Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 21(5): 6077–6096.
DOI: 10.3934/mbe.2024267
Received: 04 December 2023
Revised: 05 May 2024
Accepted: 13 May 2024
Published: 24 May 2024

Research article

Intelligent control of self-driving vehicles based on adaptive sampling supervised actor-critic and human driving experience

Jin Zhang[1], Nan Ma[2,*], Zhixuan Wu[3], Cheng Wang[1] and Yongqiang Yao[4]

[1] Beijing Key Laboratory of Information Service Engineering, Beijing Union University, Beijing 100101, China

[2] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

[3] Beijing University of Posts and Telecommunications, Beijing 100876, China

[4] Beijing Shuncheng High Technology Corporation, Beijing 102206, China

* **Correspondence:** Email: manan123@bjut.edu.cn.

Abstract: Due to the complexity of the driving environment and the dynamics of the behavior of traffic participants, self-driving in dense traffic flow is very challenging. Traditional methods usually rely on predefined rules, which are difficult to adapt to various driving scenarios. Deep reinforcement learning (DRL) shows advantages over rule-based methods in complex self-driving environments, demonstrating the great potential of intelligent decision-making. However, one of the problems of DRL is the inefficiency of exploration; typically, it requires a lot of trial and error to learn the optimal policy, which leads to its slow learning rate and makes it difficult for the agent to learn well-performing decision-making policies in self-driving scenarios. Inspired by the outstanding performance of supervised learning in classification tasks, we propose a self-driving intelligent control method that combines human driving experience and adaptive sampling supervised actor-critic algorithm. Unlike traditional DRL, we modified the learning process of the policy network by combining supervised learning and DRL and adding human driving experience to the learning samples to better guide the self-driving vehicle to learn the optimal policy through human driving experience and real-time human guidance. In addition, in order to make the agent learn more efficiently, we introduced real-time human guidance in its learning process, and an adaptive balanced sampling method was designed for improving the sampling performance. We also designed the reward function in detail for different evaluation indexes such as traffic efficiency, which further guides the agent to learn the self-driving intelligent control policy in a better way. The experimental results show that the method is able to control vehicles in complex traffic environments for self-driving tasks and exhibits better performance than other DRL methods.

Keywords: self-driving; intelligent control; actor-critic; deep reinforcement learning

1. Introduction

With the continuous advancement of artificial intelligence and machine learning, self-driving technology has seen persistent development [1] and has become one of the most popular research areas. The huge potential of artificial intelligence has prompted the research and development of self-driving vehicles to become the focus of attention in various countries. Self-driving vehicles aim to improve passenger safety, enhance driving comfort, optimize vehicle performance [2], and improve the efficiency of time use. According to statistics, 90% of traffic accident injuries and fatalities are caused by driver error behavior. In order to reduce traffic accidents, self-driving vehicles should achieve a sufficient level of intelligence [3, 4] to be able to correctly perceive their surroundings and make safe decisions in complex traffic scenarios and hazardous road conditions [5]. Urban driving [6] is considered one of the most challenging self-driving tasks, because urban environments are highly complex and include dynamic elements such as neighboring vehicles and pedestrians. Given the starting point and destination, the goal of the vehicle is to successfully complete the route in a finite amount of time and satisfy predefined conditions, such as no-collision [7].

Reinforcement learning is a machine learning method centered around the concept that an agent learns to make decisions in order to maximize anticipated rewards through its interactions with the environment [8]. This interactive process involves the agent observing the state of the environment, taking actions, and receiving rewards or punishments based on the results of the environment after the action is performed by the agent. Through much interaction and feedback, the agent gradually learns to choose the best driving policy to achieve the goal of maximizing long-term rewards [9]. Deep reinforcement learning (DRL) represents a fusion of reinforcement learning and deep learning, constituting a specialized subfield within the broader realm of machine learning, allowing the agent to effectively handle high-dimensional state and action spaces [10]. In the field of self-driving, DRL allows self-driving vehicles to make decisions and control their own behavior in complex and changing traffic environments. Compared with traditional rule-based methods [11], DRL can adapt to various urban road scenarios. DRL enables self-driving vehicles to extract knowledge from human driving experiences, which means that self-driving vehicles can learn the operations of human drivers and control themselves to complete self-driving tasks in complex traffic environments. These technologies provide effective methods to deal with self-driving problems [12], which is crucial to ensuring that self-driving vehicles drive safely in complex traffic situations. The decision-making process of DRL helps intelligent vehicles implement self-driving.

Although DRL has great application potential in self-driving, it still has some defects and shortcomings. In the self-driving scenario, the agent interacts with an environment in which both its state space and action space are continuous; this will lead to a slow learning rate of the agent, difficult in convergence, and challenges in learning a well-performing decision-making policy. DRL in self-driving has a large range of action space (e.g., the steering wheel angle of the vehicle is between -540 and $+540$). However, most of the time, this angle is within a very small interval (e.g., between -10 and $+10$), and only very rarely is a large angle needed. Consequently, at the beginning of the learning process, the agent is most likely to choose a very large steering wheel angle (resulting in a collision or driving off the road), causing the learning process to crash right from the outset. Inspired by supervised learning [13], in this paper, we propose a new self-driving vehicle control method that combines human driving experience and adaptive sampling supervised

actor-critic (ASS-AC). We used supervised learning to pre-train the policy network so that it learns the driving skills of a human driver to a certain extent, preventing it from making more serious mistakes at the beginning of reinforcement learning, and then improving and adjusting its own driving decision-making policy through reinforcement learning. During the reinforcement learning process, we also used a human driver to guide the agent in real time. At the same time, in order to speed up the learning rate, we designed an adaptive experience sampling method to enable the ego vehicle to automatically adjust the proportion of human driving experience in a batch sample at different learning stages. We also noticed that the reward function plays a key role in the learning process, so we designed a specificity reward function for the self-driving task and guided the ego vehicle to learn faster through a more refined reward function design so that it could get a better self-driving control policy as much as possible. Contributions of this work are listed below:

- A real-time human-guided ASS-AC algorithmic framework is proposed and applied to the control of autonomous vehicles;
- An adaptive experience sampling method is proposed, which automatically adjusts the proportion of self-discovery experience of autonomous vehicles and human driving experience sampled in the samples learned by the model;
- The proposed method is compared with several other DRL methods in CARLA simulator and shows better performance than others.

2. Related works

Since self-driving was proposed, some researchers have been conducting research on it with great enthusiasm. In the early days, self-driving decision-making was dominated by the traditional rule-based method [14], but this relied heavily on manually formulated rules [15]; since it is not possible to predict all traffic scenarios, this method is not the most reliable. With the development of machine learning, methods based on learning [16] provided new research directions for self-driving decision-making. DRL perfectly combines the representation ability of deep learning with the trial-and-error mechanism of reinforcement learning [17]. It obtains better expected rewards through continuous training and improvement of the agent's policy. Self-driving based on DRL can use sensory inputs such as millimeter-wave radar, lidar, and cameras to directly obtain control actions such as throttle, brake, and steering wheel angle [18], which greatly reduces the workload and parameter adjustment costs of each layer of algorithm construction. At the same time, it gets rid of the dependence on complicated rule formulation work and improves the generalization ability of self-driving [19, 20].

2.1. Rule-based self-driving

Traditional self-driving mainly uses a layered framework that combines perception, decision-making, and control [21], in which the decision-making and control modules are usually designed using a rule-based approach. The finite state machine (FSM) is widely used in behavioral decision-making due to its simplicity and practicality. Jo et al. [22] used FSM to select driving strategies among predefined rules to meet traffic rules, such as speed limits, traffic light signals, and the prohibition of lane changes. Okumura et al. [23] combined FSM and support vector machines (SVM) to construct a behavioral decision maker suitable for the island roundabout scenario. Guidolini et al. [24] used FSM to solve the decision-making problem in zebra crossing scenarios

based on input information such as maps, vehicle status, current path, pedestrians on zebra crossings, and traffic signals.

Methods based on trajectory planning are also used in self-driving vehicles [25]. The vehicle Odin [26] in the DARPA Challenge uses the Dijkstra algorithm for trajectory planning and completes self-driving through trajectory tracking control. Kala and Warwick [27] applied it to self-driving vehicles and simulated a multi-vehicle environment. As an extension of the Dijkstra algorithm, the A* algorithm uses the heuristic cost estimation method to allocate weights to each node, thereby improving the planning speed. Li et al. [28] used cubic polynomial curve alternative paths to construct the state grid and used a cost function to evaluate and select the generated trajectory. The interpolation curve method uses an interpolation function to calculate a smooth desired trajectory based on the pre-planned trajectory points and vehicle status set. Commonly used interpolation curves include polynomial curves, Bezier curves [29], etc. Methods based on numerical optimization achieve trajectory planning by solving a constrained objective function. The solution process usually also needs to satisfy vehicle dynamics constraints, safety constraints, comfort constraints, and other conditions.

2.2. Self-driving based on DRL

Compared with rule-based methods, the representation and generalization abilities of DRL [30] have shown great potential in self-driving. Sallab et al. [31] first applied DRL to the research of lane keeping, using the distance between the vehicle and the lane center and the vehicle speed as state inputs to output steering angle, gear level, and the values of acceleration or braking. In the simulation environment TORCS, training and testing were conducted. The experiments showed that the Deep Q-Network (DQN) and the Deep Deterministic Policy Gradient (DDPG) performed equally well in the straight road scene, but in the curved driveway, the DDPG, which outputs continuous actions, showed more stable control effect and better comfort. Chae et al. [32] used the method of separating sparse collision samples and ordinary samples to train the DQN, which improved the safety performance of the DQN in autonomous obstacle avoidance tasks in zebra crossing scenes. Zhu et al. [33] used the time to collision (TTC), headway time, and acceleration to characterize safety, efficiency, and comfort, and combined the above factors with the reward function of the DDPG. The experimental results showed that the proposed method is consistent with the real vehicle driving performance recorded in the NGSIM data set, having advantages in both safety and efficiency. Jaritz et al. [34] used the Asynchronous Advantage Actor-Critic (A3C) algorithm in the rally game to make the vehicle learn drifting and other actions, and achieved similarly good results when using real images similar to the training scene as input to the algorithm. Qian et al. [35] inputed the path features extracted from the path planning space into the Twin Delayed Deep Deterministic (TD3) algorithm, allowing the actor network to learn better policies from safe and feasible candidate paths, improving the performance of vehicle motion control and path planning. Ure et al. [36] successfully introduced DRL to tune parameters in the model predictive control (MPC) controller to obtain better and more stable performance in path tracking.

DRL requires constant interaction and feedback from the environment, and learning optimal behavior policy under the guidance of reward functions. Factors such as low sample efficiency and unreasonable reward function design will affect the learning rate of the agent and even make it impossible to learn a policy that performs well. In this paper, we combine the DRL algorithm

actor-critic with supervised learning, introduce human driving experience in the learning samples, adaptively adjust the sampling ratio of self-exploration experience and human driving experience in the learning samples at different stages, and at the same time improve its decision-making results by real-time human guidance during the training process. Reward functions were designed in detail for different metrics used to compute rewards when the ego vehicle interacts with the environment. We experimentally validated the proposed method in a simulation environment with adversarial situations to evaluate its decision-making performance and capability when facing rare and challenging scenarios [37].

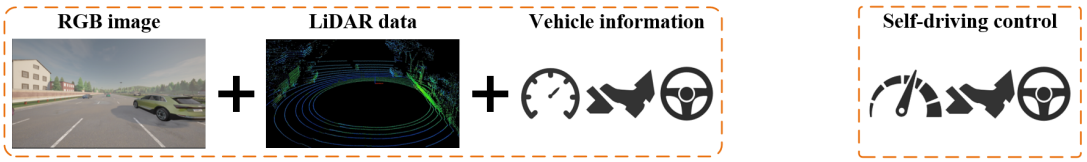
3. Methods

In the reinforcement learning process, the agent observes the environment state s_t at moment t, selects and executes the action a_t according to the policy $\pi(a_t|s_t)$, and after the action a_t is executed, the environment transitions to the next state s_{t+1} and returns the corresponding reward r_t. In this section, we will give a detailed description of our proposed method. First are the inputs s_t and outputs a_t of the model; next, the formulation of the reward function is introduced, then the collection and use of human driving experience and pre-training of the decision-making network using supervised learning, and finally, the general framework of the proposed method and the reinforcement learning process are presented.

3.1. Inputs and outputs

In the end-to-end self-driving system, it is crucial to accurately understand environmental information. In this work, we take the state of the environment and the state of the vehicle itself together as the state inputs to the model, and the outputs as the driving actions that the self-driving vehicle should perform, i.e., how to control the accelerator/brake and steering wheel of the vehicle itself. Specifically, we utilize three different types of data to jointly characterize the input state s_t: RGB images captured by the vehicle's front-facing camera, 16-line LiDAR point cloud data, and the vehicle's own state information. With this multimodal input, our system is able to capture more comprehensive information about the surrounding environment and thus make more accurate driving decisions. First, RGB images provide rich visual information, which are crucial for understanding complex traffic scenarios. However, relying only on image data may lead to inaccurate or missing information under changing lighting or obstructed vision. To overcome this limitation, we introduce 16-line LiDAR point cloud data. LiDAR provides accurate distance measurements, generates a 3D representation of the surrounding environment, and is unaffected by lighting conditions, thus enabling it to work effectively at night or in other low-light conditions. Inputs from the vehicle's own information, such as velocity and acceleration, are important for understanding the current state and predicting future states of the vehicle. This information helps the algorithm to evaluate the possible consequences of different maneuvers and thus make faster and safer decisions at high speeds or in emergency situations. Unet and PiontNet are used to process RGB images and point cloud data, respectively. We assume that the inputs to these data correctly represent information about the surroundings and that the Unet network correctly semantically segments the RGB images. The driving actions a_t output from the model are the actions that the vehicle should perform, specifically the steering wheel angle values and the throttle or brake values. The inputs (states) and

outputs (actions) of the model are shown in Figure 1.



(a) Input (state). **(b)** Output (action).

Figure 1. Input and output representations of the model.

3.2. Reward function

The reward function used to compute the reward r_t is one of the most important factors influencing learning in reinforcement learning. A reasonable reward function can speed up the learning process and show great performance. We comprehensively considered various related factors in self-driving, fully considered the driving safety, driving efficiency, and comfort of self-driving vehicles, and designed the following reward function:

$$r_t = r_{speed} + r_{collision} + r_{mindis} + r_{acc} + r_{close} \tag{3.1}$$

The first item r_{speed} is used to evaluate the transportation efficiency. We hope that the vehicle can go fast, but at the same time, faster may not be better, therefore:

$$r_{speed} = \begin{cases} min(v_{car}, 10) - 5, & v_{car} \leq 10 \\ 10 - v_{car}, & v_{car} > 10 \end{cases} \tag{3.2}$$

v_{car} represents the driving speed of the vehicle, whose unit is m/s. The reward is negative when the vehicle is traveling at less than 5 m/s, and a positive reward is only given when the vehicle speed is greater than 5 m/s. Also for safety, the vehicle speed should not be too fast: If the vehicle speed is higher than 10 m/s, the reward is also negative. The second and third items $r_{collision}$ and r_{mindis} represent whether the vehicle collides and the minimum distance from surrounding vehicles, respectively:

$$r_{collision} = \begin{cases} 1, & no\text{-}collision \\ -1000, & collision \end{cases} \tag{3.3}$$

$$r_{mindis} = d_{min} - 5 \tag{3.4}$$

That is, when the vehicle collides while driving, the value of $r_{collision}$ is -1000, and this training episode will end. It is a relatively serious penalty item, which means that the vehicle should avoid collisions as much as possible. The value of $r_{collision}$ is 1 when no collision occurs. The r_{mindis} is for the distance reward, and d_{min} represents the minimum distance between the center of the ego vehicle and other surrounding vehicles. When the minimum distance is less than 5 m, a corresponding penalty will be given, which means that the ego vehicle should try its best to stay away from other vehicles to avoid a collision. r_{acc} is a measure of comfort:

$$r_{acc} = min(0, 2 - |acc|) \tag{3.5}$$

The *acc* is the acceleration of the ego vehicle. When the absolute value of the acceleration of the ego vehicle is greater than 2 m/s^2, a penalty will be received. Otherwise, this value is 0, which means we hope that the speed of the vehicle should not change too drastically. The last item r_{closse} is used to measure whether the self-vehicle is approaching the destination, which can be expressed as:

$$r_{closse} = distance_t - distance_{t-1} \tag{3.6}$$

$distance_t$ and $distance_{t-1}$ represent the distance of the ego vehicle from the destination in meters at the current and previous moments, respectively; the reward is positive when the ego vehicle is moving closer to the destination and negative when the vehicle is moving away from the destination. With this setting, we encourage the ego vehicle to keep moving closer to the target position to complete the self-driving task.

3.3. Human driving experience and the pre-training of the decision-making network

We noticed that the direct use of AC algorithms in self-driving tasks suffers from inefficient exploration, and it takes a long time for self-driving vehicles to learn seemingly feasible decision-making policy, with some even failing to do so. Therefore, we improved it accordingly by introducing human driving experience to assist the algorithmic model to learn better. Human driving experience has two purposes. Firstly, it is used for the pre-training of the decision-making network *actor*, which is pre-trained using the human driving experience, so as to make it learn some basic driving skills and to prevent it from committing more serious mistakes in the beginning of the reinforcement learning, which will lead to the end of this round of training. Secondly, during the reinforcement learning training phase, which is used to learn and keep up to date with the *critic*, the *critic* is able to learn more quickly which maneuvers are good and which are not by using a combination of human driving experience. Because human driving experience provides high-quality, successful action sequences, these action sequences are used as references during the training process to help the *critic* more accurately assess the value of actions. By utilizing human driving experience, the *critic* can quickly learn the value of effective actions, which in turn guides the *actor* to more targeted exploration. The human driving experience is obtained by the human driver controlling the vehicle in the environment of the simulator, and it is denoted by tuple (s_t, a_t, r_t, s_{t+1}). After completing the collection of human driving experience, the *actor* is pre-trained by supervised learning with s_t as training data and a_t as labeling. The input s_t consists of three parts: RGB images, point cloud data, and the vehicle's own information, such as speed. Unet and PiontNet are used to process the RGB image and point cloud data, respectively, while the vehicle information is represented by one-dimensional vectors. After extracting the features of the three, they are fused, and the fused features are used as inputs to the *actor* and output the control information of the vehicle, such as steering wheel angle and the value of throttle or brake a_t, as shown in Figure 2.

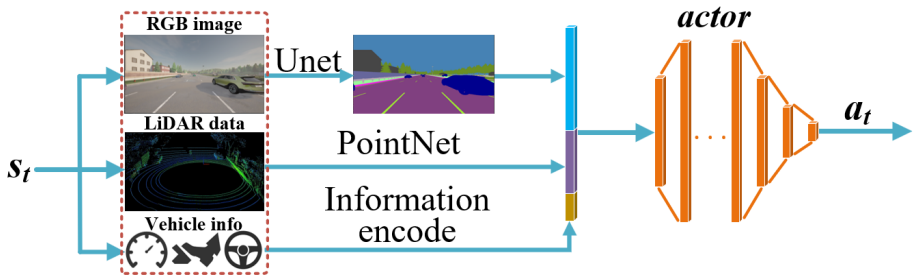


Figure 2. The inputs and outputs of actor.

3.4. ASS-AC framework

Figure 3 shows the structure of the ASS-AC framework. We maintain an *actor* network and two *critic* networks $critic_{target}$ and $critic_{value}$. We use the fixed $critic_{target}$ method, that is, we let the $critic_{target}$ equalize $critic_{value}$ every time a fixed number of learning steps are completed so as to ensure the stability of the learning. An experience pool D [38] storing experience samples is also maintained for storing self-exploration experiences of self-driving vehicles and human driving experiences. The *actor* is responsible for outputting action a_t based on the state s_t, the $critic_{value}$ guiding *actor* to update, and continuously optimizing its own policy.

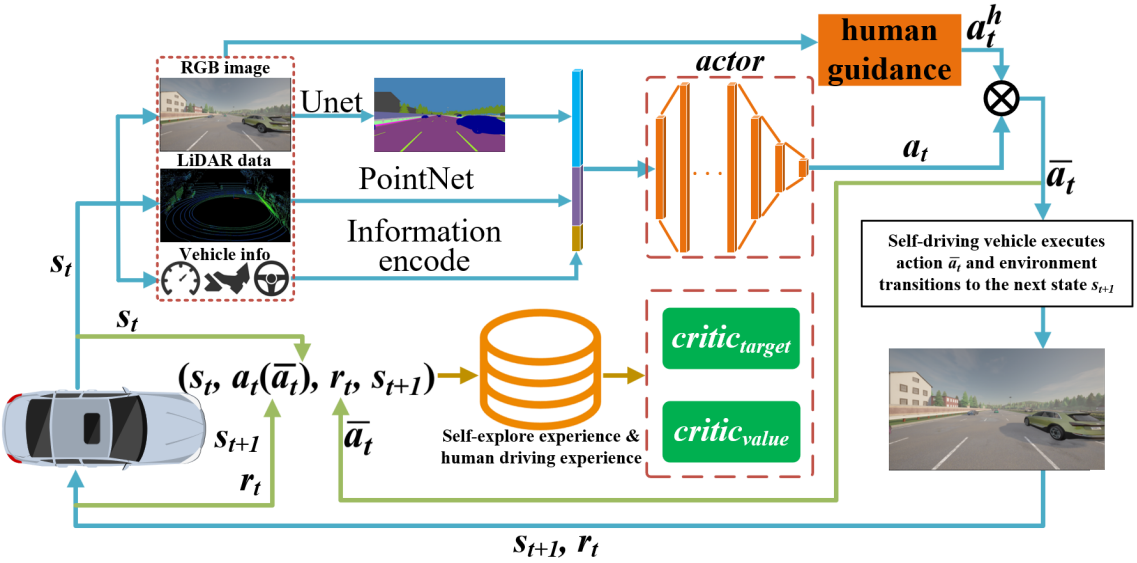


Figure 3. ASS-AC framework.

The framework shows the process in which the ego vehicle continuously interacts with the environment and learns and optimizes its behavior policy $\pi(a_t|s_t)$. The ego vehicle observes the state of the environment s_t at moment t, which is represented by three parts: the RGB image captured by the vehicle's front camera, the 16-line LiDAR point cloud data, and the vehicle's own state information. For the RGB image, we are more concerned about the relative position of obstacles in the field of view than their color information; therefore, we use Unet to semantically segment the original image. The purpose of this is to enable key features in the image (such as the locations of the

surrounding vehicles) to be better extracted while ignoring other useless information. Then, the semantic segmentation image is subjected to feature extraction, and the point cloud data is subjected to feature extraction using PointNet network. The vehicle's own state information includes seven items: steering wheel angle, accelerator/brake value, vehicle speed, acceleration, steering angle speed, steering angle acceleration, and distance from the destination, which are represented by one-dimensional vectors $[n_1, n_2, n_3, n_4, n_5, n_6, n_7]$, and then encoded with features. Then, the features of image, point cloud, and vehicle state are fused, the fused features are used as inputs to the *actor*, and the action a_t is output according to the policy $\pi(a_t|s_t)$. The action a_t is represented by vector $[a_{t_1}, a_{t_2}]$. The value range of each component is -1 to 1. a_{t_1} represents the steering wheel angle of the vehicle: If it is less than 0, it means turning left, while a value greater than 0 means turning right. a_{t_2} indicates throttle and brake: A value lower than 0 means braking, and greater than 0 means throttle.

In the training phase of the algorithm, in order to better learn the decision-making policy, we introduce real-time human guidance. When the *actor* outputs the action a_t, it does not directly execute the action. As the ego vehicle interacts with its environment, the human driver also controls the ego vehicle in real time based on the state s_t. The human driver's control action for the vehicle is a_t^h, the $\bar{a}_t$ is obtained by weighting and summing a_t, a_t^h is the final action to be executed, and the weight of the two is related to the reward value of the previous moment r_{t-1}:

$$\bar{a}_t = \begin{cases} \frac{r_{t-1}}{\bar{r^H}} \times a_t + (1 - \frac{r_{t-1}}{\bar{r^H}}) \times a_t^h, & \frac{r_{t-1}}{\bar{r^H}} < 0.5 \\ a_t, & \frac{r_{t-1}}{\bar{r^H}} \geq 0.5 \end{cases} \tag{3.7}$$

$\bar{r_H}$ is the average reward value of human driving experience, i.e., the mean value of all r_t in the collected human driving experience (s_t, a_t, r_t, s_{t+1}). $\frac{r_{t-1}}{\bar{r_H}}$ has a minimum value of 0 and a maximum value of 1. If the ego vehicle performs better in the previous moment, we believe that its decision can bring better rewards, and the weight of its output a_t will be increased accordingly in the next moment. When its performance is poor in the previous moment, we consider that its decision cannot bring better rewards, and the next moment will need real-time guidance of the human driver to improve its policy, which reduces the weight of a_t and increases the weight of a_t^h. When $\bar{a}_t$ is executed, the environment transitions to the next state s_{t+1} and returns the reward r_t. After the ego vehicle completes one interaction with the environment, one self-exploratory experience $(s_t, \bar{a}_t, r_t, s_{t+1})$ will be stored in the experience pool D just like the human driving experience (s_t, a_t, r_t, s_{t+1}) and will be used for learning and update the parameters of the $critic_{value}$ network. For the sampling process, we designed a new adaptive sampling method for automatically adjusting the proportion of self-exploration experience and human driving experience in each batch of learning samples. The sampling learning is performed in the training phase by the following equation:

$$B = (p \sim D^H) \cup (1 - p \sim D^H) \tag{3.8}$$

B is a batch of learning samples, $p \in [0, 1]$ represents the proportion of human driving experience D^E in this batch of learning samples, and D^H is the self-exploration experience of the ego vehicle, where:

$$p = 1 - \frac{\overline{r^E}}{\overline{r^H}} \tag{3.9}$$

r_t is the current reward of the ego vehicle when exploring, the minimum value of p is 0, and the maximum value is 1. Therefore, when the average reward of the ego vehicle is low, it will increase the sampling proportion of human driving experience and learn with high-quality human driving experience; when the average reward of the ego vehicle has some improvement, the percentage of self-exploration experience in its learning samples will be increased accordingly. This means that ego vehicles are able to intelligently decide when to use self-exploration experiences and when to rely on human driving experiences, thus improving the learning efficiency.

3.5. Loss function

In order to overcome the problem of inefficient exploration and to prevent the model from making more serious mistakes at the beginning of reinforcement learning that could lead to the end of training, we pre-trained the *actor* network using collected human driving experiences and by means of supervised learning. The *actor* performs parameter updates through the following loss function in the pre-training stage:

$$L_{actor,pre}(\theta) = \frac{1}{N}\sum_{i=1}^{N}|a_{\theta i} - a_i|^2 \tag{3.10}$$

$i = 1, 2$, θ is the parameter of *acto*, $a_{\theta i}$ is the corresponding component of the action output by the *actor*, and a_i represents the corresponding component of the action in human driving experience.

As shown in Figure 1, $critic_{value}$ is used to evaluate the quality of the driving action generated by the *actor*, that is, to evaluate the behavioral policy of the *actor*. It should guide the *actor* to select actions that can bring more rewards. Its loss function is as follows:

$$L_{critic_{value}}(\phi) = E_{(s_t,a_t(\bar{a}_t),r_t,s_{t+1})}[(Q(s_t, a_t(\bar{a}_t); \phi) - (r_t + \gamma Q(s_{t+1}, \mu(s_{t+1}; \theta'); \phi')))^2] \tag{3.11}$$

$a_t(\bar{a}_t)$ denotes that the data used for $critic_{value}$ learning consists of human driving experience (s_t, a_t, r_t, s_{t+1}) and ego vehicle exploratory experience $(s_t, \bar{a}_t, r_t, s_{t+1})$, ϕ and ϕ' are the parameters of $critic_{value}$ and $critic_{target}$ parameters, r_t is the current reward, $\mu(s_{t+1}; \theta')$ is the action in the next state s_{t+1} determined by *actor*, and $Q(s_{t+1}, \mu(s_{t+1}; \theta'); \phi')$ is the estimation of value of $critic_{target}$ for the next state and the next action. $Q(s_t, a_t(\bar{a}_t); \phi)$ is the value estimate of $critic_{value}$ for the current state and action.

In the exploration stage, we hope that the *actor* can generate actions with greater rewards; its loss function is usually defined by using $critic_{target}$ to evaluate the value of the action under the current policy. In the case of deterministic policy, the loss function of *actor* can be designed to maximize the estimate of the value of the actions generated by the current *actor*. Thus, the loss function of *actor* can be the negative of the value of the action generated by the *actor* as evaluated by the $critic_{target}$, so:

$$L_{actor}(\theta) = -E[Q(s_t, \mu(s_t; \theta); \phi)] \tag{3.12}$$

$\mu(s_t; \theta)$ is the action a_t under the $actor_{network}$ determining state s_t, and $Q(s_t, \mu(s_t; \theta); \phi)$ is the value of the action under the given state and action. In order to better illustrate the proposed methodology, the ASS-AC pseudocode is as described in Algorithm 1.

Algorithm 1 Adaptive sampling supervised actor-critic

Initialize the experience pool D, collect human driving experience and action labels, and store them in the experience pool D, initialize the *actor* θ .

Using supervised learning to pre-train the *actor* θ.

Initialize $critic_{value}$ ϕ , $critic_{target}$ ϕ', discount factor γ.

for episodes

initial the ego-vehicle at starting point

while true:

Observation state s_t and generation a_t according to $\pi_\theta(.|s_t)$

Compute $\bar{a}_t$ using Eq (3.7)

Execute $\bar{a}_t$ in the environment

Return the next state s_{t+1} and reward r_t

if s_{t+1} is the terminal

break

Store transition (s_t, a_t, r_t, s_{t+1}) in the experience pool D

Sample a batch data B from D using Eq (3.8)

Update the $critic_{value}$ ϕ using Eq (3.11)

Update the *actor* θ using Eq (3.12)

If have study fixed steps

update $critic_{target}$ ϕ' using $\phi' = \phi$

Save the best *actor* θ

End for

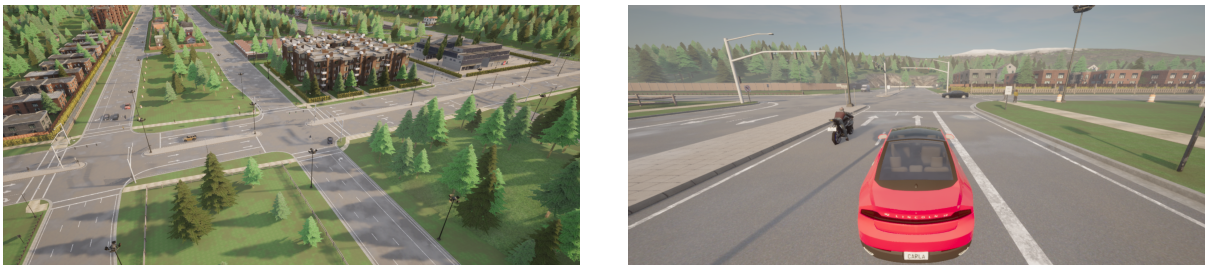
4. Experiment

4.1. Simulation setting

We experimentally validated our proposed method in the CARLA simulator and chose Town06 as our training scenario. We randomly initialized 40 traffic participants in the scene, including vehicles, pedestrians, etc.; the scene is shown in Figure 4(a). In the scenario, we randomly selected two far apart locations as the start and destination of the ego vehicle (the distance between the start location and the destination is the same each time we initialize), and initialize the ego vehicle in the start location; the ego vehicle needs to complete the decision-making according to our proposed method to control itself from the start location to the destination, to complete the self-driving task. The red vehicle in Figure 4(b) is the ego vehicle. The train episode ends when one of the following situations occurs: 1) The ego vehicle collides with another vehicle or runs off the road; 2) the ego vehicle completes the task and reaches the end point; or 3) the simulation step length is greater than 2000 steps.

In order to make our method effective in dealing with adversarial scenarios, during each training session, we set up unexpected situations, these make it challenging for the self-driving vehicle to

verify the safety of the must-go road between the starting point and the destination with a probability of 25%. Specifically, as shown in Figure 5, we considered the following three adversarial scenarios: 1) pedestrians rushing out from the blind spot of the self-driving vehicle's field of vision; 2) a vehicle in front suddenly changing lanes and occupying the ego vehicle lane; and 3) traffic congestion situations. We randomly choose an adversarial situation with 25% probability so that it occurs on the mandatory route between the starting point and the destination in order to make the ego vehicle learn how to deal with these rare and challenging contingencies.



(a) Town06 in CARLA. **(b)** The red ego vehicle.

Figure 4. Training environment of CARLA.

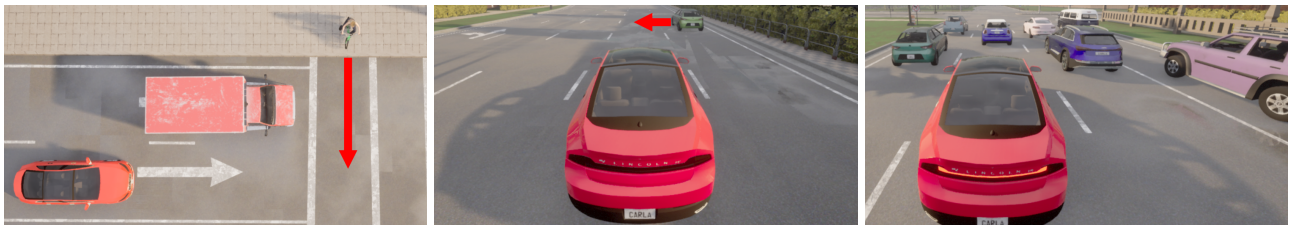


Figure 5. Adversarial scenarios.

4.2. Training settings

We trained the ASS-AC network in 50000 episodes in the simulation environment, with a maximum step size of 2000 for each episode. The neural network was trained on two NVIDIA RTX 3080Ti GPU using Pytorch and Adam optimizers with a learning rate of 1×10^{-4}, with a total of more than 150 million training parameters. After each training episode, the *actor* with the highest reward is saved for subsequent testing. For the ablation experiments, we used the AC algorithm as a baseline model and verified the effect of different components on the algorithm's performance improvement, i.e., whether or not to pre-train the *actor* network, whether or not to allow a human driver to provide real-time guidance, whether or not to incorporate the human driving experience in the experience pool, and the performance of including all three components at the same time. In order to conduct a comprehensive evaluation of the performance of the proposed method, we compared it with other existing methods, including DQN, Policy Gradient (PG), Proximal Policy Optimization(PPO), A3C, and DDPG.

4.3. Experiment results

For the ablation experiments, after completing the training, we saved the set of parameters with the highest average reward for each. Table 1 shows the detailed results of the ablation tests. The actor-critic algorithm is a reinforcement learning method that combines the learning of policies and value functions to achieve faster learning and better policy performance. This approach has been shown to be effective in numerous domains, but its direct application in complex self-driving environments may face challenges such as low sample efficiency and difficulty in learning strategies that meet safety requirements. The pre-training can help the model to have a more reasonable strategy than random in the initial stage, thus accelerating the subsequent learning process. In the reinforcement learning stage, adding human driving experience to the learning process in the form of samples and dynamically adjusting the ratio of human experience and self-exploration samples can further improve the learning efficiency and safety of the strategy. This approach utilizes human intuition and experience to reduce dangerous situations that the model may encounter during the exploration process, while accelerating the learning of effective strategies. By incorporating real-time human guidance in the execution phase, i.e., weighted summation of the decision network's output and the human driver's actions, the model's behavior can be corrected in real time to ensure safety, while also providing immediate feedback to the model, which helps it adapt to complex driving environments and unknown situations more quickly. Ultimately, the combination of these three components was used to achieve the greatest performance gain, demonstrating that human knowledge and experience can be a significant aid to reinforcement learning models for complex tasks, especially in the field of autonomous driving where a high degree of safety and reliability is required. This integrated approach takes full advantage of human intuition and experience to quickly reach a better starting point through pre-training, accelerates the learning process by dynamically incorporating human experience samples, and corrects model deficiencies through real-time coaching, demonstrating an effective collaborative human-machine learning framework.

Table 1. Ablation test results in CARLA Town06.

Method	**SR(%)**	**RCR(%)**	**CR(%)**	**AR**	**AS**
AC(baseline)	65	73.53	18	2074.5	1803
AC+Pre-train *actor*	89	79.76	14	2407.8	1678
AC+Human driving experience	73	78.50	15	2583.5	1698
AC+ Real-time human guidance	88	87.33	7	3138.6	1380
ASS-AC(**Ours**)	**93**	**95.72**	**2**	**3345.3**	**1226**

SR represents the success rate, meaning that the ego vehicle successfully reaches the destination, RCR represents the average route completion rate. AR is the average reward (the higher the better). CR is the collision rate, and AS represents the average steps length of the ego vehicle successfully driving from the start point to the destination (the lower the better).

Figure 6 shows the comparison of training results between our method and other methods. From the figure, we can find that the proposed method has better performance and faster convergence speed. The reward of the proposed method increases steadily with the increase in training episodes, and begins to converge at about 30,000 episodes. The biggest advantage of the proposed method is that it combines the advantages of supervised learning and reinforcement learning, which quickly improves

the starting performance of the model through the pre-training of human driving experience, and the dynamic incorporation of human driving experience and real-time guidance further improves the adaptability and safety of the model in complex situations. This method is able to react more flexibly and accurately when dealing with unexpected situations because it has incorporated a large amount of human intuition and judgment during model training. DQN is able to deal with high-dimensional observation spaces by combining deep learning and Q-learning, but may not be as flexible as policy-based approaches when dealing with continuous action spaces and tasks that require long-term policy considerations. In complex scenarios of autonomous driving, especially in unexpected situations that require fast reactions, DQN may struggle to make optimal decisions due to their over-reliance on the estimation of value functions. The PG method directly optimizes the policy function, which can better handle the continuous action space and is suitable for application scenarios such as autonomous driving. However, PG methods usually face higher variance and slower convergence, and may not be able to learn adaptive policies quickly in unexpected situations. A3C is able to accelerate the learning process and reduce the training time by executing and learning in parallel with multiple worker threads. It combines the advantages of PG and value function to improve stability and efficiency. Nevertheless, in complex autonomous driving scenarios, A3C may still face challenges from high uncertainty and dynamic changes in the environment. PPO improves the stability of learning by optimizing a specific objective function to avoid making excessive paces when updating the policy. This approach performs well in many complex environments. However, it may not be as flexible as methods that combine human intuition and real-time feedback in very dynamic situations such as contingency processing. DDPG is an algorithm that combines PG and Q-learning for continuous action spaces and improves the stability of learning through the use of an objective Q-network and a policy network. Although DDPG performs well in many tasks, it may not perform as well as methods that incorporate human experience when dealing with complex, highly uncertain environments. The proposed method achieves the best performance in challenging test scenarios of autonomous driving that include pedestrians rushing out of blind spots in the field of view, sudden lane changes of the vehicle in front, and traffic congestion situations. This is mainly attributed to the method's ability to leverage human experience and intuition to provide immediate adaptation and decision correction, thus maintaining high safety and effectiveness in complex and unexpected driving situations. In contrast, other methods, while having their own advantages, may lack sufficient flexibility and adaptability when dealing with such highly dynamic and uncertain environments.

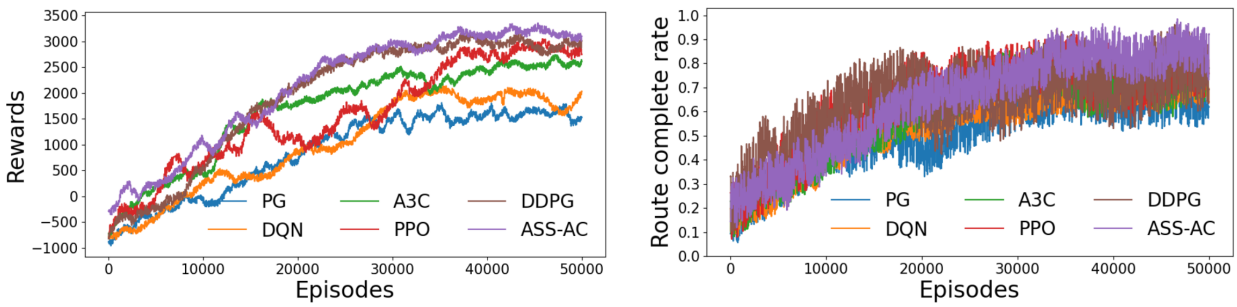
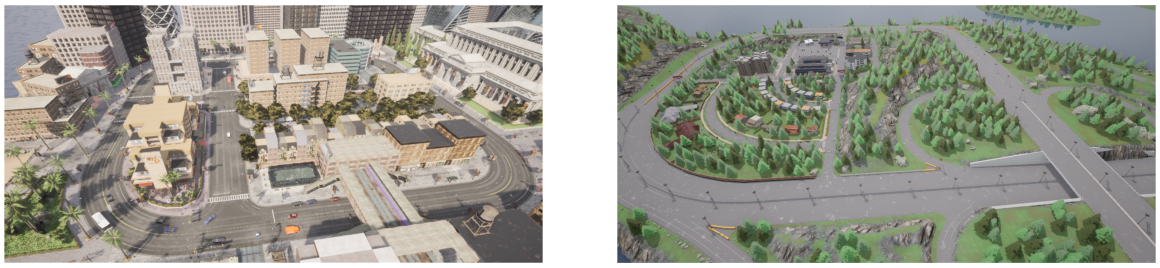


Figure 6. Training results in CARLA Town06.

We tested our method and others 100 times in the training scenario Town06; the test results are shown in Table 2. Although our proposed method does not perform the best in RCR, its overall performance is better than other methods. We also tested the trained algorithm in new scenarios (as shown in Figure 7): the new test scenarios are Town10, a complex urban traffic scenario, and Town04.

Table 2. Test results in CARLA Town06.

Method	SR(%)	RCR(%)	CR(%)	AR	AS
DQN	65	74.53	16	2133.6	1684
PG	57	77.80	22	1805.4	1770
A3C	80	86.35	13	2735.5	1642
PPO	87	93.68	9	3044.9	1479
DDPG	90	**96.03**	5	3145.0	1337
ASS-AC(**Ours**)	**93**	95.72	**2**	**3345.3**	**1226**



(a) Town10. **(b)** Town04.

Figure 7. New test scenarios.

As we can see from Tables 3 and 4, in the urban environment of Town10, although the performance of our method somewhat decreases, such decrease is not significant, opposed to the performance of the other methods that decreases significantly. The main reason for this decrease in performance is the higher complexity of these scenarios compared with the training scenarios. The difference between Town04 and the training scenario Town06 is not as large as the difference between Town10 and the Town06, so the degradation in performance in Town04 is not as obvious as the degradation in Town10. This shows that our proposed method can also show relatively good performance in new scenarios.

Table 3. Test results in CARLA Town10.

Method	SR(%)	RCR(%)	CR(%)	AR	AS
DQN	49	65.63	19	1863.5	1780
PG	50	68.53	23	1633.8	1875
A3C	73	78.66	14	2452.6	1744
PPO	81	85.28	10	2633.0	1680
DDPG	88	88.68	7	3055.3	1456
ASS-AC(**Ours**)	**90**	**93.23**	**4**	**3205.1**	**1305**

Table 4. Test results in CARLA Town04.

Method	SR(%)	RCR(%)	CR(%)	AR	AS
DQN	61	70.48	14	2058.9	1659
PG	55	79.55	22	1904.3	1780
A3C	73	85.64	15	2677.8	1637
PPO	85	92.48	8	2770.5	1503
DDPG	89	89.59	6	**3088.4**	1409
ASS-AC(**Ours**)	**92**	**92.04**	**2**	3063.8	**1238**

5. Conclusions

In this paper, we proposed an improved actor-critic ASS-AC algorithm and applied it to self-driving vehicle control. In order to reduce the agent's ineffective exploration and make its driving behavior more humane-like, we used supervised learning methods to pre-train the policy network $actor_{network}$, allowing the vehicle to learn some human driving skills. At the same time, in order to improve the accuracy of decision-making during the learning process, we weighted and summed the output of human and ego vehicle decision-making results by introducing real-time human guidance. We designed an adaptive experience sampling method to enable the agent to automatically adjust the proportion of human driving experience in a batch sample; the actor network is updated by combining human driving experience and maximizing rewards to accelerate the learning rate. We verified our proposed method in a simulation environment with dense traffic flow and compared it with other methods. The results show that our proposed method can effectively control the self-driving vehicle to complete self-driving tasks in dense traffic flow and perform well, and showing better performance than other methods.

As far as real-time computational efficiency is concerned, our method's computational requirements on a device with two NVIDIA RTX 3080TIi GPUs is 20 25Hz, which basically meets the driving requirements when the self-vehicle is traveling at a low speed (less than 30 km/h). However, when with the acceleration of the vehicle's traveling speed, the algorithm's performance will be degraded. Also, there are still some challenges when implementing and deploying the method in real scenarios; the real world is more complex than the simulation environment and there are more emergent situations, which may bring new challenges to the algorithm performance [39]. Finally, while self-driving systems can better adapt to unknown environments or rare events through real-time human guidance, and human experience can help the system to cross situations that are not covered by insufficient data or algorithms, implementing real-time human guidance in a wide range of autonomous driving applications, especially when deployed at scale, requires a large number of human resources to provide guidance, which is potentially costly in terms of labor. Also, the degree of reliance on human decision-making may limit the ability of the self-driving system to learn and self-optimize, reducing its ability to operate independently in the absence of human guidance. How to solve these problems and find new solutions will be the focus of subsequent research.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the National Key R&D Program of China (No.2023YFF0615800), the National Natural Science Foundation of China (No. 62371013, 61931012), the Beijing Natural Science Foundation (No. 4222025) and QIYUAN LAB Innovation Fundation (Innovation Research) Project (No. S20210201107).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, et al., Deep reinforcement learning for autonomous driving: A survey, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 4909–4926. https://doi.org/10.1109/TITS.2021.3054625
2. J. Chen, B. Yuan, M. Tomizuka, Model-free deep reinforcement learning for urban autonomous driving, in *2019 IEEE intelligent transportation systems conference (ITSC)*, (2019), 2765–2771. https://doi.org/10.1109/ITSC.2019.8917306
3. M. Panzer, B. Bender, Deep reinforcement learning in production systems: a systematic literature review, *Int. J. Prod. Res.*, **60** (2022), 4316–4341. https://doi.org/10.1080/00207543.2021.1973138
4. N. Ma, Y. Gao, J. Li, D. Li, Interactive cognition in self-driving, *Sci. Sin. Inf.*, **48** (2018), 1083–1096. https://doi.org/10.1360/N112018-00028
5. H. Shi, D. Chen, N. Zheng, X. Wang, Y. Zhou, B. Ran, A deep reinforcement learning based distributed control strategy for connected automated vehicles in mixed traffic platoon, *Transp. Res. Part C: Emerging Technol.*, **148** (2023), 104019. https://doi.org/10.1016/j.trc.2023.104019
6. Y. Zhao, K. Wu, Z. Xu, Z. Che, Q. Lu, J. Tang, et al., Cadre: A cascade deep reinforcement learning framework for vision-based autonomous urban driving, preprint, arXiv:2202.08557.
7. S. Feng, H. Sun, X. Yan, H. Zhu, Z. Zou, S. Shen, et al., Dense reinforcement learning for safety validation of autonomous vehicles, *Nature*, **615** (2023), 620–627. https://doi.org/10.1038/s41586-023-05732-2
8. S. B. Prathiba, G. Raja, K. Dev, N. Kumar, M. Guizani, A hybrid deep reinforcement learning for autonomous vehicles smart-platooning, *IEEE Trans. Veh. Technol.*, **70** (2021), 13340–13350. https://doi.org/10.1109/TVT.2021.3122257
9. Y. Yao, N. Ma, C. Wang, Z. Wu, C. Xu, J. Zhang, Research and implementation of variable-domain fuzzy pid intelligent control method based on q-learning for self-driving in complex scenarios, *Math. Biosci. Eng.*, **20** (2023), 6016–6029. https://doi.org/10.3934/mbe.2023260

10. Z. Cao, S. Xu, X. Jiao, H. Peng, D. Yang, Trustworthy safety improvement for autonomous driving using reinforcement learning, *Transp. Res. part C: Emerging Technol.*, **138** (2022), 103656. https://doi.org/10.1016/j.trc.2022.103656

11. D. Rempe, J. Philion, L. J. Guibas, S. Fidler, O. Litany, Generating useful accident-prone driving scenarios via a learned traffic prior, preprint, arXiv:2112.05077.

12. G. Li, Y. Yang, S. Li, X. Qu, N. Lyu, S. E. Li, Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness, *Transp. Res. part C: Emerging Technol.*, **134** (2022), 103452. https://doi.org/10.1016/j.trc.2021.103452

13. P. Bhattacharyya, C. Huang, K. Czarnecki, Ssl-lanes: Self-supervised learning for motion forecasting in autonomous driving, preprint, arXiv:2206.14116.

14. Y. Du, J. Chen, C. Zhao, C. Liu, F. Liao, C. Chan, Comfortable and energy-efficient speed control of autonomous vehicles on rough pavements using deep reinforcement learning, *Transp. Res. Part C: Emerging Technol.*, **134** (2022), 103489. https://doi.org/10.1016/j.trc.2021.103489

15. B. Zou, J. Peng, S. Li, Y. Li, J. Yan, H. Yang, Comparative study of the dynamic programming-based and rule-based operation strategies for grid-connected pv-battery systems of office buildings, *Appl. Energy*, **305** (2022), 117875. https://doi.org/10.1016/j.apenergy.2021.117875

16. B. Du, B. Lin, C. Zhang, B. Dong, W. Zhang, Safe deep reinforcement learning-based adaptive control for usv interception mission, *Ocean Eng.*, **246** (2022), 110477. https://doi.org/10.1016/j.oceaneng.2021.110477

17. P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, et al., Outracing champion gran turismo drivers with deep reinforcement learning, *Nature*, **602** (2022), 223–228. https://doi.org/10.1038/s41586-021-04357-7

18. J. Duan, D. Shi, R. Diao, H. Li, Z. Wang, B. Zhang, et al., Deep-reinforcement-learning-based autonomous voltage control for power grid operations, *IEEE Trans. Power Syst.*, **35** (2020), 814–817. https://doi.org/10.1109/TPWRS.2019.2941134

19. S. Aradi, Survey of deep reinforcement learning for motion planning of autonomous vehicles, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 740–759. https://doi.org/10.1109/TITS.2020.3024655

20. H. An, J. Jung, Decision-making system for lane change using deep reinforcement learning in connected and automated driving, *Electronics*, **8** (2019), 543. https://doi.org/10.3390/electronics8050543

21. Y. Du, J. Chen, C. Zhao, F. Liao, M. Zhu, A hierarchical framework for improving ride comfort of autonomous vehicles via deep reinforcement learning with external knowledge, *Comput.-Aided Civ. Infrastruct. Eng.*, **38** (2023), 1059–1078. https://doi.org/10.1111/mice.12934

22. K. Jo, Y. Jo, J. K. Suhr, H. G. Jung, M. Sunwoo, Precise localization of an autonomous car based on probabilistic noise models of road surface marker features using multiple cameras, *IEEE Trans. Intell. Transp. Syst.*, **16** (2015), 3377–3392. https://doi.org/10.1109/TITS.2015.2450738

23. B. Okumura, M. R. James, Y. Kanzawa, M. Derry, K. Sakai, T. Nishi, et al., Challenges in perception and decision making for intelligent automotive vehicles: A case study, *IEEE Trans. Intell. Veh.*, **1** (2016), 20–32. https://doi.org/10.1109/TIV.2016.2551545

24. R. Guidolini, L. G. Scart, L. F. R. Jesus, V. B. Cardoso, C. Badue, T. Oliveira-Santos, Handling pedestrians in crosswalks using deep neural networks in the iara autonomous car, in *2018 International Joint Conference on Neural Networks (IJCNN)*, (2018), 1–8. https://doi.org/10.1109/IJCNN.2018.8489397
25. A. Sadat, M. Ren, A. Pokrovsky, Y. Lin, E. Yumer, R. Urtasun, Jointly learnable behavior and trajectory planning for self-driving vehicles, preprint, arXiv:1910.04586.
26. A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, et al., Odin: Team victortango's entry in the darpa urban challenge, *J. Field Rob.*, **25** (2008), 467–492. https://doi.org/10.1002/rob.20248
27. R. Kala, K. Warwick, Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization, *J. Intell. Rob. Syst.*, **72** (2013), 559–590. https://doi.org/10.1007/s10846-013-9817-7
28. X. Li, Z. Sun, D. Cao, Z. He, Q. Zhu, Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications, *IEEE/ASME Trans. Mechatron.*, **21** (2016), 740–753. https://doi.org/10.1109/TMECH.2015.2493980
29. S. Xie, J. Hu, P. Bhowmick, Z. Ding, F. Arvin, Distributed motion planning for safe autonomous vehicle overtaking via artificial potential field, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 21531–21547. https://doi.org/10.1109/TITS.2022.3189741
30. A. E. Sallab, M. Abdou, E. Perot, S. Yogamani, Deep reinforcement learning framework for autonomous driving, preprint, arXiv:1704.02532.
31. A. E. Sallab, M. Abdou, E. Perot, S. Yogamani, End-to-end deep reinforcement learning for lane keeping assist, preprint, arXiv:1612.04340.
32. H. Chae, C. M. Kang, B. Kim, J. Kim, C. C. Chung, J. W. Choi, Autonomous braking system via deep reinforcement learning, in *2017 IEEE 20th International conference on intelligent transportation systems (ITSC)*, (2017), 1–6. https://doi.org/10.1109/ITSC.2017.8317839
33. M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, R. Ke, Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving, *Transp. Res. Part C: Emerging Technol.*, **117** (2020), 102662. https://doi.org/10.1016/j.trc.2020.102662
34. M. Jaritz, R. De Charette, M. Toromanoff, E. Perot, F. Nashashibi, End-to-end race driving with deep reinforcement learning, in *2018 IEEE international conference on robotics and automation (ICRA)*, (2018), 2070–2075. https://doi.org/10.1109/ICRA.2018.8460934
35. L. Qian, X. Xu, Y. Zeng, J. Huang, Deep, consistent behavioral decision making with planning features for autonomous vehicles, *Electronics*, **8** (2019), 1492. https://doi.org/10.3390/electronics8121492
36. N. K. Ure, M. U. Yavas, A. Alizadeh, C. Kurtulus, Enhancing situational awareness and performance of adaptive cruise control through model predictive control and deep reinforcement learning, in *2019 IEEE Intelligent Vehicles Symposium (IV)*, (2019), 626–631. https://doi.org/10.1109/IVS.2019.8813905

37. S. Feng, X. Yan, H. Sun, Y. Feng, H. X. Liu, Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment, *Nature Commun.*, **748** (2021). https://doi.org/10.1038/s41467-021-21007-8

38. S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, J. Gao, Deep learning-based text classification: A comprehensive review, *ACM Comput. Surv.*, **54** (2021), 1–40. https://doi.org/10.1145/3439726

39. G. Li, S. Lin, S. Li, X. Qu, Learning automated driving in complex intersection scenarios based on camera sensors: A deep reinforcement learning approach, *IEEE Sens. J.*, **22** (2022), 4687–4696. https://doi.org/10.1109/JSEN.2022.3146307

AIMS Press