Hindawi
Journal of Advanced Transportation
Volume 2023, Article ID 4127486, 12 pages
https://doi.org/10.1155/2023/4127486

Research Article

Learning to Drive in the NGSIM Simulator Using Proximal Policy Optimization

Yang Zhou [1] **and Yunxing Chen** [2,3]

[1]*School of Vehicle Engineering, Xi'an Aeronautical University, No. 259, Xi Erhuan Road, Xi'an 710077, China*
[2]*Hubei Key Laboratory of Power System Design and Test for Electrical Vehicle, Hubei University of Arts and Science, Xiangyang 441053, China*
[3]*School of Automotive and Traffic Engineering, Hubei University of Arts and Science, Xiangyang 441053, China*

Correspondence should be addressed to Yunxing Chen; chenyunxing@chd.edu.cn

Received 28 October 2022; Revised 24 November 2022; Accepted 25 March 2023; Published 3 April 2023

Academic Editor: Wenxiang Li

Copyright © 2023 Yang Zhou and Yunxing Chen. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As a popular research field, autonomous driving may offer great benefits for human society. To achieve that, current studies often applied machine learning methods like reinforcement learning to enable an agent to interact and learn in a stimulating environment. However, most simulators lack realistic traffic which may cause a deficiency in realistic interaction. The present study adopted the SMARTS platform to create a simulator in which the trajectories of the vehicles in the NGSIM I-80 dataset were extracted as the background traffic. The built NGSIM simulator was used to train a model using the proximal policy optimization method. The actor-critic neural network was applied, and the model takes inputs including 38 features that encode the information of the host vehicle and the nearest surrounding vehicles in the current lane and adjacent lane. A2C was selected as a comparative method. The results revealed that the PPO model outperformed the A2C model in the current task by collecting more rewards, traveling longer distances, and encountering less dangerous events during model training and testing. The PPO model achieved an 84% success rate in the test which is comparable to the related studies. The present study proved that the public driving dataset and reinforcement learning can provide a useful tool to achieve autonomous driving.

1. Introduction

It is wildly accepted that autonomous driving (AD) may help alleviate the problem of traffic congestion and reduce vehicle accidents and drivers' fatigue accompanied by manual driving [1, 2]. To achieve AD, researchers and car companies devoted themselves, and in recent years, the development of AD has witnessed big progress, and the commercialization of AD has been realized in some specific closed low-speed scenarios.

The key technologies of autonomous vehicles (AVs) include perception, decision-making, planning, and control [3]. Among them, decision-making which is also referred to as driving policy is responsible for deciding the behavior of AVs [4]. The driving policy of AVs takes the information collected from perception and outputs an appropriate action for the AVs. In real-world scenarios with complex road environments and dynamic traffic, it is vital to design a driving policy that considers the driving environment's uncertainty and negotiates with surrounding traffic safely.

Two common approaches have been applied to establish the driving policy of AVs [5]. One approach is the rule-based method which adopts traffic rules and expert knowledge to construct a rule library and an appropriate rule will be selected according to the current situation of AVs [6, 7]. As it is hard to consider all the situations in real-world situations, the rule-based method may not generate well when confronted with a new situation [8].

Another approach is the learning-based method. In contrast with manually designing the rules, the learning-based method forms the driving strategy from data automatically. The flourishing of machine learning enables using expressive models like neural networks to represent complex relationships like driving. Learning-based methods are attracting more attention in recent years [9, 10]. Imitation learning (IL), one of the famous learning-based methods, has been applied by researchers to achieve AD [11, 12]. The principle of IL is to directly learn the mapping between drivers' actions and the corresponding states. Despite the fact that IL has been proven to be effective in some studies, several shortcomings have also been found. First, IL requires collecting a large number of drivers' demonstrations which can be expensive and time-consuming. Second, the learned policy of IL may face the problem of the covariate shift problem [13].

Another type of learning-based method is reinforcement learning (RL). In RL, an agent learns from interacting with the environment in a trial-and-error form [14]. RL does not require collecting expert demonstrations, and the agent learns by maximizing the long-term returns which helps avoid the covariate shift problem in IL. Combined with deep learning, deep reinforcement learning (DRL) has been successfully applied to solve the game of GO [15], play Atari games [16], and accomplish loco-moto tasks [17]. In the application of applying DRL in AD, Zhang et al. [9] used the deep deterministic policy gradient algorithm to achieve automatic driving. The trained model can reach the defined goal and successfully avoid obstacles. Cai et al. [18] proposed an algorithm called DQ-GAT which combined deep Q-learning and graph attention-based networks to achieve safe and efficient autonomous driving in different urban environments. Shi et al. [19] sought to solve the problem of controlling AVs driving in urban unsignalized intersections using the proximal policy optimization (PPO) algorithm.

In the related studies, relatively simple and unrealistic background traffic was used in their simulator. Since the RL algorithms require interaction with the environment, an unrealistic environment may lead to unsafe or unrealistic behavior of the learned policy. In this study, the dataset from next-generation simulation (NGSIM) was extracted and used as the background traffic for the simulated environment. NGSIM provides, so far, the largest traffic dataset recorded by roadside cameras on US national highways [20]. The realistic and diverse features of the NGSIM dataset make it suitable for creating a simulator to train and test the RL algorithm to achieve AD. Therefore, in the present study, a simulating environment was built incorporating the NGSIM traffic dataset upon the SMARTs platform [21], and the DRL algorithm-proximal policy optimization (PPO) [22] was applied to realize AD in the highway scenario.

The contribution of this paper can be summarized as follows: (1) Introduce a data-driven approach to establish a realistic environment in which the background traffic is reproduced using the NGSIM dataset. (2) Propose a modern DRL algorithm (PPO) to train an agent to learn to drive in this environment. (3) Propose a way of state representation that extracts the most relevant information about the surrounding traffic. (4) Apply multiple indexes to analyze the training and test results of the trained models.

The rest of the paper is organized as follows: Section 2 briefly reviews the background of this study. Section 3 describes the architecture of the proposed model, the details of the state representation, and the proposed algorithm. Section 4 describes the baseline method and the evaluation metrics. Section 5 presents the training and the test results. The final section presents the discussion and conclusion.

2. Background

2.1. Reinforcement Learning. The Markov-decision process (MDP) is often used to model the sequential decision-making problem in RL. A MDP consists of a tuple $M = \{S, A, T, r, \gamma\}$. Specifically, S denotes the state space and A denotes the action space, $T(s^{'}|s, a)$ denotes the transition matrix which is the probability of transition from state s to $s^{'}$ after taking action a, r denotes the reward function which encodes the objectives or preferences of the agent in RL, γ is the discount factor.

The objective of RL is to seek the optimal policy $\pi^* = p(a|s)$ which has the maximum value $V^\pi(s)$ for all states $s \in S$ as illustrated in the following equation[14]:

$$V^\pi(s) = E_{a_t, s_{t+1}, \ldots}\left\{\sum_{l=t}^{\infty} \gamma_{l-t} r_l (s_t = s)\right\}, \tag{1}$$

where $V^\pi(s)$ denotes the expected total rewards when following policy π, a_t is sampled from π, and s_{t+1} is the next state when taking action a_t in state s_t which is determined by the transition matrix.

In model-free RL, there are three fundamental methods to solve the optimal policy which include the value-based method, the policy-based method, and the Actor-Critic method. Among them, the Actor-Critic method combines the advantages of the other two methods, and it has become the basis of the modern algorithms in RL [23].

In the framework of Actor-Critic, the actor is responsible for choosing the action of the agent in order to interact with the environment, and the critic is responsible for evaluating the agent's actions. The state-action value function $Q^\pi(s_t, a_t)$ and advantage $A^\pi(s_t, a_t)$ in actor-critic are defined in equations (2) and (3) respectively as follows:

$$Q^\pi(s_t, a_t) = E_{s_{t+1}, a_{t+1}}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right\}, \tag{2}$$

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - v^\pi(s_t). \tag{3}$$

2.2. Proximal Policy Optimization. PPO is one of the improved versions of the policy-based method which adopts the framework of actor-critic. PPO is well known for its prominent performance in a wild range of tasks [24]. It is therefore the first choice for OpenAIs projects.

There are two main features of PPO. First, it is found that the vanilla policy gradient often leads to extensive policy updates which may bring high variance and difficulty of convergence to the training of the model. PPO follows the theory of trust region policy optimization which is an earlier version of PPO and constructs a clipped surrogate objective to constrain the excessive updating of policies at every step of the policy gradient. The objective of the actor in PPO is as follows:

$$L_t^{\mathrm{CLIP}}(\theta) = \widehat{E}_t\left[\min\left(\rho_t(\theta)\widehat{A}_t, \mathrm{clip}\left(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\widehat{A}_t\right)\right], \tag{4}$$

where $\rho_t(\theta)$ is the probability ratio defined as $\rho_t(\theta) = \pi_\theta(a|s)/\pi_{\theta_{\mathrm{old}}}(a|s)$ and $\mathrm{clip}(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon)$ clips the values of $\rho_t(\theta)$ outside of the range $[1-\varepsilon, 1+\varepsilon]$. ε is a hyperparameter which is recommended to be 0.2 and θ is the parameter of the actor.

Secondly, generalized advantage estimation (GAE) is required for the calculation of the gradient of the PPO algorithm. A linear combination of n-step bootstrapping is used in GAE to get a low bias and variance estimation of $Q^\pi(s_t, a_t)$, defined as follows:

$$Q^\pi(s_t, a_t) = \sum_{l=t}^{\infty}(\gamma\lambda)^{l-t}\delta_l + V(s_t, \omega), \tag{5}$$

where δ_l is the temporal difference (TD) error defined as follows and ω is the parameters of the critic.

$$\delta_t = r_l + \gamma V(s_{l+1}, \omega) - V(s_l, \omega). \tag{6}$$

For the whole Actor-Critic, the loss function which combines the clipped loss for the actor and the squared error for the critic is defined in equation (7).

$$L(\theta) = \widehat{E}_t\left[L_t^{\mathrm{CLIP}}(\theta) - c_1 L_t^{VF}(\theta) + c_2 E[\pi_\theta](s_t)\right], \tag{7}$$

where $L_t^{VF}(\theta)$ denotes the loss for the critic which is calculated as $L_t^{VF}(\theta) = (\sum_{t'>t}\gamma^{t'-t}r_{t'} - V_\theta(s_t))^2$; $E[\pi_\theta](s_t)$ denotes the entropy of the policy π_θ; and c_1, c_2 are hyperparameters which are set to be 0.5 and 0.01, respectively.

2.3. NGSIM Simulator. A driving simulator with high fidelity is crucial to training an RL agent to learn to drive. Current studies make use of open-source simulators like CARLA [25] and SUMO [26]. Despite the many successful applications that have been made using these simulators, none of them has addressed the problem of multiagent interactions in driving. Interaction with diverse road users is found to be the current challenge for AV. SMARTS is developed to enable a realistic and diverse multiagent interaction to help the research community to solve the interaction challenge in AD [21]. The key features of SMARTS involve realistic physics supported by the PyBullet physics engine, traffic simulation with SUMO integration, and web-based visualization with recording.

The present study adopted the SMARTS platform to build the NGSIM simulator. The NGSIM I-80 dataset was extracted and integrated as the background traffic. The NGSIM dataset is so far unique in the study of traffic which has been wildly applied and analyzed by lots of researchers [27, 28]. The I-80 dataset is one of three public datasets included in the NGSIM dataset. The I-80 dataset contains three 15 minutes periods which are 4:00 p.m. to 4:15 p.m., 5:00 p.m. to 5:15 p.m., and 5:15 p.m. to 5:30 p.m. [20]. These periods represent the situation of traffic before rushed hours, the transition to rushed hours, and during rushed hours. The camera located on the roadside recorded the video of the traffic in the monitored area, and the trajectories of the vehicles were extracted from the recorded video. The extracted dataset contains 3366 vehicle trajectories. For each vehicle, speed, longitudinal and lateral position, vehicle length and width, the ID of the following vehicle, and the lead vehicle in the current lane are included in the data.

As illustrated in Figure 1, to set up the NGSIM simulator, first, the road map of the I-80 road segment needs to be created. The specific road alignment parameters of the I-80 road were thoroughly investigated and written in SUMO road network format which is supported by SMARTS. The road is about 310 m in length and has 6 lanes which include a merging ramp lane. Then, the SMARTS scenario studio library was applied to generate the traffic. The routes of every vehicle in the traffic were assigned according to the historical trajectories recorded in the I-80 dataset. Finally, when the NGSIM driving simulator is used for training, a random vehicle in the traffic is selected as the host vehicle which is to be controlled by the RL agent. The bicycle model is employed as the kinematics model for the host vehicle. The motion state of the other vehicles in the traffic is updated according to their history trajectories which have been smoothed by an extended Kalman filter. Since there are 3366 vehicles included in the traffic, a random pick of one vehicle in the traffic as the host vehicle can bring high diversity for the simulator which facilitates the training and testing of the model.

During simulation using the NGSIM simulator, the agent library of SMARTS provides rich sets of observations. A simulating radar is used to collect information on nearby vehicles. The events of the host vehicles such as collisions, off roads, and driving in the wrong direction are also provided. SMARTS also provides a visualization tool called envision which enables visually checking the simulation with the web browser.

3. The Proposed Model

As mentioned, the PPO algorithm was selected to train an agent to learn to drive in the aforementioned NGSIM environment. This section will introduce the network architecture of the actor-critic which is the core component of PPO, the design of state representation, and the reward function.

3.1. The Architecture of Actor-Critic. After being tested, the architecture of the actor-critic was determined as demonstrated in Figure 2. The input of the actor-critic is the states including 38 features. It represents the information that the agent observes during driving. The actor and critic networks have a similar structure which all have two hidden layers with 200 units except that the last layer of the actor uses a tanh activation function to convert the output value to the range of [−1, 1]. The outputs are then multiplied by the max range of each action to determine the proper range of each action. The critic outputs the value of the present state, and the actor outputs the action which is going to be taken by the agent in the present state. Fully connected networks (FCNs) and tanh activation functions are used throughout this architecture.

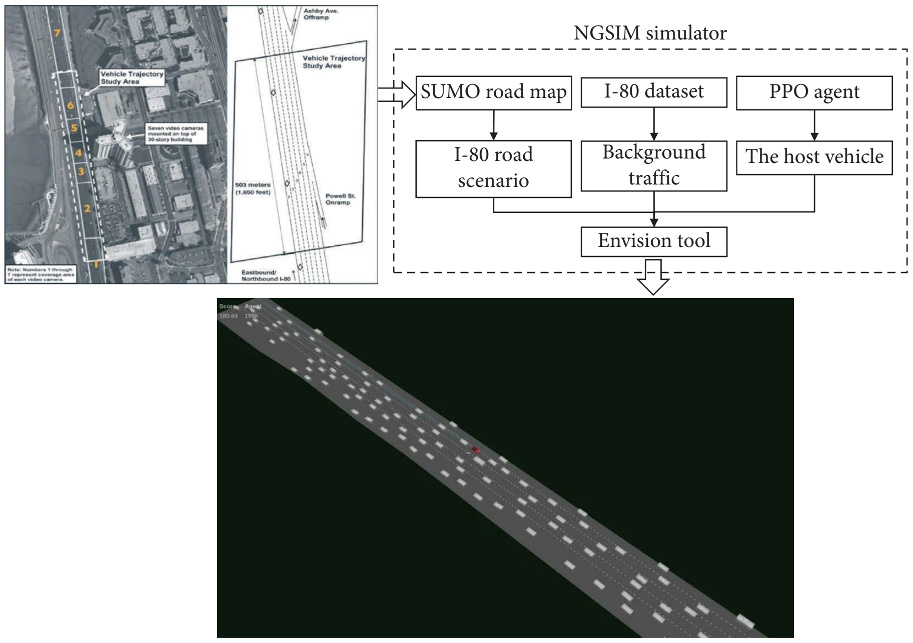

Figure 1: The framework of the NGSIM simulator.

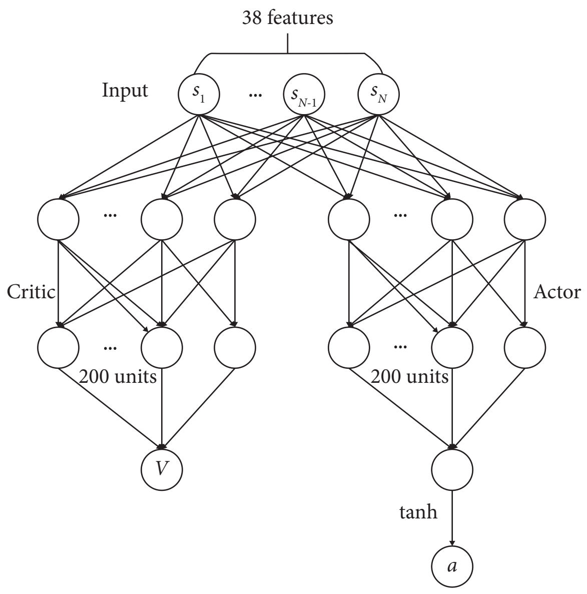

Figure 2: The architecture of the actor-critic.

3.2. State Representation and Action Space. The state space should contain all the necessary information for the agent to decide on the appropriate action. In the present study, the state is represented by 38 features grouped into two parts, as illustrated in Table 1. The first part contains features about the host vehicle. The second part is mainly about the surrounding traffic. Different ways of representing the surrounding traffic were tested. To extract the most relevant information and to reduce the dimension of the state space, the nearby vehicles provided by the NGSIM simulator were further filtered to the six nearest vehicles in blue, as illustrated in Figure 3. The six nearest vehicles include the vehicle in front and behind on the left and right adjacent lanes, the lead vehicle, and the following vehicle in the same lane. Then, Euclid distance d, relative speed defined as $\Delta v = v_1 - v_e$, longitudinal distance d_x, and lateral distance d_y between the host vehicles in yellow and the surrounding six vehicles were calculated. Time to collision (TTC) defined as TTC $= d/\Delta v$ was also calculated to represent the risk of collision. It should be noted that the distance here refers to the distance between the bounding boxes of the two vehicles which were calculated by considering the width and length of the vehicles.

The present study adopted a continuous action space. The action space includes acceleration and yaw rate. To limit the value of the action to a reasonable scope, the max range

Table 1: State representation.

Types	The description of features
The host vehicle	Longitudinal position, lateral position, heading, longitudinal speed, later speed, lane offset, the distance to the leftmost lane boundary, and the distance to the rightmost lane boundary
The surrounding vehicles (6 nearby vehicles)	Distance, relative speed, TTC, longitudinal distance, and lateral distance

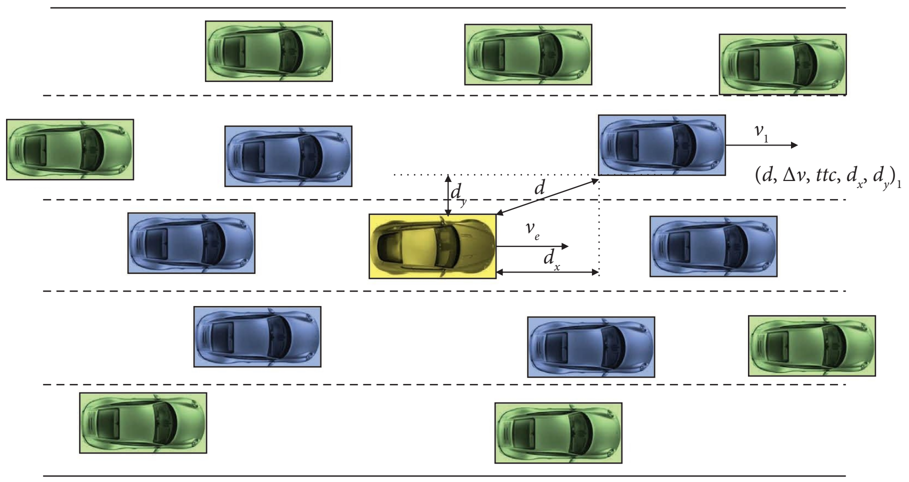

Figure 3: State representation of the surrounding vehicles.

of acceleration was set to the range of [−3, 3], and the max range of yaw rate was set to the range of [−1, 1] following the study of Xiao-fei et al. [8].

3.3. Reward Function. The design of the reward function is vital for ensuring the convergence and the performance of the RL algorithm. The reward function should encode the objective of the agent. The present study first tried a sparse rewards configuration. The agent received a large reward when reaching the destination and a negative reward when occurring the aforementioned dangerous events. However, the results showed that the agent could not learn effectively in the NGSIM simulator which has high-dimensional observations.

Therefore, the present study used a shaped reward function. After being tested carefully, the reward function was determined to be a linear combination of the three parts below.

3.3.1. Speed Reward. The speed of the host vehicle should not exceed the speed limit of the road. Therefore, the speed reward R_v is set to discourage the agent to violate the speed limit. When the speed of the agent exceeds the speed limit, the agent receives a negative reward of −1, otherwise, the agent can get a small reward of 0.01.

3.3.2. Lane-Keeping Reward. Lane keeping is the most common task in daily driving, especially in the NGSIM simulator which has rather congested traffic. To encourage the agent to maintain the lateral position around the lane center, the lane-keeping reward R_l is calculated by a gauss function as equation (8). When the vehicle drifts far from the lane center, the reward gets smaller:

$$R_l = \frac{1}{\sqrt{2\pi\sigma}} e^{-(d_l-\mu)^2/2\sigma^2}, \tag{8}$$

where d_l denotes the distance to the center of the current lane and μ and σ denote the mean and the variance of the lateral position being set to 0.9 and 0.2, respectively.

3.3.3. The Terminal Reward. When the ego vehicle reaches the terminal states, such as colliding with other vehicles, reaching the destination, or driving off the road, the simulation ends and restarts from a random initial position. The terminal reward is defined in the following equation:

$$R_t = \begin{cases} -5 \text{ if collision, dr iving in the wrong di rection or off road,} \\ 20 \text{ if reaching the de stination,} \\ 10 \text{ if the host vehicle safely dr ives for a certain period.} \end{cases} \tag{9}$$

The overall reward function is defined as equation (10). A constant c is introduced in the formulation to encourage the agent to continue to explore.

$$R = \alpha_1 R_v + \alpha_2 R_l + \alpha_3 R_t + c, \tag{10}$$

where α_1, α_2, and α_3 are the weights of the speed reward, the lane-keeping reward, and the terminal reward, respectively.

3.4. The Proposed Algorithm. The proposed algorithm is presented in Algorithm 1 below. PyTorch was used for implementing the proposed model. The Adam optimizer was used for training. Several tricks were implemented to facilitate the training of the model including using a linear decay of the learning rate, orthogonal initialization of the networks' parameters, and normalization of the state input [29].

3.5. Hyperparameters. The choice of hyperparameters is an important factor that may greatly affect the performance of the RL algorithm. The hyperparameters used in this study are carefully tuned to achieve the best training performance as listed in Table 2.

4. Model Investigated and Evaluation Metrics

4.1. A2C. A2C which stands for advantage actor-critic was proposed to solve the problem of high variance in the original actor-critic method [30]. The algorithm estimates the advantage by subtracting a baseline value from the reward. A2C was found to be more stable on training and effective for continuous problems. Therefore, A2C was chosen as a baseline method in the present study to be compared with PPO.

The same network architecture used in the PPO model as described above was adopted for the A2C model for a fair comparison. The state input, action space, and reward design are all the same as those of the PPO model. RMSProp optimizer was used to train the A2C model, and the learning rate was set to $5e-5$. The hyperparameters used in A2C were set according to the original paper [30].

4.2. Evaluation Metrics. To evaluate the performance of the model, three kinds of metrics were adopted as follows:

(1) As the most representative index in RL, the mean trajectory rewards which were calculated by averaging the accumulated rewards in every trajectory in an episode were selected as an index. The RL agent should collect more rewards during the training process following the objective of RL.

(2) To evaluate the agent's ability to drive in the NGSIM simulator safely, the mean distance traveled by the agent in each episode was chosen as another metric. A longer distance traveled by the agent represents a better skill in negotiating with road traffic.

(3) The respective numbers of the three different dangerous events including colliding with other vehicles, driving in the wrong direction, and driving off the road, were selected as metrics.

5. Results

Figure 4 presents the change in mean trajectory rewards for the PPO and A2C models during training. Global steps which represent the total steps that the agent interacts with the NGSIM simulator are used as the X-axis. As can be seen, the mean trajectory rewards increased gradually for both models as the training process. The reward curve fluctuates, but it is very common in the training of the RL model [22]. The PPO agent collects much higher mean trajectory rewards than the A2C agent. For the PPO model, the mean trajectory rewards tend to stabilize after training about $3e5$ global steps revealing that the model converges.

Figure 5 shows the trend of the mean travel distance during training. It can be seen that the mean travel distance increases with the same trend as the growth of the mean trajectory rewards. The results prove that the PPO agent and A2C agent can both learn to improve the skill of driving during training. The PPO agent maintains a longer distance than the A2C agent throughout the training process which fits well with the trend of the mean trajectory rewards reflected in Figure 4. For the PPO agent, in the beginning, the agent can only drive about 30 m on average, however, when the model converges at about $3e5$ global steps, the mean travel distance can increase up to 310 m in some episodes. The value of the mean travel distance fluctuates around 280 m at the end of training for the PPO agent. Since the overall length of the road is about 310 m, the result shows that the trained PP0 agent can nearly finish the NGSIM environment.

To test the trained models, the simulation was repeated 100 times running the two models in the NGSIM environment respectively. Table 3 lists the test results including the success rate, the mean travel distance, and the respective number of the aforementioned events. During the test, the PPO agent reached the destination 84 times. The remaining 16 times in which the agent did not finish the scenario are mainly collisions. The mean travel distance is 281.37 m which is very similar to the mean travel distance at the end of training. It demonstrates that the trained model has a fair generalization ability. As a comparison, the A2C agent only has a success rate of 38%. The mean travel distance is 122.95 m which is much less than that of the PPO agent. The A2C agent encounters a large number of collisions in the test.

The results above all demonstrated a superior performance of the PPO model compared with the A2C model. Therefore, the rest of this section only presents further analysis of the PPO model. During the test, speed, headway, and relative speed were recorded and analyzed to gain a better understanding of the PPO model. The test simulation includes about 30613 data points which are about 51 minutes in total. The normal distribution was applied to fit the distribution of the aforementioned data.

Figure 6 presents the distribution of speed from the test data. The mean speed is 7.75 m/s and the standard

```
Input: Randomly initialize the parameters of the Actor-Critic as $\theta^0$, the initial learning rate $l_r$
For $i = 0$ to $N_1$, repeat the following steps
    Using the policy $\pi_i$ to interact with the NGSIM environment for $N_2$ steps, record the trajectories of the agent as $D_i = \{\tau_1, \tau_2, \ldots, \tau_k\}, \tau_k = \{s_1, a_1, s_1, a_2, \ldots, s_j.a_j\}$, calculate the reward according to equation (10) for every state in the trajectories.
    Compute advantage $\hat{A}_t$ using GAE.
    Compute the gradient according to equation (7) with $K$ epochs and minibatch size $N_3$, and update $\theta_i$ using Adam optimizer.
    Linearly decay the value of the learning rate $l_r = l_r(1 - i/N_1)$
End
```

Algorithm 1: PPO.

Table 2: The hyperparameters.

Symbols	Meaning	Values
N_1	The total training episodes	200
N_2	The total simulation steps (batch size)	2048
l_r	Learning rate	0.0003
K	The number of repetitions of PPO training	10
N_3	Minibatch size	256

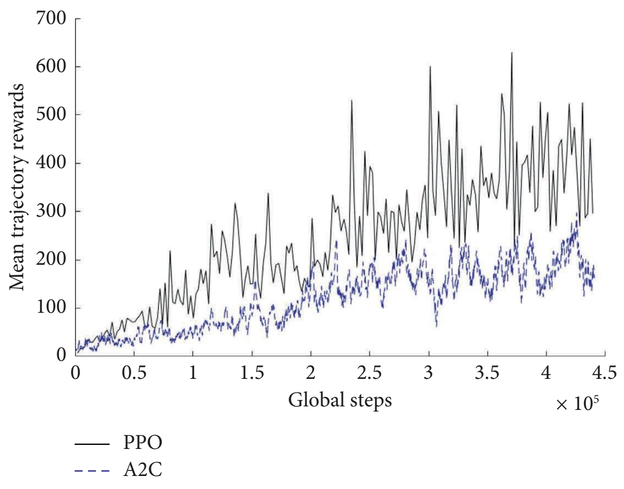

Figure 4: The change of the mean trajectory rewards.

derivation is 3.76 m/s. The relatively low speed indicates the simulated environment has very congested traffic. Figure 7 presents the time headway distribution during the test. The data with a time headway (TH) higher than 10 s were excluded which leaves about 30177 data points. As can be seen, the value of TH centralizes around 2.8 s with only a small part distributed below 2 s. The distribution of TH shows that the PPO agent can maintain a safe distance from the lead vehicle.

Figure 8 shows the distribution of TTC. Data points with TTC higher than 8 s and below 0 were excluded. The remaining 1880 data points were used for analysis. The majority of TTC is above 3 s revealing a relatively safe following strategy of the host vehicle. However, a small part of TTC is below 1.5 s which represents a high probability of collision, and some of them did eventually lead to a collision.

To show the performance of the trained agent, the visualizations of two simulations are presented below. As shown in Figure 9, the host vehicle in red (ID 1695) was initialized in a position as keyframe 1 (Figure 9(a)), it was maintained in the same lane, and kept a proper distance from the front vehicle until the end of the road. In Figure 9(c), it can be seen that the front traffic was congested, and the agent even learned to stop for a little period to wait for the front traffic to move.

Figure 10 shows the simulation of the host vehicle merging into the adjacent lane. Starting from the rightmost lane (Figure 10(a)), the host vehicle in red has to merge into the left adjacent lane because the rightmost lane will narrow ahead. As can be seen, the agent successfully negotiated with the surrounding vehicles and merged into the target lane (Figure 10(d)). However, it should be noted that during our test described above, most collisions happened when the host vehicle was initialized in the rightmost lane and had to merge into the left traffic. The success rate of merging is lower than the average.

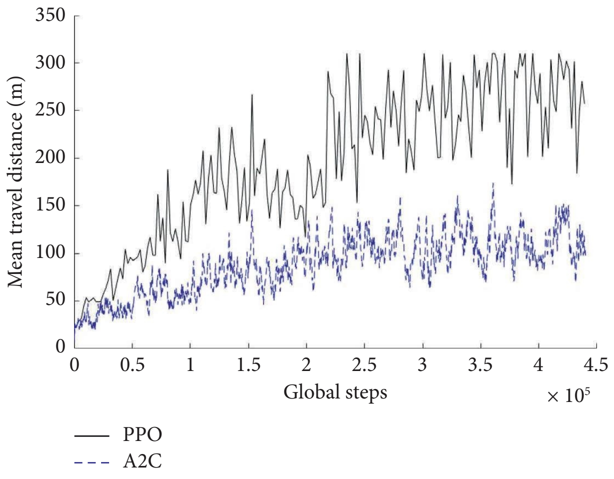

Figure 5: The change of the mean travel distance.

Table 3: The test results.

Methods	Metrics				
	Success rate (%)	Mean travel distance	The number of collisions	The number of off roads	The number of people driving in a wrong direction
PPO	84	281.37	15	0	1
A2C	38	122.95	49	5	8

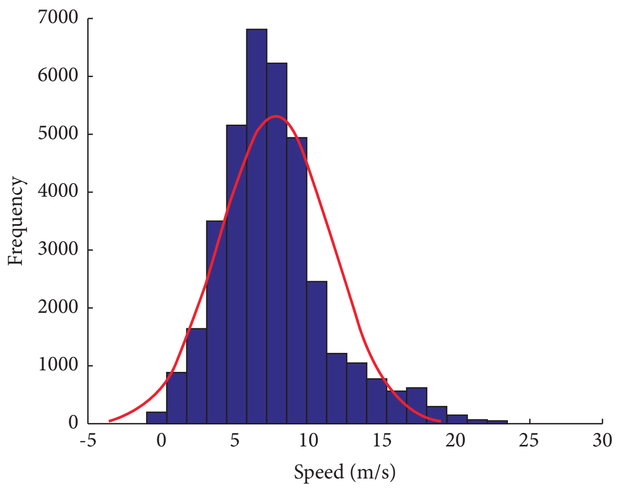

Figure 6: The distribution of speed.

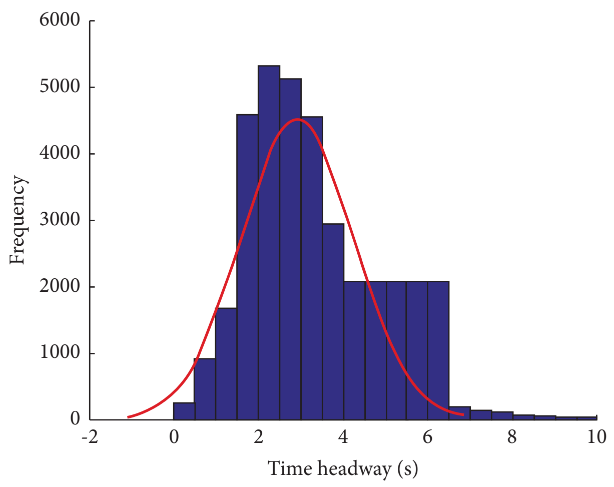

Figure 7: The distribution of TH.

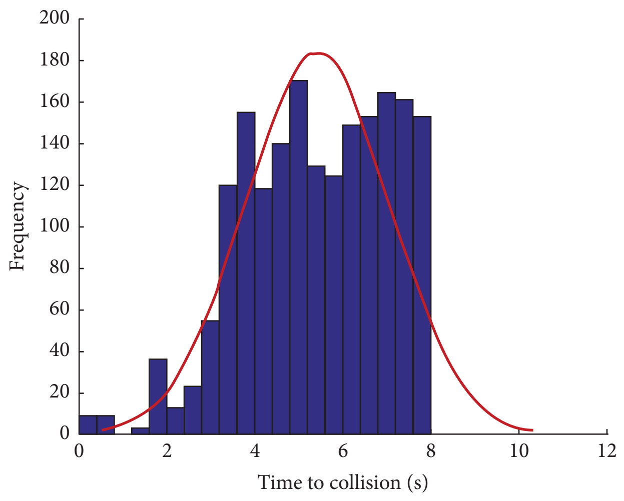

Figure 8: The distribution of TTC.

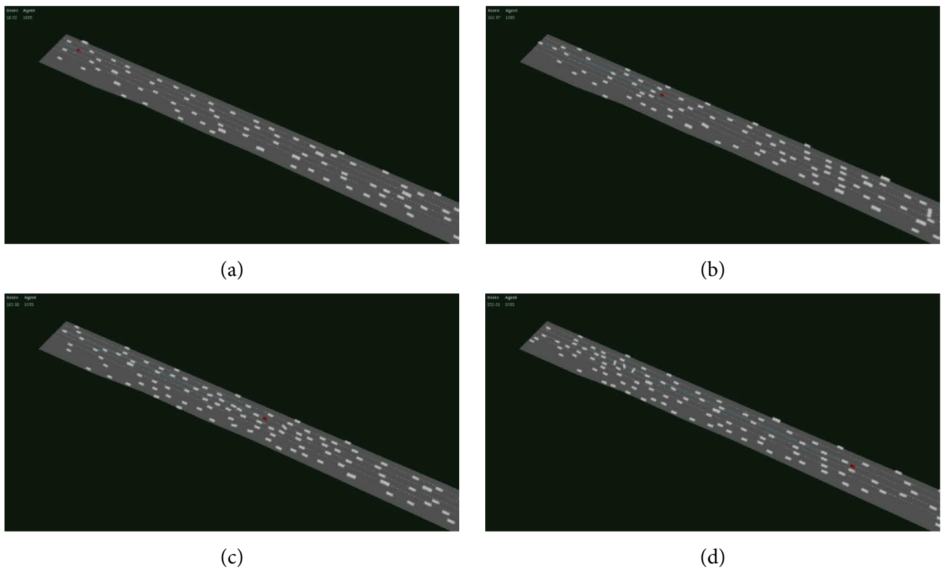

Figure 9: Simulation 1. (a) Keyframe 1. (b) Keyframe 2. (c) Keyframe 3. (d) Keyframe 4.

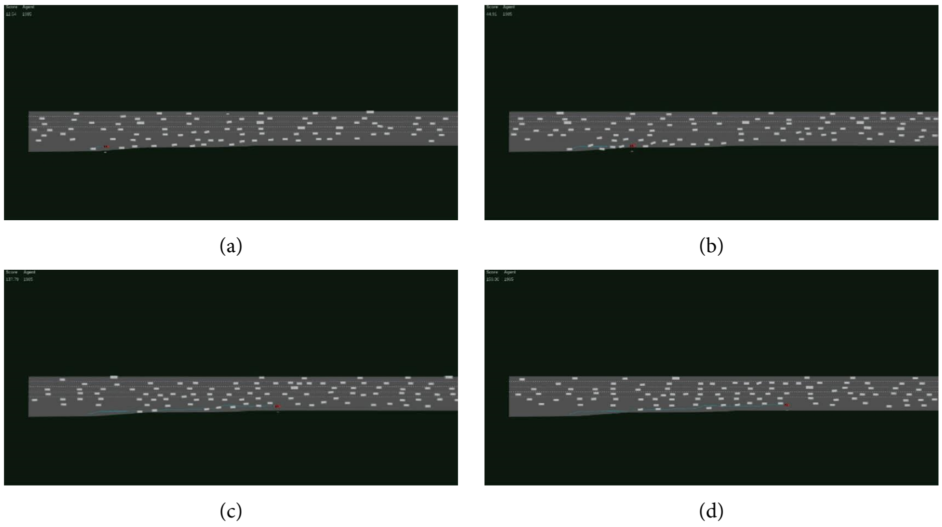

Figure 10: Simulation 2. (a) Keyframe 1. (b) Keyframe 2. (c) Keyframe 3. (d) Keyframe 4.

6. Discussion and Conclusion

In the present study, the SMARTS platform which features multiagent interaction was adopted to build a simulator for training an agent to learn to drive. The NGSIM I-80 dataset was applied to generate the background traffic for the simulator. The built-in NGSIM simulator has high diversity in which the vehicle in the background traffic is randomly picked as the host vehicle to be controlled by the agent. The PPO algorithm was applied to train the agent, and the proposed model used an actor-critic neural network. 38 features are selected as inputs, including 8 features related to the host vehicle and 30 features associated with surrounding vehicles. The linear combination of three parts of rewards is used as the reward function, which includes the reward of punishing the vehicle for exceeding the maximum speed, the reward of encouraging lane maintenance, and the relevant reward for the termination state. Another DRL method A2C was selected as a baseline algorithm for comparison.

The results showed that the PPO method outperformed the A2C method both in the training and test phases. The results from the present study are in line with the study of Schulman et al. [22] in which better performance was found for PPO compared with other algorithms on different continuous control environments. As for the PPO model, when it converged, the mean trajectory reward and the mean travel distance increased greatly. The trained PPO model achieved an 84% success rate and a 281.37 m mean travel distance in the model test. As reported by Chen et al. [31], the trained model achieved a success rate of over 80% in the roundabout scenario. In the study of Folkers et al. [32], the proposed model had a similar success rate of over 80% in the urban scenario. The comparable success rate in this study indicates that the proposed model has learned to drive in the NGSIM environment. The two simulations presented in the results reveal that the model can handle the most common and important daily driving skills like lane keeping and car-following, and that the model preliminarily can merge into the lane.

The proposed model still has a failure rate especially higher in the merging situation. The reason is summarized in two aspects. First, the reward function designed in this study encourages lane keeping; however, lane changing is needed in the process of merging; therefore, a new design of the reward function may be required for a better performance of the merging scenario. Secondly, the PPO agent can only encounter the merging situation when it is initialized in the rightmost lane which has a low probability, the relatively low interaction or experience of the PPO agent in the merging scenario may cause worse performance. Merging is also an open problem in the study of AD [33, 34], and further study should be specifically conducted on this topic.

The present study had some important limitations. The proposed model focused on the scenario of multivehicle interaction on the expressway. The road geometry in the simulator is relatively simple, and no curves or intersections are involved. Future studies should consider these scenarios into account. Also, the proposed model used precise information about the road environment and the surrounding vehicles which may be expensive to get in a real-world situation; image inputs are a promising choice in the future study. Finally, the present study did not consider riding comfort in the design of the reward function. Recent studies emphasized the significance of accomplishing human-like AD models in which riding comfort and human reaction characteristics were considered to improve the acceptability of AVs [35, 36]. The present study should investigate incorporating these factors into the design of AD models.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was jointly supported by the Research and Development Project of Xiangyang City under Grant no. 2022ABH006510, the Hubei Key Laboratory of Power System Design and Test for Electrical Vehicle Opening Fund under Grant no. ZDSYS202217, the Hubei Superior and Distinctive Discipline Group of "New Energy Vehicle and Smart Transportation" under Grant no. XKTD012023, and the Industry-University Cooperation Education Program of the Ministry of Education under Grant no. 220600805293405.

References

[1] Y. Guan, Y. Ren, S. E. Li, Q. Sun, L. Luo, and K. Li, "Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12597–12608, 2020.

[2] J. Wang, L. Zhang, Y. Huang, J. Zhao, and F. Bella, "Safety of autonomous vehicles," *Journal of Advanced Transportation*, vol. 2020, Article ID 8867757, 13 pages, 2020.

[3] O. Sharma, N. C. Sahoo, and N. B. Puhan, "Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: a state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 101, Article ID 104211, 2021.

[4] W. Fang, S. Zhang, H. Huang et al., "Learn to make decision with small data for autonomous driving: deep Gaussian process and feedback control," *Journal of Advanced Transportation*, vol. 2020, Article ID 8495264, 11 pages, 2020.

[5] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826–1848, 2020.

[6] C. Urmson, J. Anhalt, D. Bagnell et al., "Junior: the stanford entry in the urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[7] C. Urmson, J. Anhalt, D. Duggins et al., "Autonomous driving in urban environments: boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[8] P. E. I. Xiao-fei, M. O. Shuo-jie, C. H. E. N. Zhen-fu, and Y. A. N. G. Bo, "Lane changing of autonomous vehicle based on TD3 algorithm in human-machine hybrid driving environment," *China Journal of Highway and Transport*, vol. 34, no. 11, p. 246, 2021.

[9] H. Zhang, J. Xu, and J. Qiu, "An automatic driving control method based on deep deterministic policy gradient," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 7739440, 9 pages, 2022.

[10] P. Cai, S. Wang, Y. Sun, and M. Liu, "Probabilistic end-to-end vehicle navigation in complex dynamic environments with multimodal sensor fusion," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 1–4224, 2020.

[11] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4693–4700, IEEE, Brisbane, Australia, 2018, May.

[12] Y. Pan, C. A. Cheng, K. Saigol et al., "Imitation learning for agile autonomous driving," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 286–302, 2020.

[13] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7262–7269, 2021.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, Cambridge, MA, USA, 2018.

[15] D. Silver, J. Schrittwieser, K. Simonyan et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[16] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2015, https://arxiv.org/abs/1509.02971.

[18] P. Cai, H. Wang, Y. Sun, and M. Liu, "DQ-GAT: towards safe and efficient autonomous driving with deep Q-learning and graph attention networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21102–21112, 2022.

[19] Y. Shi, Y. Liu, Y. Qi, and Q. Han, "A control method with reinforcement learning for urban un-signalized intersection in hybrid traffic environment," *Sensors*, vol. 22, no. 3, p. 779, 2022.

[20] Ngsim, *US Department of Transportation, NGSIM—Next generation simulation*, 2007.

[21] M. Zhou, J. Luo, J. Villella et al., "Smarts: scalable multi-agent reinforcement learning training school for autonomous driving," 2020, https://arxiv.org/abs/2010.09776.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, https://arxiv.org/abs/1707.06347.

[23] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[24] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen, "Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization," in *Proceedings of the 2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 523–533, IEEE, Atlanta, GA, USA, 2019, June.

[25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: an open urban driving simulator," in *Proceedings of the Conference on Robot Learning*, pp. 1–16, PMLR, London, UK, 2017, October.

[26] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "SUMO (Simulation of Urban MObility)-an open-source traffic simulation," in *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM20002)*, pp. 183–187, Sharjah, UAE, September, 2002.

[27] V. Punzo, M. T. Borzacchiello, and B. Ciuffo, "On the assessment of vehicle trajectory data accuracy and application to the Next Generation SIMulation (NGSIM) program data," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1243–1262, 2011.

[28] B. Coifman and L. Li, "A critical evaluation of the Next Generation Simulation (NGSIM) vehicle trajectory dataset," *Transportation Research Part B: Methodological*, vol. 105, pp. 362–377, 2017.

[29] L. Engstrom, A. Ilyas, S. Santurkar et al., "Implementation matters in deep policy gradients: a case study on PPO and TRPO," 2020, https://arxiv.org/abs/2005.12729.

[30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016, https://arxiv.org/abs/1602.01783.

[31] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 2765–2771, IEEE, Auckland, New Zealand, 2019, October.

[32] A. Folkers, M. Rick, and C. Büskens, "Controlling an autonomous vehicle with deep reinforcement learning," in *Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 2025–2031, IEEE, Paris, France, 2019, June.

[33] P. Hang, C. Lv, C. Huang, Y. Xing, and Z. Hu, "Cooperative decision making of connected automated vehicles at multilane merging zone: a coalitional game approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3829–3841, 2022.

[34] B. K. Xiong, R. Jiang, and X. Li, "Managing merging from a CAV lane to a human-driven vehicle lane considering the uncertainty of human driving," *Transportation Research Part C: Emerging Technologies*, vol. 142, Article ID 103775, 2022.

[35] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 97, pp. 348–368, 2018.

[36] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," *Transportation Research Part C: Emerging Technologies*, vol. 117, Article ID 102662, 2020.