control precision, adaptability, and real-time decision-making are essential.

The design of an effective ACC system involves a careful balance between simplicity, reliability, and adaptability to complex driving scenarios such as varying traffic densities, sudden braking, or erratic behavior of leading vehicles. Traditionally, Proportional-Integral-Derivative (PID) [3] controllers have been the method of choice due to their straightforward implementation and reliable performance under stable conditions. However, PID controllers often struggle to respond effectively to nonlinear or rapidly changing dynamics, limiting their effectiveness in more complex scenarios. In contrast, machine learning (ML)-based control strategies offer a data-driven alternative that can capture complex relationships between input variables and control actions [4]. These models can learn from both historical real-world and simulated driving data, enabling them to generalize across diverse situations and improving performance in terms of responsiveness and adaptability.

In this paper, we present a comparative analysis of ACC systems implemented using two distinct approaches: a standard PID controller and a machine learning-based model using reinforcement learning (RL). Both implementations are developed and evaluated within the CARLA simulator[1], an open-source platform for autonomous driving research. Our research hypothesis posits that a machine learning approach, while maintaining a level of simplicity similar to that of the PID controller, can achieve superior performance in complex or dynamic scenarios, making ACC systems more effective and robust. The objective is to assess the advantages and limitations of each approach under different driving scenarios, including both steady-state cruising and abrupt changes in the behavior of leading vehicles.

II. Related Works

Adaptive Cruise Control (ACC) systems have been extensively studied and implemented using both traditional control methods and advanced machine learning techniques. Early solutions often relied on PID controllers due to their simplicity and ease of integration. However, modern advancements have shifted towards incorporating more sophisticated approaches such as Model Predictive Control (MPC) and Adaptive PID, which offer better adaptability to changing conditions. Recently, machine learning techniques, including Deep Reinforcement Learning (DRL), have shown promise in improving ACC performance by learning optimal driving policies through interaction with the environment. Furthermore, hybrid approaches are being introduced, combining classical control methods and machine learning to leverage the advantages of both techniques. These hybrid models aim to optimize

[1]https://carla.org

performance and adaptability while maintaining stability and effectiveness across different driving conditions.

A. Traditional Control Systems

The PID controller is one of the most widely used controllers in industry due to its simplicity and ease of integration, made possible by automatic tuning techniques that eliminate the need for a detailed system model analysis. When using a basic PID controller, it is possible to develop an ACC system with various features, such as cruise control, following function, speed selection, stop-and-go, and an emergency braking system. Canale and Malan [5] presented a robust design approach for ACC systems with stop-and-go feature using simple P and PI controllers. The method incorporates safety and comfort constraints through a tailored reference acceleration profile generator that mimics driver behaviour. The control system relies on standard vehicle documentation and data fields to model dynamics via interval plant families. This modelling enables the design of low-complexity, but robust controllers capable of handling nonlinear behaviours and ensuring smooth driving experiences.

However, a key limitation of PID is its difficulty in adapting to different setpoints. When the setpoint changes, especially if the change is rapid or significant, the PID controller may not respond optimally. A more sophisticated controller such as MPC [6] can optimize control actions based on the system model, taking into account predicted changes, which allows better adaptation to setpoint variations and system conditions, ensuring a smoother and more accurate response. In [7] Naus et al. present the implementation and evaluation of an ACC with a stop-and-go feature using an MPC approach. The study distinguishes between comfort metrics – such as acceleration and jerk – and traffic-required behavior – such as relative distance, Time-To-Collision (TTC) – implementing a hybrid controller that optimizes both aspects and solving it as multi-parametric quadratic programming.

Another possible solution is the use of Adaptive PID [8]. Adaptive PID is particularly useful, as it adjusts its parameters in real-time based on changing system dynamics, which helps to improve performance under varying conditions, such as disturbances or changes in system behavior over time. Unlike traditional PID, which uses fixed gains, an adaptive PID ensures that the control system remains stable and effective even when facing setpoint changes or environmental factors. In [9] Simorgh et al. present an adaptive PID control design for longitudinal velocity control of autonomous vehicles, leveraging an adaptive control model to address nonlinearities, time-varying conditions, and disturbances such as aerodynamic forces and varying road conditions. The proposed controller dynamically adjusts throttle and brake policies to ensure accurate velocity tracking, incorporating robust update laws to prevent parameter drift and maintain stability.

Moreover, modern approaches are increasingly integrating standard control models with machine learning to enhance their performance. Solutions leveraging these techniques have been proposed for both PID [10] and MPC [11].

B. ML Control Systems

Recent advancements have seen the integration of machine learning (ML) into ACC systems. In [12] Mosharafian et al. present a Gaussian Process(GP)-based stochastic model for Cooperative ACC (CACC) systems, aiming to enhance highway safety and capacity while reducing reliance on frequent communication. By leveraging GP regression, vehicles predict the speed trajectories of preceding vehicles using model-based communication, which broadcasts hyper-parameters of local GP models instead of raw data. This approach allows vehicles to maintain situational awareness even during communication failures, preventing the system from degrading performance.

Moreover, a significant contribution is the application of Deep Reinforcement Learning (DRL) in ACC systems. DRL enables the system to learn optimal driving policies through interaction with the environment, improving decision-making in complex driving scenarios. More advanced techniques, like Deep Q-Learning (DQL) and the use of camera images for training instead of traditional sensor-based detection, have shown promising results. In [13] Das and Won propose a dynamic ACC system for highways with ramps, leveraging deep reinforcement learning to optimize headway distance in real-time based on traffic conditions. The model employs a Markov Decision Process and a deep Q-network to adaptively adjust distances by considering factors such as traffic density, vehicle speed, and ramp length. The system dynamically respond to traffic and other perturbations highlights.

Other approaches have used GYM environments [14], with techniques like Proximal Policy Optimization (PPO) [15], which has become popular due to its ability to handle continuous action spaces and its stability during training. Further promising results have been shown using the Deep Deterministic Policy Gradient (DDPG) algorithm, which offers improvements over traditional methods in managing the complexity of multi-objective driving scenarios. In [16] Yan et al. present a car-following strategy that combines DDPG and CACC to enhance autonomous driving performance. The strategy leverages the reliability of CACC for basic car-following tasks and the exploratory strength of DDPG for complex scenarios, achieving a balance between robustness and flexibility in autonomous vehicle control.

Machine learning techniques face significant challenges related to user data privacy, as traditional methods often require centralized data collection, which increases the risk of data breaches and misuse. To address this issue, Federated Learning (FL) has been proposed as a solution [17]. This approach mitigates privacy concerns by enabling distributed learning, where individual devices collaboratively train models without sharing raw data. In doing so, FL enhances ACC performance by improving scalability and personalization while safeguarding user data privacy.

III. Implementation and evaluation of PID control

This section outlines the implementation and evaluation of the PID control strategy. First, we provide details on the design process and the selection of appropriate parameters.

The evaluation then focuses on analyzing the controller's performance across various test scenarios, comparing alternative configurations, and identifying key factors that influence the system's stability and responsiveness.

A. Model Architecture and Algorithm Implementation

The ACC system was implemented using a dual PID controller architecture, designed to separately manage acceleration and braking. The PID controller for acceleration calculates errors based on velocity, whereas the PID controller for braking utilizes the inverse of the TTC (1/TTC), a parameter derived from radar data that quantifies the urgency of potential collisions. Both controllers operate using a buffer of size 42, which stores the most recent errors to calculate the proportional, integral, and derivative terms as defined in Equation 1.

$$u(k) = K_p e(k) + K_i \Delta t \sum_{i=0}^{k} e(i) + K_d \frac{e(k) - e(k-1)}{\Delta t} \quad (1)$$

The system calculates the safety distance to maintain based on the vehicle's current speed. Depending on the deviation from the detected distance to the object with the lowest TTC, it operates in one of four defined operational modes :

1) **Cruise Mode:** When no obstacles are present, the system calculates the throttle value required to reach and maintain the target speed.
2) **Following Mode:** If a moving obstacle is detected, the vehicle adapts its speed to follow it while maintaining the safety distance.
3) **Braking Mode:** When the obstacle ahead is closer than the security distance, the system engages the braking PID controller to bring the vehicle to reduce velocity avoiding collision.
4) **Stationary Mode**: When the distance to the obstacle is less than a certain threshold, the vehicle remains braked.

In addition, it is possible to choose from three different actuation preferences, which will change the thresholds for switching between modes.

B. Model Selection and Parameter Tuning

Parameter tuning was conducted iteratively to balance responsiveness and stability. Key considerations included:

- **Proportional Gain (P):** Determines the reaction strength to the current error. High values improve responsiveness but risk overshooting.
- **Integral Gain (I):** Compensates for accumulated past errors. Excessive values can lead to instability due to integral windup.
- **Derivative Gain (D):** Dampens the system response to sudden changes, reducing overshoot and oscillation.

The values for K_P, K_I, and K_D were manually tuned through extensive testing to achieve optimal performance. Additionally, the buffer size of 42 and the sampling time of 0.01s were chosen after experimentation to ensure smooth control while avoiding excessive latency in error calculation.

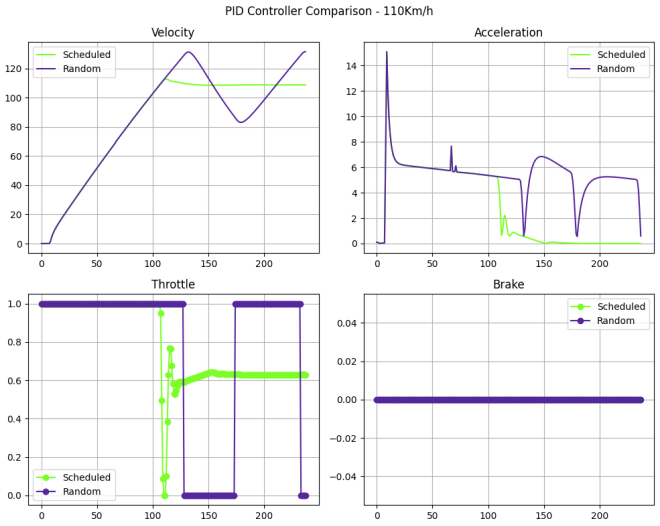

Fig. 1. Comparison between PID with Scheduled Sampling time and PID with Random Sampling time

The system was also tested at various target speeds, ensuring consistent behavior across different velocity ranges.

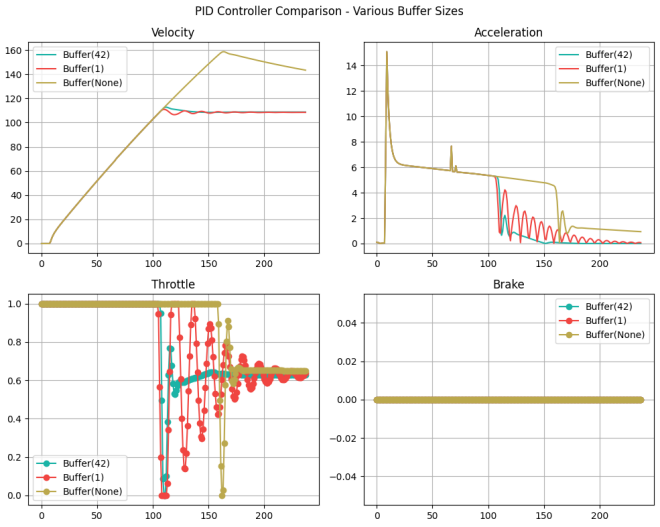

Fig. 2. Comparison between different Buffer Size

C. Experimental Results

Performance evaluation involved four primary scenarios:

1) **Scheduled vs. Random Sampling Time:** In the scheduled approach, error updates occurred at fixed intervals, while the random sampling approach updated errors at each iteration of the control loop. The scheduled sampling method demonstrated more consistent performance, with reduced oscillations and improved stability in throttle and brake responses. The result are shown in Fig. 1, 2, 3
2) **Buffer Size Consideration:** A critical factor in the system's performance is the choice of buffer size. The

buffer size impacts the calculation of the control action, influencing the overall responsiveness of the control system.

3) **Distance vs. 1/TTC for Error Calculation:** A comparison was made between using raw obstacle distances and 1/TTC as inputs to the Breaking PID controllers. The 1/TTC approach provided more reliable and precise control as it inherently captures both distance and relative velocity.
4) **Introduction of Hysteresis Control Mechanism:** An additional control mechanism was introduced, incorporating hysteresis to prevent rapid switching in the control actions. This helps to improve the overall robustness and smoothness of the control system.

Additional tests were also conducted using speeds expressed in meters per second (m/s) and kilometers per hour (km/h). No significant differences in performance were observed, so the latter was used.

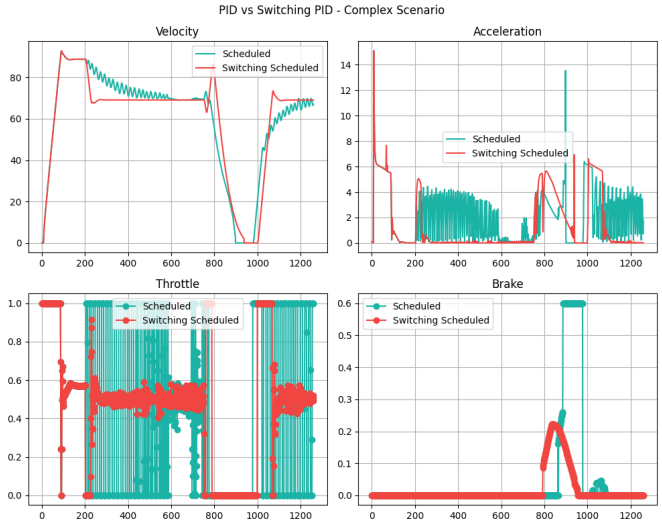

Fig. 3. Comparison between PID controller with and without Hysteresis

D. Challenges and Limitations

During implementation, several challenges were encountered:

- **Noise in Radar Data:** Variations in radar readings led to fluctuations in error calculations.
- **Integral Windup:** Prolonged errors in acceleration or braking modes caused an oversaturation of the integral term, requiring a bounded buffer mechanism to mitigate this issue.
- **Control Function:** In addition, a potential limitation is that the input to the PID is a simple operation, and not a (non-)linear function.

E. Future Developments and Improvements

Several future improvements are proposed to enhance the PID controller's performance. One promising direction is dynamic parameter tuning, where adaptive mechanisms adjust the PID gains in real-time based on current traffic conditions, allowing for more responsive and stable control. Additionally, integrating advanced sensing technologies—such as combining radar data with LiDAR and camera inputs—can significantly improve obstacle detection and classification accuracy. Another avenue involves exploring hybrid control approaches that merge traditional PID control with machine learning techniques, aiming to harness the strengths of both methodologies. Lastly, a more complex control structure could be adopted by combining multiple PID controllers, enabling the system to apply a more accurate and nuanced control function across different driving scenarios.

IV. Implementation and Evaluation of Reinforcement Learning Control

This section outlines the implementation and evaluation of a reinforcement learning (RL) control strategy, using the PPO algorithm within a GYM-based simulation environment. First, we describe the model architecture and the design of the RL agent, followed by its evaluation across various test scenarios. The evaluation focuses on comparing the RL controller's performance to traditional PID controllers.

A. Model Architecture and Algorithm Implementation

The ACC system was implemented using a dual-agent strategy. In accordance with the operational modes described in Section III-A, the first agent controls the cruise mode by adjusting the vehicle's speed based on the target velocity and the vehicle's current speed, analogous to the PID controller used for acceleration. The second agent is responsible for the remaining functions, specifically managing the following and braking mechanism, by considering safety distance and the proximity of the vehicle with the lowest TTC.

In this setup, the RL agents can control both throttle and brake learning it through interaction with the environment and feedback based on their actions.

B. Model Selection and Hyperparameter Tuning

For the model development, the primary focus was on defining the architecture of the RL agent rather than extensively tuning hyperparameters. However, key hyperparameters were set as follows:

- **Learning Rate:** A small learning rate of 3×10^{-5} was chosen, despite its slower convergence rate. A lower learning rate allows for more gradual updates, which may help stabilize the learning process.
- **Number of Steps**: 1024 steps were set to allow the agent sufficient interaction with the environment.
- **Batch Size:** A batch size of 64 was selected to ensure stability in learning.
- **Seed:** A fixed random seed was used during training to ensure reproducibility.
- **Evaluation Callback:** An evaluation callback was introduced to assess the model's performance at the end of each training epoch. If the new model showed improved

performance compared to the previous best-recorded model, it was saved.

For both agents, the action space was composed of a single action ranging from -1.0 (maximum brake) to 1.0 (maximum throttle), with actions for acceleration and braking being mutually exclusive.

The termination condition for the scenario was not defined by a negative event, but rather aligned with the completion of each training epoch.

For the first agent, the observation space includes the target velocity and the ego vehicle's velocity, and the reward function was defined as:

$$\text{reward}(observation) = -\left|\frac{\text{Target\_Speed - Ego\_Speed}}{10}\right| \quad (2)$$

For the second agent, the observation space includes the security distance and the nearest obstacle distance and the reward function was defined as:

$$\text{error}_d = \text{Security\_Distance} - \text{Obstacle\_Distance}$$

$$\text{reward}(observation) = \begin{cases} \frac{\text{error}_d}{10} & \text{if error}_d \leq 0, \\ \frac{1}{\left(\frac{-\text{error}_d}{10}\right)} & \text{if error}_d > 0 \end{cases} \quad (3)$$

For the first agent, the initial scenario involved spawning the ego vehicle with a random initial velocity between 0 and 180 km/h, while the target velocity ranged from 30 to 150 km/h.

For the second agent, the initial scenario involved spawning the ego vehicle with a random velocity between 0 and 150 km/h, with another vehicle placed ahead, at a safety distance based on the ego vehicle's initial velocity. The leading vehicle's speed was also randomly set within 0 and the ego vehicle's velocity.

C. Experimental Results

The performance was evaluated by comparing the RL-based model with the developed PID controller. Initially, the evaluation focused on the cruise mode performance to assess the potential of the RL controller. In Fig.4 the graph shows the results of this experiment.

Subsequently, a more complex scenario was tested by combining both RL agents and incorporating the same hysteresis improvement discussed in 4. This scenario allowed for a comprehensive evaluation of the overall performance of the RL controller, particularly compared to the PID controller, in a dynamic environment. The results are shown in Fig.5.

D. Challenges and Limitations

Several challenges were encountered during the implementation of the RL-based control:

- **Convergence Time:** The RL agent took significantly longer to converge compared to the PID controller. This was primarily due to the need for extensive training in a dynamic, continuous environment, which required many episodes for the agent's policy to stabilize.
- **Observation Space Design:** Designing an effective observation space that provides all the necessary information for decision-making was a non-trivial task. Balancing the amount of information without overwhelming the agent was crucial to achieving efficient learning.
- **Reward Design:** Crafting an appropriate reward function posed a significant challenge. A complex reward function could lead to degraded model performance, as the agent struggled to comprehend the situation properly.

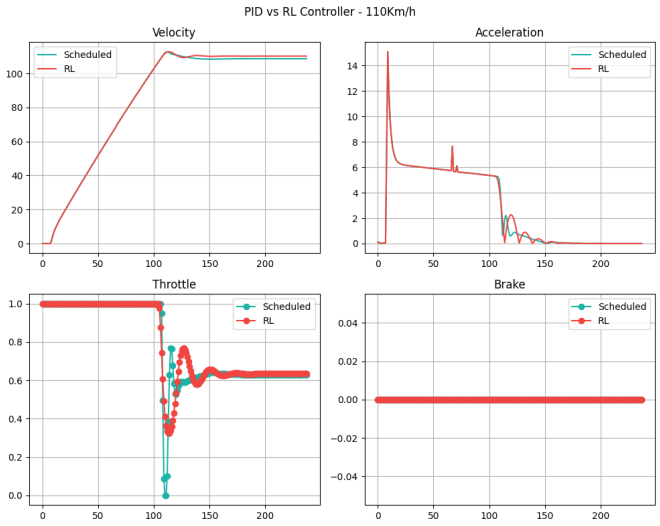

Fig. 4. Comparison between PID controller and RL controller

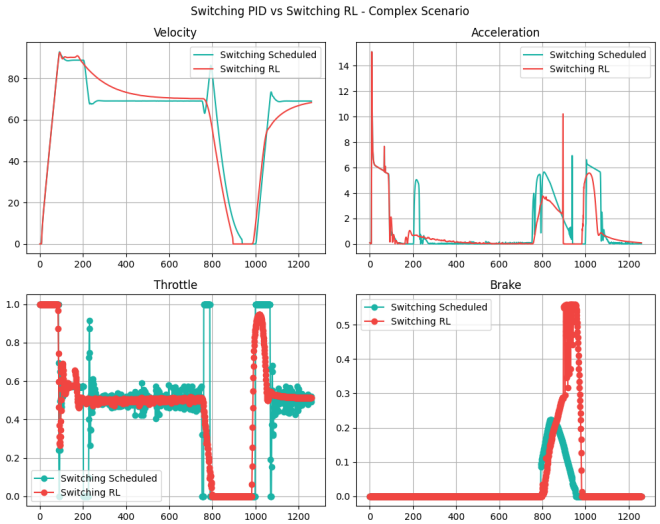

Fig. 5. Comparison between PID controller with Hysteresis and RL controller with Hysteresis

E. Future Developments and Improvements

Several improvements are proposed to enhance the performance of the RL-based controller. One possible direction

is the fine-tuning of a more complex model capable of simultaneously learning both acceleration and braking policies through an iterative approach, which could lead to more efficient and balanced behavior. Given the "black box" nature of reinforcement learning, it is also important to incorporate additional safety layers; for instance, static safety policies (i.e., triggering brake in risk scenarios) can act as safeguards, complementing the learned behavior.

Furthermore, to support real-world deployment, especially in resource-constrained vehicular environments, it is essential to explore Edge AI approaches. This involves implementing lightweight models that can operate on low-power embedded boards, potentially leveraging TinyML techniques [18], to ensure real-time decision-making without the need for constant connectivity or high-performance hardware.

In addition, supervised approaches [19] could be utilized to pre-train models on labeled driving data to learn foundational behaviors such as lane-keeping or obstacle avoidance to accelerate convergence and improve stability. On the other hand, unsupervised approaches [20], could improve the representation of driving environments. With both strategies, the controller can become more robust, responsive, and viable for practical applications in autonomous driving systems. Finally, the user interface design [21] and subsequent user validation [22] represent another important avenue for future research.

V. Conclusions

The PID controller demonstrated strong robustness, even when facing variations in setpoints. However, its performance was highly dependent on the careful management of switching between operational modes, requiring manual adjustments for optimal operation. In contrast, the RL-based controller outperformed the PID, particularly in more complex scenarios. It effectively handled both the following mechanism and the stop-and-go feature, leveraging a unique model that provided greater adaptability and effectiveness. An additional advantage of this controller is its ability to learn effective braking dynamics based solely on distance information, whereas the PID controller typically relies on TTC data, which provides strong semantic meaning for managing braking actions, especially in dynamic situations.

In conclusion, while the PID controller remains a reliable and stable option for simpler conditions, the RL-based controller offers enhanced adaptability and performance, especially in dynamic and complex driving environments, underscoring its potential for future autonomous driving systems.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement N. 101095882 (EBRT2030).

References

[1] M. M. Antony and R. Whenish, *Advanced Driver Assistance Systems (ADAS)*. Springer International Publishing, 2021, p. 165–181.

[2] L. Yu and R. Wang, "Researches on adaptive cruise control system: A state of the art review," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 236, no. 2–3, p. 211–240, May 2021.

[3] C. Knospe, "Pid control," *IEEE Control Systems*, vol. 26, no. 1, p. 30–31, Feb. 2006.

[4] A. Moujahid, M. ElAraki Tantaoui, M. D. Hina, A. Soukane, A. Ortalda, A. ElKhadimi, and A. Ramdane-Cherif, "Machine learning techniques in adas: A review," in *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*. IEEE, Jun. 2018, p. 235–242.

[5] M. Canale and S. Malan, "Robust design of pid based acc s&g systems," *IFAC Proceedings Volumes*, vol. 36, no. 18, p. 333–338, Sep. 2003.

[6] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," *Springer Nature Link*, 2021.

[7] G. Naus, R. van den Bleek, J. Ploeg, B. Scheepers, R. van de Molengraft, and M. Steinbuch, "Explicit mpc design and performance evaluation of an acc stop-&-go," in *2008 American Control Conference*. IEEE, Jun. 2008, p. 224–229.

[8] A. G. Alexandrov and M. V. Palenov, "Adaptive pid controllers: State of the art and development prospects," *Automation and Remote Control*, vol. 75, no. 2, p. 188–199, Feb. 2014.

[9] A. Simorgh, A. Marashian, and A. Razminia, "Adaptive pid control design for longitudinal velocity control of autonomous vehicles," in *2019 6th International Conference on Control, Instrumentation and Automation (ICCIA)*. IEEE, Oct. 2019, p. 1–6.

[10] R. Li, S. Deng, and Y. Hu, "Autonomous vehicle modeling and velocity control based on decomposed fuzzy pid," *International Journal of Fuzzy Systems*, vol. 24, no. 5, p. 2354–2362, Apr. 2022.

[11] J. Guo, Y. Wang, L. Chu, C. Bai, Z. Hou, and D. Zhao, "Adaptive cruise system based on fuzzy mpc and machine learning state observer," *Sensors*, vol. 23, no. 12, p. 5722, Jun. 2023.

[12] S. Mosharafian, M. Razzaghpour, Y. P. Fallah, and J. M. Velni, "Gaussian process based stochastic model predictive control for cooperative adaptive cruise control," in *2021 IEEE Vehicular Networking Conference (VNC)*. IEEE, Nov. 2021, p. 17–23.

[13] L. Das and M. Won, "D-acc: Dynamic adaptive cruise control for highways with ramps based on deep q-learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2021, p. 1572–1578.

[14] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[16] R. Yan, R. Jiang, B. Jia, J. Huang, and D. Yang, "Hybrid car-following strategy based on deep deterministic policy gradient and cooperative adaptive cruise control," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, p. 2816–2824, Oct. 2022.

[17] M. Gregurić, F. Vrbanić, and E. Ivanjko, "Impact of federated deep learning on vehicle-based speed control in mixed traffic flows," *Journal of Parallel and Distributed Computing*, vol. 186, p. 104812, Apr. 2024.

[18] G. Delnevo, S. Mirri, C. Prandi, and P. Manzoni, "An evaluation methodology to determine the actual limitations of a tinyml-based solution," *Internet of Things*, vol. 22, p. 100729, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.iot.2023.100729

[19] Z. Ling, G. Delnevo, P. Salomoni, and S. Mirri, "Findings on machine learning for identification of archaeological ceramics: A systematic literature review," *IEEE Access*, vol. 12, p. 100167–100185, 2024. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2024.3429623

[20] L. Liu, G. Delnevo, and S. Mirri, "Unsupervised hyperspectral image segmentation of films: a hierarchical clustering-based approach," *Journal of Big Data*, vol. 10, no. 1, Mar. 2023. [Online]. Available: http://dx.doi.org/10.1186/s40537-023-00713-8

[21] C. Ceccarini, S. Mirri, and C. Prandi, "Designing interfaces to display sensor data: A case study in the human-building interaction field targeting a university community," *Sensors*, vol. 22, no. 9, p. 3361, Apr. 2022. [Online]. Available: http://dx.doi.org/10.3390/s22093361

[22] C. Ceccarini, V. Nisi, and C. Prandi, "Exploring proximity-based recommendation criteria as a tool for information exchange and interactions between locals and tourists," *Multimedia Tools and Applications*, vol. 82, no. 4, p. 5229–5252, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1007/s11042-022-13369-y