Application of a Maneuver-Based Decision Making Approach for an Autonomous System Using a Learning Approach

Xin Xing, Sebastian Ohl
Faculty of Electrical Engineering
Ostfalia University of Applied Sciences
Wolfenbuettel, Germany
e-mail: {xi.xing | s.ohl}@ostfalia.de

***Abstract*—Autonomous driving technology has progressed significantly, necessitating advanced maneuver-based decision-making systems for complex driving environments. Traditional methods often fail in unpredictable real-world scenarios, leading to the adoption of learning-based approaches, such as Deep Learning (DL) and Reinforcement Learning (RL). This paper explores safety-critical car-following models and traffic management, focusing on Adaptive Cruise Control (ACC) and Automatic Emergency Braking (AEB). Traditional mathematical models are limited under extreme conditions, thus this study leverages machine learning to enhance vehicle responsiveness. Specifically, we apply RL to train car-following models. We emphasize policy-based RL methods, including Policy Gradient (PG) and Proximal Policy Optimization (PPO), within a simulated environment. The results demonstrate that PPO converges faster and exhibits fewer errors compared to PG. This study confirms that RL can effectively automate maneuver-based decision-making, highlighting the need for further research in diverse traffic conditions.**

Keywords-Autonomous Driving; Decision-making; Reinforcement Learning; Car-following models; Adaptive Cruise Control; Automatic Emergency Braking; Proximal Policy Optimization; Policy Gradient.

I. Introduction

Autonomous driving technology has made significant strides in recent years, driven by the imperative need for a decision-making system that can navigate complex and evolving driving environments. Traditionally, decision-making methods in autonomous driving have relied on robust, yet often rigid, frameworks that struggle to accommodate the unpredictable nature of real-world scenarios [1]. This limitation has led to the growing adoption of learning-based approaches, especially utilizing Deep Learning (DL) and Reinforcement Learning (RL), aimed at enhancing the adaptability and accuracy of Advanced Driver Assistance Systems (ADAS) [2][3][4]. Similarly, the *ExerShuttle* project is focused on the development of autonomous campus shuttle services that can transport passengers to desired locations within the campus, such as a library or cafeteria. This initiative aims to leverage intelligent driving technologies to enhance accessibility and convenience in campus environments.

A critical aspect of intelligent driving systems is car-following [5], which involves high-precision models crucial for ensuring driving safety, alleviating urban traffic congestion, and reducing the driver's workload. In this context, systems, such as Adaptive Cruise Control (ACC) [6] and Automatic Emergency Braking (AEB) [7] play pivotal roles. ACC adjusts the vehicle's velocity to maintain a safe distance from the car ahead, thereby easing the driver's burden. Conversely, AEB systems engage automatically to mitigate or prevent collisions by applying brakes when a potential risk is detected. The operational efficacy of both ACC and AEB is contingent upon accurate vehicle tracking models that respond promptly and reliably in varied driving conditions [8].

Traditional car-following models have largely been mathematical and, while useful, occasionally fall short under extreme conditions, thereby compromising safety. To overcome these limitations, large amounts of trajectory data are utilized and machine learning techniques are applied to reveal underlying patterns. These models, which include traditional Machine Learning, Deep Learning, and Deep Reinforcement Learning approaches, potentially enhance the responsiveness of vehicles to diverse driving scenarios, thus improving system accuracy and generalizability [9][10][11].

The paper particularly focuses on the use of Deep Reinforcement Learning (DRL) to train car-following models. RL is a policy-oriented decision-making method that aims to maximize rewards through trial-and-error behaviors, such as Policy Gradient (PG) [12]. DRL algorithms that integrate deep neural networks with RL principles, such as Deep Q-Networks (DQNs), have been demonstrated to effectively manage the complexity of ADAS algorithms and significantly enhance the system's ability to respond effectively to hazardous situations [2]. In [8], a novel RL-based longitudinal control and collision avoidance algorithm is developed that effectively takes into account the behavior of both the front and rear vehicles using the Deep Deterministic Policy Gradient (DDPG) model. The algorithm is shown to be capable of preventing potential serial collisions.

DQN and DDPG are mainly value-based methods, while PG and PPO [13] are direct policy optimization methods. DQN and DDPG usually require more samples for training, so they may encounter policy instability and convergence problems in actual training. In comparison, PG and PPO are more easily adapted to environments that have specific requirements on the form of the policy, such as scenarios that require the policy to output specific probabilistic information or continuous action [2]. Furthermore, PG and PPO demonstrate superior adaptability in environments where the policy must respond to dynamic or uncertain factors. This makes them well-suited for training car-following models, offering more robust and flexible solutions. This paper therefore emphasizes policy-based Reinforcement Learning methods, including PG and PPO algorithms, for training sophisticated car-following models.

The remainder of the paper is organized as follows: Section II discusses the problem formulation. Section III provides theoretical background. Section IV details the methodology. Section V evaluates the results, and Section VI concludes the paper and outlines future work.

II. Problem Formulation

In this paper, we explore the ACC and AEB systems within autonomous car-following models. The Autonomous Vehicle (AV) maintains a safe following distance from a Leading Vehicle (LV) or brakes urgently to avoid obstacles, such as a yellow duck used in simulations.

RL typically employs a Markov Decision Processes (MDP) to represent the interactions between the vehicle and its environment, taking actions based on the state of the environment and then receiving new states in response. For situations if states are not fully observable, a Partially Observable Markov Decision Process (POMDP) is employed to provide a more realistic representation of the state space. In order to design advanced policy using RL techniques, we formulate the switching between ACC and AEB as a POMDP.

A. States and Observations

To simplify the training model, the state and observation parameters are the AV's velocity, the LV's velocity, the gap between the AV and the LV or obstacle ahead, and the current action: $s_t = [V_{AV}, V_{LV}, G, A]$. The termination state is defined as when the AV collides with the LV or obstacle ahead, or if the LV's velocity is 0 and the gap between the AV and the LV or obstacle in front is less than the safe distance, which is calculated from the Time To Collision (TTC).

B. Actions

The action space consists of two elements, ACC and AEB: $A = [ACC, AEB]$. ACC adjusts the AV's velocity based on the Intelligent Driver Model (IDM) [14] to maintain a safe distance from the LV or an obstacle ahead. In contrast, AEB maximizes the AV's negative acceleration in order to halt the vehicle quickly when necessary.

C. Reward

Rewards depend on maintaining or breaching a safe distance between the AV and the LV or an obstacle. Safety enhances rewards, while penalties are assigned for risky proximities, balancing safety with comfort. The simulation terminates upon collision, adding penalties to prevent future rewards.

III. Background

RL consists of three components: the actor, the environment, and the reward function. The policy inside the actor determines the actor's actions, i.e., given an input, it outputs the action that the actor should now perform. All we have to do is to adjust the policy inside the actor so that the actor gets the maximum reward. The formulas presented below have been derived from [12] and [13].

A means of optimizing a policy π to solve the problem is provided by RL

$$\theta^* = \arg\max_{\theta} R(\pi_\theta), \tag{1}$$

where π_θ denotes a policy with parameters θ and $R(\pi_\theta)$ denotes the expected finite-horizon undiscounted return of the policy, often as a neural network.

A. Policy Gradient

PG method is a common method in RL. It uses gradient ascent to maximise the expected reward

$$\nabla R(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} [\Sigma_{t=0}^{T} \nabla \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)], \tag{2}$$

where τ is a trajectory and A^{π_θ} is the expected sum of rewards for the current policy. The policy parameter is updated via stochastic gradient ascent

$$\theta_{k+1} = \theta_k + \eta \nabla R(\pi_\theta), \tag{3}$$

where η is learning rate of neural network.

B. Proximal Policy Optimization

PPO is a state-of-the-art RL algorithm that belongs to the type of actor-critic algorithm. The actor is responsible for deciding which actions to take, while the critic is responsible for evaluating the actions taken by the actor. It is an on-policy algorithm, which means that it learns from the actions taken within the current policy, rather than from a separate set of data.

PPO is an improvement on the Trust Domain Policy Optimization (TRPO) [15] algorithm, which uses trust domain constraints to ensure that the new policy does not deviate too far from the old policy, thus providing stability. PPO-Clip builds on this idea by using a clipping function to limit policy changes. This allows PPO to make major policy updates while still maintaining stability.

The loss for the actor network is called Conservative Policy Iteration (CPI), which is the ratio between the policy under old parameters to the policy under new parameters multiplied by the advantage value

$$L_t^{CPI}(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}_t, \tag{4}$$

where π_{θ_k} is old policy parameter and $\hat{A}_t$ is the advantage reward value.

PPO-Clip adds an additional parameter ϵ. With the help of ϵ, the actor loss will be calculated by taking the minimum value between the cropped and uncropped values and multiplying it by the dominance:

$$L_t^{PPO}(\theta) = \mathbb{E}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}_t, \right. \right. \\ \left. \left. \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right]. \tag{5}$$

The critic loss is calculated as the Mean Square Error (MSE) between the predicted value estimate and the true value estimate. In other words, the critic loss is the MSE between the predicted value function and the true value function:

$$L_{\text{critical}} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{V}(s_i) - V(s_i) \right)^2 , \tag{6}$$

where $\hat{V}(s_i)$ is the predicted value function and $V(s_i)$ is the true value function for state s_i, and N is the number of samples.

The learning process in the context of PPO-Clip model is visualised in Figure 1. This model employs an actor-critic framework where two distinct networks are utilized: the Actor Network and the Critic Network. The Actor Network proposes actions given the current state of the environment, which are evaluated both by the environment and the Critic Network. The Critic Network estimates the value function of a given state, helping in the calculation of the Advantage Function, which measures how much better an action is compared to the average. The CPI ensures that the updates of the policy are kept within a certain range, preventing large policy updates that might destabilize learning.

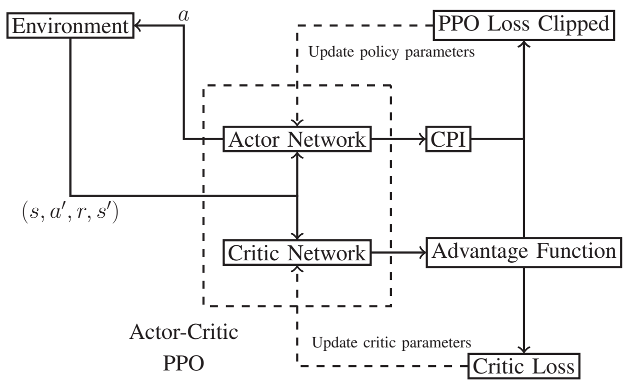

Figure 1. Reinforcement Learning PPO-Clip model

IV. Methodology

A. Simulation Environment

A simulated test environment constructed from the main test field of the *ExerShuttle* project is shown in Figure 2. The road network is mainly a closed road with two lanes. It has a maximum allowable velocity of 30 km/h and is connected to the 50 km/h road at the bottom of the figure. Since the test field is an university campus, the roadway will be relatively complex. There are private vehicles, buses, motorcycles, bicycles, and pedestrians on the road. There are no traffic lights at the 4-way stops road, so special attention must be paid to suddenly appearing vehicles and pedestrians. To reduce the reset time of the training environment, the simulated environment depicted in Figure 3. is used for training the model.

During training and evaluation, the IDM is used as the velocity control model for ACC. During the training period, the desired velocity of the vehicle is 30 km/h. The safe time

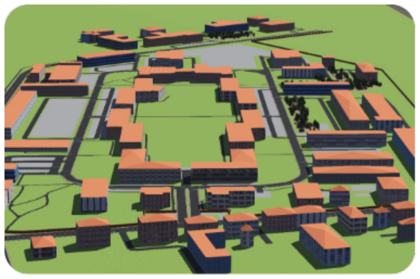

Figure 2. Simulation Environment of *ExerShuttle* Project

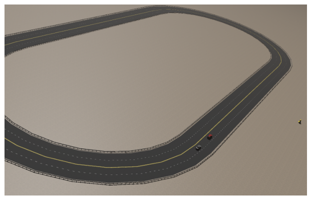

Figure 3. Simple Simulation Environment

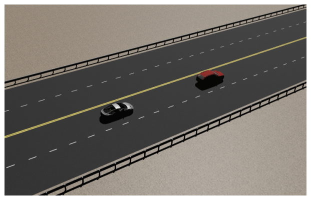

Figure 4. AV follows the LV if there is no obstacle

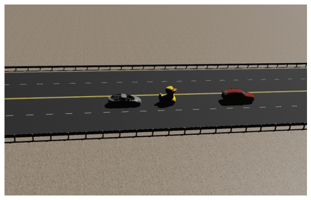

Figure 5. A yellow duck in front of the AV

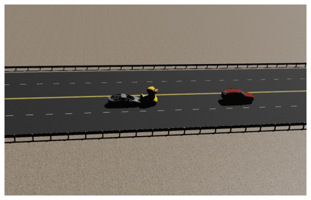

Figure 6. Collision between the yellow duck and the AV

headway is set to 1.5 s and the minimum distance is set to 7 m. The absolute values of both acceleration and negative acceleration are set to 1.5 m/s^2.

The training environment is a sequential sequence, as illustrated in Figures 4 to 6. During the training period, the AV will initially follow the LV, which is traveling at a speed of 20 km/h. After the AV has traveled for 4 s, an obstacle, such as a yellow duck, will randomly appear in front of the AV. If the car does not brake in time, the car will collide with the duck. In the absence of an obstacle, the LV will cease movement after 25 s. The AV must therefore be able to make the appropriate decision in a variety of circumstances.

B. Action Space

As stated in subsection II-B, the action space A includes only two actions: ACC and AEB. The AEB action is selected only if there is a sudden close-by obstacle in front of the AV, or if the LV applies an emergency brake. In the car-following model, the driving behavior of the AV is a comfortable driving behavior similar to human driving by setting the appropriate parameters of IDM. It is inadvisable to select the AEB action in inappropriate situations. For instance, the AV should have followed the LV using ACC, or the AV should have braked slowly and decelerated to a stop but emergency braking is selected instead. This also results in a significant decrease in passenger comfort. Mostly the AV uses ACC to follow the LV.

C. State Space

The gap between the AV and the LV and their velocities are included in the state space. The velocity of the AV is to be determined via the Global Positioning System (GPS) sensor, while that of the LV is to be gauged by the Radar. The distance of the obstacle in front of the AV will be determined by the distance sensor. The maximum range of the distance sensor is set to $50\,\mathrm{m}$. Obstacles or vehicles in front of the vehicle will be ignored when the distance is greater than $50\,\mathrm{m}$. This allows the vehicle to be driven at the maximum allowed velocity.

The selection of actions as part of the state can be described as history-dependent [16]. In partially observable environments, state information may not be sufficient to fully characterize the current state of the environment. By combining previous actions and states, an augmented state representation can be formed, which allows the policy to better capture the dynamics of the environment.

D. Reward Function

In an automated driving system, the reward functions of ACC and AEB are designed to ensure that the vehicle's behavior is safe and comfortable. Therefore, it is necessary to design the reward functions of ACC and AEB separately. The goal of ACC is to maintain a safe distance between the vehicles and the appropriate velocity. The AV should maintain a safe distance from the LV or from an obstacle, too close will be penalized:

$$R_{\mathrm{acc}_{\mathrm{dist}}} = \begin{cases} 10, & \text{if } D_{\mathrm{actual}} > D_{\mathrm{safe}} \\ -10, & \text{otherwise} \end{cases}. \tag{7}$$

where D_{actual} is the current distance between the AV and the LV and D_{safe} is Distance to Collision (DTC), calculated from the TTC. The AV maintains a consistent speed with the LV, whenever possible. Excessive velocity difference will be penalized:

$$R_{\mathrm{acc}_{\mathrm{velocity}}} = -k_1 \times V_{\mathrm{diff}}. \tag{8}$$

The primary objective of AEB is to prevent collisions and provide safe braking in emergency situations. The occurrence of a collision is subject to significant penalties:

$$R_{\mathrm{aeb}_{\mathrm{collision}}} = \begin{cases} -100, & \text{if collision} \\ 0, & \text{otherwise} \end{cases}. \tag{9}$$

In the event that the distance between vehicles is too close, the vehicle slows down quickly to avoid a collision:

$$R_{\mathrm{aeb}_{\mathrm{dist}}} = \begin{cases} -10, & \text{if } D_{\mathrm{actual}} > D_{\mathrm{safe}} \\ 10, & \text{otherwise} \end{cases}. \tag{10}$$

In addition, AVs should avoid using the emergency brake while following. Thus, the decision to take longer to follow a vehicle in the same following situation will be penalized more severely:

$$R_{\mathrm{T}} = -k_2 \times T. \tag{11}$$

Equation (12) is the total reward function. The k_1, k_2 in (8) and (11) are weight coefficients.

$$R_{\mathrm{totel}} = R_{\mathrm{acc}_{\mathrm{dist}}} + R_{\mathrm{acc}_{\mathrm{velocity}}} + R_{\mathrm{aeb}_{\mathrm{collision}}} + R_{\mathrm{aeb}_{\mathrm{dist}}} + R_{\mathrm{T}} \tag{12}$$

E. Training Architecture

The RL Environment is constructed using the OpenAI Gym [17] and the Webots simulator [18] in Python. Webots is an open-source application for simulating robots. It provides a Driver controller for controlling the vehicle and a Supervisor for modifying the parameters of the simulation environment. For instance, the Driver is capable of acquiring and controlling the vehicle's velocity and steering. The Supervisor is more powerful and interacts with the environment to obtain and set state variables, such as the position of an obstacle. The Driver and Supervisor sample state variables and select actions at a rate of $5\,\mathrm{Hz}$, which corresponds to a time step of $0.2\,\mathrm{s}$ in the simulation. In the event of a collision or timeout, the simulator is reset. A timeout is initiated after $100\,\mathrm{s}$ of simulation. The architectural framework is depicted in Fig 7. As presented in Subsection IV-A, Driver 1 and Driver 2 control the motion of the AV and the LV, respectively. The Supervisor obtains and transmits information about the vehicles, such as velocity or sensor data, by interacting with the parameters of the two Drivers. Furthermore, the Supervisor also controls the translation of obstacles in the environment. A gym-based simulation environment is used to train the models for RL.

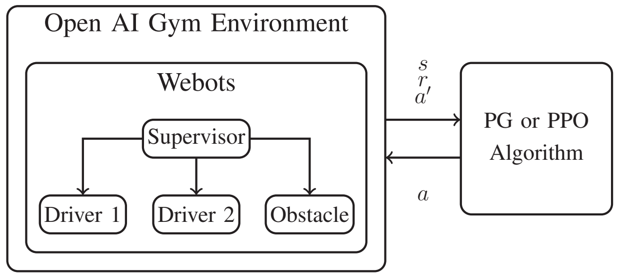

Figure 7. Training Architecture of Reinforcement Learning

The agents are trained using the online policy algorithms PG and PPO. A broader range of algorithms has not been evaluated, as our focus is on exploring the feasibility of using RL models for state switching in car-following models. In order to accommodate the training environment, the gym-based environment is rebuilt in Webots. The agent is trained using PPO-Clip for 1000 episodes on an Intel i9-8950HK and a NVIDIA Quadro P2000.

The neural network utilized during training comprises two hidden layers with a width of 256, as shown in Figure 8. The widths of the input and output layers correspond to the number of items in the state and action sets, respectively. Furthermore, the neural network is employed with ReLU and Softmax activation functions to streamline the computation and circumvent the gradient vanishing issue. The output is transformed into a probability distribution, which is also suitable for classification

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

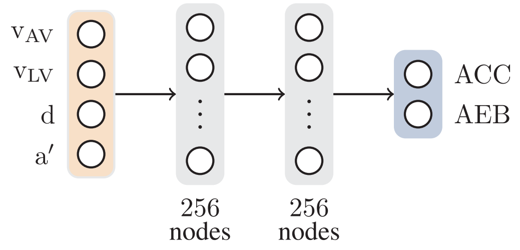

Figure 8. Simple Actor Network for PPO-Clip. Orange layer: inputs, blue layer: outputs, grey layers: hidden nodes.

tasks. Furthermore, the PPO experience pool is employed in the training process. At the conclusion of each iteration, the previous data set is discarded and a new round of data collection and training commences. The objective of this process is to ensure that when the policy is updated, the data collected based on the latest policy is utilized. The values of the key parameters of the PG and PPO algorithms are presented in Tables I and II, respectively.

TABLE I. AGENT PARAMETERS FOR PG

Parameters	*Value*
Learning rate	0.003
Discount factor	0.8

TABLE II. AGENT PARAMETERS FOR PPO-CLIP

Parameters	*Value*
Learning rate	0.0003
Number of steps	200
Number of epochs	10
Batch Size	2048
λ of GAE[a]	0.95
Clipping range	0.2
Discount factor	0.99

Note: [a]Generalized Advantage Estimation

V. EVALUATION

A. Training

The model successfully converges using both the PG and PPO-Clip algorithms. However, by adjusting the parameters of the training models, it is found that the PG algorithm is more likely to converge successfully than the PPO algorithm. For both implementations of the algorithm, the reward values begin at approximately -1800, as shown in Figure 9. The mean reward during training of the PG algprithm shwon in royal blue while the episode reward shown in light royal blue. The mean reward during training of the PPO algprithm shwon in orange while the episode reward shown in light orange. However, if the PG algorithm is employed, the reward value stabilizes at approximately 500 after approximately 150 episodes. In contrast, if the PPO-Clip algorithm is utilized, the reward value stabilizes at approximately 500 after about 50 episodes. It can be observed that the PPO model converges at a faster rate than the traditional PG model. This is primarily due to the fact that the PPO model limits the magnitude of change in the policy update step and avoids the introduction of excessive policy changes. PPO employs multi-step data sampling to optimize the policy and enhance the efficiency of data utilization.

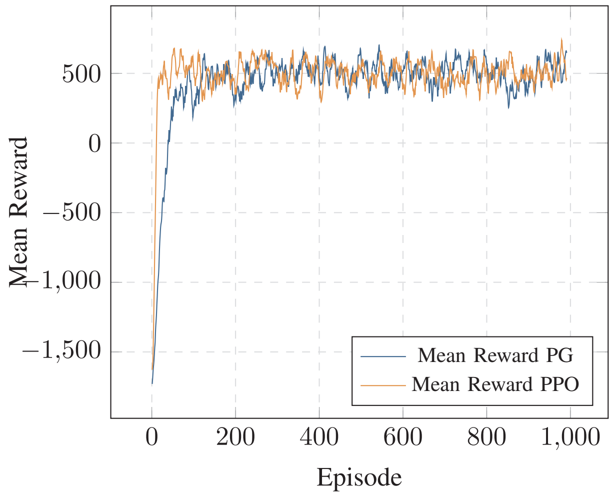

Figure 9. Training results of the PG and the PPO model

Furthermore, PPO employs additional optimizations when addressing rewards. These include the use of GAE to balance the variance and bias, and the estimation of the policy gradient with greater accuracy.

The action selection of ACC and AEB is relatively chaotic in the early stages of training, with an average of 100 to 150 episodes required for the reward value to remain stable. Once this occurs, the correctness of the action selection rate is greatly improved. Furthermore, the accuracy of the action selection also increased significantly and is maintained until the conclusion of the training period.

B. Results

The generated models has been subjected to evaluation in a simulation environment. For each evaluation, 1000 car-following tests are conducted and a random obstacle is placed in front of the AV. The results of the car-following tests are determined in two main ways: whether a collision occurred or not, and the selection of inappropriate behaviors. This includes instances where the AV braked inappropriately and collided with the obstacle and instances where the driving behavior suddenly chose the emergency braking action when it should follow the LV. If the AV is able to brake safely in front of an obstacle via the ACC system, it is also considered to be driving correctly. The results are presented in Table III.

TABLE III. COMPARISON OF DRIVING BEHAVIOR UNDER TWO ALGORITHMS

Algorithm	*Wrong behavior or Collision / %*	*AEB Selection / %*
PG	1.5	24.85
PPO	0.3	1.0

The trained models based on the PPO algorithm demonstrate superior performance overall. The mean number of erroneous behavioral choices or collisions across 1000 tests is 3. However, the mean number of instances in which the trained model based on the PG algorithm exhibited an error is 15.

When encountering obstacles, the PPO model rarely triggers the AEB, activating it less often than the PG model, which uses the AEB 24.85% of the time. This suggests that the PPO model relies primarily on the ACC system for emergency interventions. This difference can be attributed to the PG model's extensive exploration of both policy options during training, which helps it learn different emergency braking scenarios. In contrast, PPO's conservative update approach, characterized by clipped probability ratios and a targeted objective function, limits its exploration of certain actions, such as AEB. This conservative strategy may cause the PPO model to underutilize AEB in unforeseen scenarios during validation, resulting in less frequent use of emergency braking. However, this does not compromise the vehicle's ability to stop effectively, as it can still use either ACC or AEB to avoid collisions.

VI. Conclusion and Future Work

This study examines a maneuver-based decision-making approach in a simulation framework. The objective is to implement and test the selection of ACC and AEB in a car-following model using traditional PG and PPO algorithms. The results include:

- Both PG and PPO models are able to effectively select the ACC and AEB systems to follow the vehicle or emergency obstacle avoidance.
- The PPO algorithm converges faster, stabilizing at a reward value of 500 after about 50 episodes, compared to 150 episodes for the PG algorithm.
- Over multiple 1000 follow-up tests, the PPO-trained model have an average error rate of 0.3% for misbehavior or collisions, while the PG-trained model had an error rate of 1.5%.
- The simulations provide valuable insights showing that RL can automate maneuver-based decision making in driving is feasible.

The results thus far remain constrained by a number of limitations: The vehicle is capable of autonomously selecting between the ACC and AEB systems, utilizing a car-following model trained through RL. Nevertheless, the current simulation is trained in a relatively simple traffic environment. Consequently, future work should consider more diverse traffic conditions and improve the generality of the results by optimizing the training algorithm and adjusting the parameters. In addition to the ACC and AEB systems in the car-following model, it is also possible to consider the integration of systems for reasonable overtaking into the overall training environment.

The sensors utilized in vehicle simulations are still relatively simple in design. Consequently, if the results of these simulations are to be utilized in real-world environments in the future, it is imperative that the sensors employed in vehicle of the future be given greater consideration. In addition, the trained models will be validated and optimized in the *ExerShuttle* project in real world traffic. Further research should also concentrate on integrating a wider range of driving behaviours into the training models, followed by rigorous testing and validation in real-world conditions.

References

[1] F. Leon and M. Gavrilescu, “A review of tracking, prediction and decision making methods for autonomous driving”, *arXiv*, 2019.

[2] Z. Zhu and H. Zhao, “A survey of deep rl and il for autonomous driving policy learning”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 14 043–14 065, 2021.

[3] Q. Liu, X. Li, S. Yuan, and Z. Li, “Decision-making technology for autonomous vehicles: Learning-based methods, applications and future outlook”, in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 30–37.

[4] Y. Ye, X. Zhang, and J. Sun, “Automated vehicle's behavior decision making using deep reinforcement learning and high-fidelity simulation environment”, *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 155–170, 2019, ISSN: 0968-090X.

[5] M. Brackstone and M. McDonald, “Car-following: A historical review”, *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 2, no. 4, pp. 181–196, 1999, ISSN: 1369-8478.

[6] A. Vahidi and A. Eskandarian, “Research advances in intelligent collision avoidance and adaptive cruise control”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 143–153, 2003.

[7] L. Yang *et al.*, “A systematic review of autonomous emergency braking system: Impact factor, technology, and performance evaluation”, *Journal of Advanced Transportation*, vol. 2022, F. Galante, Ed., pp. 1–13, Apr. 2022, ISSN: 0197-6729.

[8] D. Chen, Y. Gong, and X. T. Yang, “Deep reinforcement learning for advanced longitudinal control and collision avoidance in high-risk driving scenarios”, *ArXiv*, 2024.

[9] P. Qin, H. Li, Z. Li, W. Guan, and Y. He, “A cnn-lstm car-following model considering generalization ability”, *Sensors*, vol. 23, no. 2, p. 660, 2023, ISSN: 1424-8220.

[10] T. Li and R. Stern, “Car-following-response-based vehicle classification via deep learning”, *ACM Journal on Autonomous Transportation Systems*, vol. 1, no. 1, p. 23, Mar. 2024.

[11] X. Yang, Y. Zou, H. Zhang, X. Qu, and L. Chen, “Improved deep reinforcement learning for car-following decision-making”, *Physica A: Statistical Mechanics and its Applications*, vol. 624, p. 128 912, 2023, ISSN: 0378-4371.

[12] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation”, in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12, MIT Press, 1999, pp. 1057–1063.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms”, *arXiv*, 2017.

[14] S. Albeaik *et al.*, “Limitations and improvements of the intelligent driver model (idm)”, *SIAM Journal on Applied Dynamical Systems*, vol. 21, no. 3, pp. 1862–1892, 2022.

[15] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization”, *arXiv*, 2015.

[16] G. Tennenholtz, N. Merlis, L. Shani, M. Mladenov, and C. Boutilier, “Reinforcement learning with history dependent dynamic contexts”, in *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, Honolulu, Hawaii, 2023.

[17] G. Brockman *et al.*, “Openai gym”, *arXiv*, 2016.

[18] O. Michel, “Webots: Professional mobile robot simulation”, *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.