sensors

MDPI

Article

Research into Autonomous Vehicles Following and Obstacle Avoidance Based on Deep Reinforcement Learning Method under Map Constraints

Zheng Li *, Shihua Yuan, Xufeng Yin, Xueyuan Li and Shouxing Tang

National Key Laboratory of Vehicular Transmission, Beijing Institute of Technology, Beijing 100081, China
* Correspondence: lizheng123@bit.edu.cn; Tel.: +86-176-1017-9690

Abstract: Compared with traditional rule-based algorithms, deep reinforcement learning methods in autonomous driving are able to reduce the response time of vehicles to the driving environment and fully exploit the advantages of autopilot. Nowadays, autonomous vehicles mainly drive on urban roads and are constrained by some map elements such as lane boundaries, lane driving rules, and lane center lines. In this paper, a deep reinforcement learning approach seriously considering map elements is proposed to deal with the autonomous driving issues of vehicles following and obstacle avoidance. When the deep reinforcement learning method is modeled, an obstacle representation method is proposed to represent the external obstacle information required by the ego vehicle input, aiming to address the problem that the number and state of external obstacles are not fixed.

Keywords: autonomous driving; deep reinforcement learning; car following; obstacle avoidance; obstacle representation

Citation: Li, Z.; Yuan, S.; Yin, X.; Li, X.; Tang, S. Research into Autonomous Vehicles Following and Obstacle Avoidance Based on Deep Reinforcement Learning Method under Map Constraints. *Sensors* **2023**, *23*, 844. https://doi.org/10.3390/s23020844

Academic Editors: Maximilian Geisslinger, Felix Fent and Markus Lienkamp

Received: 24 November 2022
Revised: 31 December 2022
Accepted: 8 January 2023
Published: 11 January 2023

1. Introduction and Backgrounds

In a mature autonomous driving system, the environment perception module is equivalent to human eyes and ears, while the decision-planning module is equivalent to the human brain.

Most traditional decision-planning modules have a layered and hierarchical architecture, where the content is transferred layer by layer. Such a structure makes it easier to decompose complex tasks; thus, it is convenient to detect problems with the specific module. In addition, relying heavily on prior knowledge and formulae, traditional methods have strong interpretability. Owing to their good performance in terms of accuracy and interpretability, the traditional decision-planning modules are the mainstream system of autonomous driving technology at present. It is indeed established that the cascade structure in traditional approaches also brings drawbacks in the aspects of instantaneity and reliability. First, data computation and transmission through different layers hinder the instantaneity and flexibility of the modules. Second, the operation of traditional modules requires all elements in it to operate under normal condition, or it will be out of order, i.e., the system of traditional decision-planning modules is vulnerable and unreliable.

The adaptive cruise control (ACC) method has been widely applied in most traditional vehicle-following practices. In order to maintain a relatively stable distance with the target vehicle, the speed of the ego vehicle changes along with the target one on the straight road. As described in [1], the paper used the SMPC method, and a conditional linear Gaussian model was proposed and trained with actual measurements to estimate the probability distribution of the future speed of the preceding vehicle. Then, the paper built the relevant driving scenarios in the single lane in the CarMaker, applied the control strategy in the simulation scene, and verified it. Ioannou et al. modeled the ACC system as a hybrid system with constant acceleration, smooth acceleration, and a linear state feedback controller in [2]. Kitazono et al. in [3] adopted an adaptive neural network scheme to solve the traffic

stability problem in convoys. In addition, there are also many other different strategies used to adjust the distance between the ego and target vehicles, with the utilization of the ACC system. In [4], Choi et al. used a fixed distance strategy and a sliding mode control method to control the speed and distance between two vehicles.

There are two main traditional obstacle avoidance methods, including reference line fitting methods and artificial potential field methods. As for the reference line fitting method, the rationale of trajectory planning is widely adopted to generate a cluster of alternative trajectories, with the consideration of vehicle dynamics and the general vehicle attitudes. Then, these trajectories are sorted on the basis of cost value, and a collision-free obstacle avoidance trajectory with the least cost is finally obtained after collision detection. Finally, the trajectory tracking control algorithm is used to control the ego vehicle to avoid the obstacles. The artificial potential field method is a virtual force method, and its basic idea is to regard the vehicle's motion in the surrounding environment as the motion of the vehicle in the artificially established virtual position. The target point generates gravity, which guides the vehicle toward the target point, and the obstacles generate repulsion to avoid a collision between the ego vehicle and the obstacles. Under the combined force of gravity and repulsion, the ego vehicle moves along the direction of the potential field decline, resulting in an optimal path without collision. To avoid the local minimum, Ref. [5] proposed a virtual potential field detection circle model with an adjustable radius. The "minimum trap" formed by the obstacle repulsion field had been detected by the model in advance, whose success rate in obstacle avoidance has reached 90% or more. In [6], the obstacle avoidance function was introduced into the path planning layer, and the trajectory tracking layer transformed the linear time-varying MPC optimization problem into a quadratic programming problem, which has realized the path emergency obstacle avoidance and tracking.

Recently, some optimization methods based on traditional obstacle avoidance have been proposed and have achieved good results. In [7], a personalized motion planning and tracking control framework was proposed to prevent collisions between autonomous vehicles and obstacles ahead. The simulation results showed that the proposed framework could improve the corresponding driving performance according to the individualized needs of different passengers. The author in [8] improved an algorithm integrating path pruning, smoothing, and optimization with geometric collision detection based on the rapidly exploring random trees (RRT) algorithm. The simulation indicated that the vehicle could successfully track the path efficiently and reach the destination safely. An improved heuristic Bi-RRT algorithm, specialized for an unknown dynamic environment, was put forward by [9] for obstacle avoidance. Thence, the related experiments have verified the good performance of such an algorithm in differentiable coherence path generation, ensuring both ride comfort and stability of the vehicle.

Today, with the continuous development of artificial intelligence technology, more industries have entered the digital deepening and intelligent extension stage. The essence of digitization is data, while the foundation of intelligence is artificial intelligence technology. In the current stage of autonomous driving, technology based on data-driven methods has been fully developed. Until now, data-driven methods have not only played a part in the perception and positioning of autonomous driving technology, but have also imposed increasing influences in decision-making planning and autonomous driving control technology.

Reinforcement learning is a machine learning method, often used to deal with sequential decision-making issues. Since the reinforcement learning method is trained through trial and error in the training process, the method strongly depends on the simulation environment. For the autonomous driving decision-planning module based on the reinforcement learning method, the system's input can be raw perceptual data, such as pixel data from an RGBD camera or lidar points cloud data. Refs. [10–13] presented some model-free deep reinforcement learning methods for autonomous driving. For those data with high dimensions, a convolutional neural network for dimensionality reduction is necessary

before being passed into the reinforcement learning network. Being mature, the external perception and recognition technology based on cameras and lidars make accurate data accessible, including the ego surrounding environment status information, target vehicle status information, and self-vehicle status information. Moreover, developers can directly obtain a list of relevant objects from the simulation platform.

Recently, reinforcement learning algorithms have been increasingly utilized to deal with complex sequential decision-making and control problems. In the following field of vehicles, the literature [14] applied the policy gradient algorithm in reinforcement learning to train the CACC longitudinal controller to eliminate the errors of car following. However, the exploitation of the discrete control variable and more straight-forward environmental reward rules caused the repeated oscillations of the control quantities during the simulation. Ref. [15] proposed a potential game-combined multi-agent deep deterministic policy gradient (MADDPG) approach to optimize multiple UAVs' trajectories. Ref. [16] put forward a method that combined imitation learning with reinforcement learning enabling the agent to achieve a higher success rate in urban autonomous driving navigation scenarios. The literature [17] designed an autonomous driving system using the Q network and A* algorithm. For an unknown rough terrain scenario, Ref. [18] constructed a deep reinforcement learning method for safe local planning of a ground vehicle. However, the above-mentioned algorithms were still restricted within the longitudinal control on straight roads, but seldom addressed the issues related to the turning part of intersections.

Similarly, in the obstacle avoidance part of the high-precision map, obstacle avoidance methods are all based on reinforcement learning methods. The literature [19] investigated the dual-target behavior of trajectory tracking and obstacle avoidance based on waypoints through augmenting the agent's observation vector by the reward trade-off parameter, thus enabling the agent to adapt to changes in its reward function. In [20], the author proposed a deep Q-network-based method for collision-free decisions. The DQN-aided algorithm was introduced for determining the trajectory and velocity of the AV by receiving real-time traffic information from the base stations. In [21], they presented a novel learning-based control algorithm for ASV systems with collision avoidance. The proposed control algorithm combined a conventional control method with deep reinforcement learning to provide a closed-loop stability guarantee, uncertainty compensation, and collision avoidance. The scholars in [22] proposed an interpretable decision-making framework for autonomous vehicles at highway on-ramps adopting a latent space actor–critic-based method with asymmetric inputs. When constructing the simulation scene, the ego vehicle and the obstacle are usually simplified as a circle or a mass point, where the circle center is the center of the vehicle, and the radius is the safety distance. In this case, the interaction between the vehicle's polys (i.e., the attitude requirements of the vehicle) has been ignored, especially when passing through a narrow passable area. Hence, such methods did not make full use of the advantages of reinforcement learning. Meanwhile, the reinforcement learning method requires the input dimension of the object list to be unified. Such requirements have imposed more challenges to the scene where the external obstacles are not fixed.

Based on the above-mentioned problems ignored in previous studies on autonomous driving related to reinforcement learning, this paper conducts related research using the CARLA simulation platform. CARLA is an open-source autonomous driving simulation platform based on the UE4 engine's rendering, which provides users with the Python API interface, which can build rich driving scenes, and rich map elements can be obtained. In addition, the CARLA platform can also import self-built OpenDRIVE map files and OpenSCENARIO autonomous driving scene files, which can be customized according to the users' own needs.

In this paper, the factor of the braker is not taken into consideration in the CARLA simulation platform, while only two control quantities are required in the controlled vehicle, i.e., the steering wheel angle and the accelerator pedal strength. Previous studies on autonomous driving technology based on reinforcement learning usually discretize both control quantities, with the utilization of the reinforcement learning model based

on the DQN method for training. From a human driver's perspective, when driving a vehicle, the steering wheel angle and the brake pedal's position can take any value within the limit range. In other words, the input variables are continuous. To give full play to the characteristics of the neural network model and get closer to the level of human driver operation, the reinforcement learning basic model in this paper adopts the twin delayed deep deterministic policy gradient network framework that can generate actions in a continuous space. Additionally, this method will be referred to as TD3 for short. To sum up, the relevant scenarios of car following and obstacle avoidance in this paper are all carried out using the CARLA simulation platform, and the TD3 method is used as the calculation benchmark. The main contributions of this article are as follows:

- A car-following model method based on reinforcement learning is proposed, which combines the relevant map elements in the map and the Frenet coordinate system, and can follow the car stably on straights, intersections, and curves.
- We propose an external obstacle state input algorithm that can be adaptive to the ego vehicle's speed. Additionally, this algorithm can represent the shape and pose of a random number of obstacles.
- An obstacle avoidance method using reinforcement learning under map constraints is designed based on the improved obstacle input method. This method can still perform well when passing through narrow passable areas.

The rest of this paper is organized as follows. In Section 2, a reinforcement learning-based target tracking model is presented, and this model can have perfect curve passing performance. Section 3 mainly introduces a multi-obstacle representation method. Finally, Section 4 concludes the paper.

2. Reinforcement-Learning-Based Car following Method Using the Map Constraints

2.1. Reinforcement Learning Theory

To meet the continuous demand of action space for policy optimization, this paper selects the TD3 as the basic algorithm model based on the actor–critic structure.

The actor–critic algorithm is an extension of the policy gradient method. In its single-step updated calculation, the core formula can be expressed as follows:

$$\begin{cases} \delta = r_t(s_t, a_t) + v_\omega(s_{t+1}) - v_\omega(s_t) \\ \omega_{t+1} = \omega_t + \alpha^\omega \delta_t \nabla_\omega v_\omega(s_t) \\ \theta_{t+1} = \theta_t + \alpha^\theta \nabla_\theta \log \pi_\theta(a_t|s_t)\delta_t \end{cases} \tag{1}$$

Among them, the first line is the calculated TD error, the second line is the updated critic model parameters, and the third line is the updated actor model parameters.

TD3 Theoretical Knowledge

Google DeepMind combines the above actor–critic method with the DQN network and proposes a deep deterministic policy network (deep deterministic policy gradient), referred to as DDPG. The calculation formula of its objective function is as follows:

$$\begin{cases} J(\omega) = min_\omega E_\beta\left[\frac{1}{2}(r_t + \gamma Q^\omega(s_{t+1}, a_{t+1}) - Q^\omega(s_t, a_t))^2\right] \\ J(\theta) = max_\theta E_\beta[Q^\omega(s_t, \mu(s_t))] \end{cases} \tag{2}$$

Among them, ω is the critic network parameter, θ refers to the actor network parameter, $Q(s, a)$ is the action value function, $\mu(s)$ is the deterministic strategy, and γ is the discount factor. By solving the above objective function, we can obtain

$$\begin{cases} \Delta\omega = E_\beta[(r_t + \gamma Q^\omega(s_{t+1}, a_{t+1}) - Q^w(s_t, a_t))\nabla_\omega Q^\omega(s_t, a_t)] \\ \Delta\theta = \alpha E_\beta\left[\nabla_\omega Q^\omega(s_t, a_t)\Big|_{a=\mu_\theta(s)} \nabla\mu_\theta(s_t)\right] \end{cases} \tag{3}$$

The TD3 method in this paper is called the twin delayed deep deterministic policy gradient, which has been improved as follows based on DDPG:

(1) Actor network and critic network delayed asynchronous updates. After updating the critic network several times, update the actor network to ensure that the training of the actor network is more stable.

The actor network will be updated to ensure its stability with regard to the training performance.

(2) Two sets of twin critical networks are used, and the smaller value is selected in the actual calculation to prevent the network from overestimating.

(3) Adding perturbations to the actor's internal network output ε_2 makes the target network estimate more accurately and robustly.

The structure of the TD3 network is constructed in Figure 1:

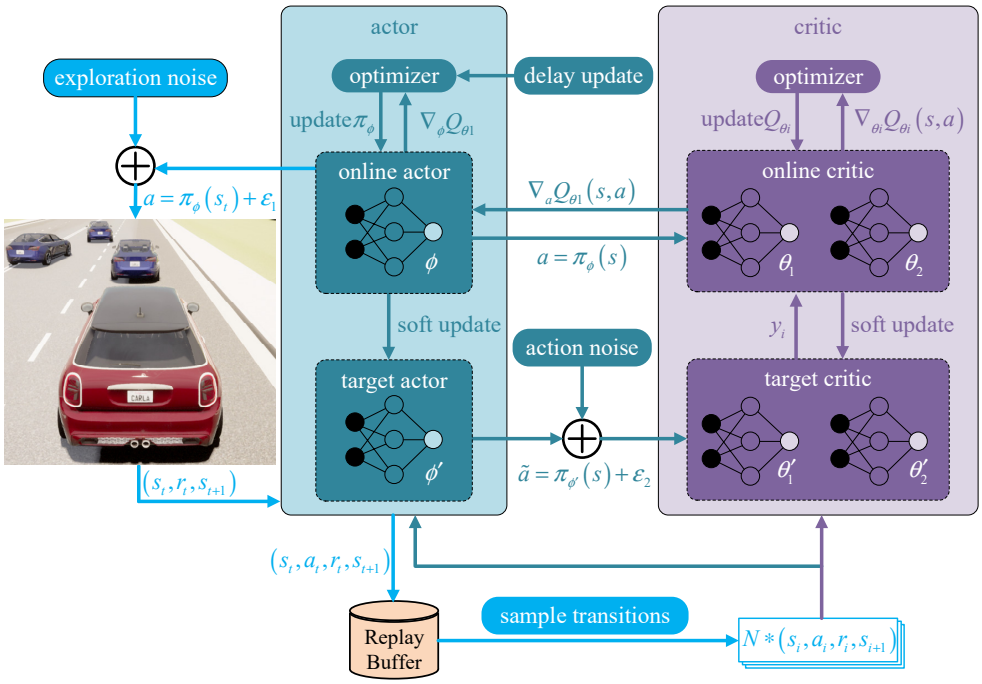

Figure 1. The improved TD3 network in this paper.

Combined with the reinforcement learning TD3 structure and the simulation process performance used in training, data units are stored in the container named replay buffer during the entire training process. This paper adopts a method that dynamically adjusts the replay buffer's size according to different training stages, which is divided into three stages [23]. The first stage refers to the process of data storage, whose capacity is small. Therefore, it is easy to reach the upper limit and enter the second stage, which refers to the process of storing data while training. In this stage, the capacity of the replay buffer is also increased until the third stage is entered. In the third stage, the final replay buffer capacity is reached, but the data volume does not increase at this stage. Whenever new data are added, the farthest data will be popped out from the replay buffer to ensure the constancy of the replay buffer's capacity.

2.2. State Function and Reward Function

2.2.1. Frenet Coordinate System Introduced

In urban structured roads, one of the challenges in the path planning module of autonomous driving technology is to continuously express the relative position between the vehicle and the road map during the whole driving process, resulting in a fuzzy relative position relationship between them. If the lane information is ignored, the degree of freedom of the path will be too high to control so that it is easy to violate road traffic rules. During the DARPA Automotive Challenge, Stanford University proposed the Frenet

coordinate system, as shown in Figure 2, in which s represents the distance along the road and l represents the displacement from the longitudinal line.

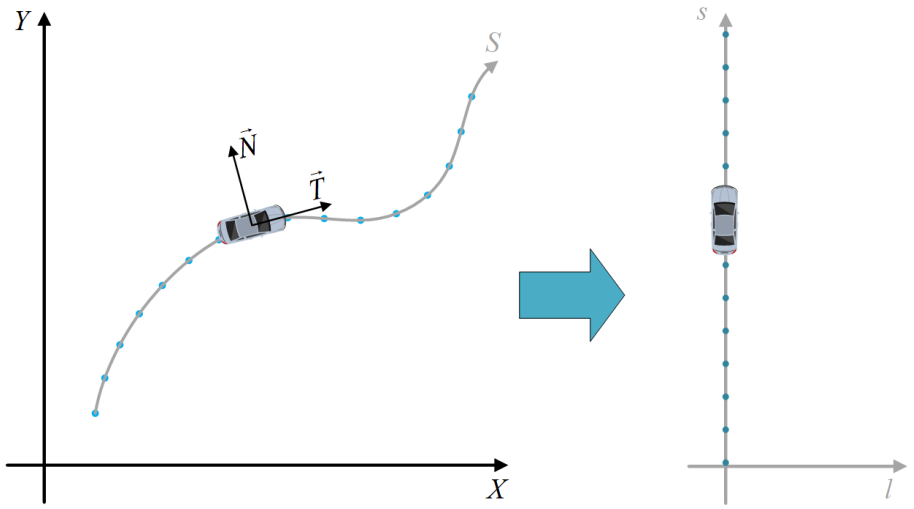

Figure 2. Comparison of the trajectory along the reference line in the Cartesian coordinate system and the Frenet coordinate system.

If the vehicle drives along the lane (usually the lane center reference line), in the Cartesian coordinate system the trajectory of the vehicle is consistent with the reference line, as shown on the left of the above figure. In the Frenet coordinate system, if the vehicle drives along the lane, the trajectory becomes a straight line traveling along s, as shown on the right of the above figure, which greatly simplifies the difficulty of trajectory calculation. In normal driving scenarios, the trajectory of the vehicle and the reference line usually do not overlap. As shown in Figure 3, the blue one is the actual driving trajectory of the vehicle, while the dark gray one represents the reference line trajectory, and the points on the reference line are waypoints. In the global coordinate system, the motion state of the ego vehicle at any time t can be described as $\left[\vec{x}, \theta_x, v_x, a_x\right]$, in which $\vec{x}$ represents the current position of the vehicle marked as Q, and in the global Cartesian coordinate system it is represented as $\vec{x}(t) = (x(t), y(t))$. In the Frenet coordinate system, P is the projection point of the current position point Q on the reference line, s refers to the arc length of the road lane from the starting point of the reference line to the projection point P, and l is the distance between the Q point and the projection point P, so it can be described by the longitudinal displacement s and the lateral displacement l relative to the reference line, i.e., $\vec{x}(t) = (x(t), y(t)) = (s(t), l(t))$. In the actual code, the position of the projected point P is calculated by the two closest waypoints to the Q point.

2.2.2. Environmental Scenario Analysis

When the ego vehicle follows the target vehicle ahead at a non-junction position such as a straight road or a one-way road, the reference waypoint is on the center line of each lane, and the tracking exercise will be simpler. At the road junction, the distribution of waypoints is relatively dense, and the calculated closest point may be different from the lane where the preceding vehicle is located, as shown in the following Figure 4.

Therefore, when building the environmental scene and driving scenario in this paper, the driving options (straight, turn left, turn right) of the preceding vehicle at the junction should be confirmed first according to the change in the heading angle of the preceding target vehicle, and then a junction option list can be obtained. The ego vehicle reference line is updated according to the target junction option set. The detailed algorithm steps are presented as Algorithm 1.

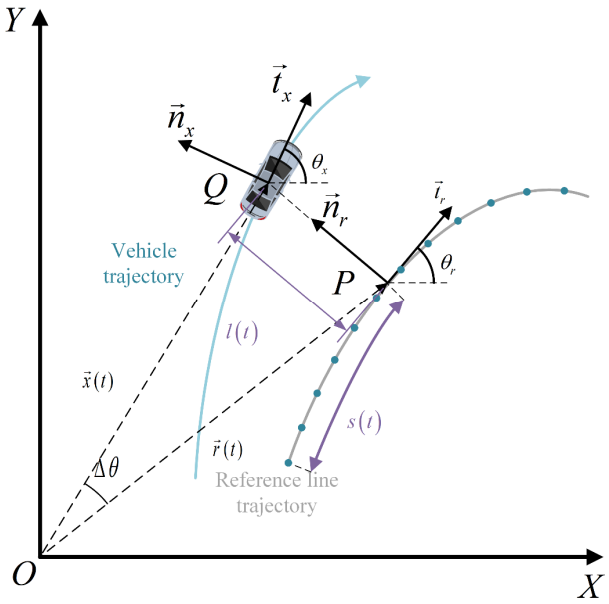

Figure 3. Schematic diagram of Frenet coordinate system and Cartesian coordinate system in normal driving state.

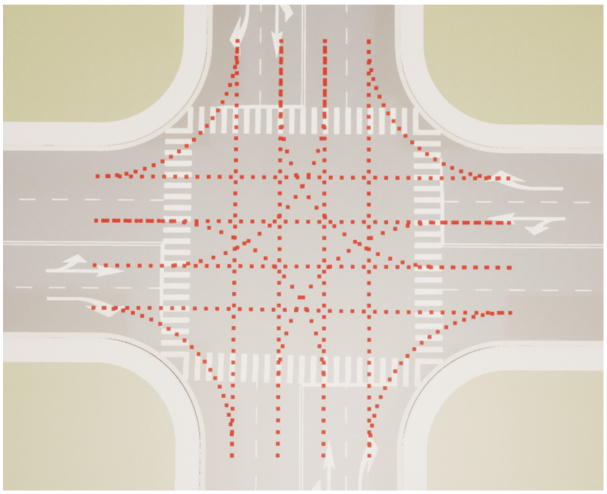

Figure 4. Waypoints at the road junction.

Algorithm 1. Reference waypoints' set acquisition algorithm in each episode

Initialize the ego vehicle reference waypoint list.
Initialize the target vehicle steering option list.
Use the nearest waypoint from the CARLA map to the location of the ego car as the current waypoint.
Repeat (for each episode):
while the size of waypoints is less than N **do**:
Add the waypoint list to the current waypoint, and find the next waypoint at the interval of *sampling_dis* according to the current waypoint.
If there is more than one choice for the next waypoint area, the next waypoint will be chosen randomly.
Set the new next waypoint as the new current waypoint.
Update the steering option of the target vehicle at the intersection to the waypoint list.
If the waypoint of the ego vehicle reference waypoint list is the first waypoint at the junction, update the waypoint according to the target steering option list.
Re-update the ego vehicle reference waypoint list.
When the ego vehicle exits the corresponding junction, pop the target vehicle steering option list.
end while

2.2.3. State Function

The set list of ego vehicle reference waypoints is made of all waypoints on the center line of the lane, which are equivalent to a reference line. According to the content of the previous section, the position of the ego vehicle can be projected onto the correct lane by using the set list of reference waypoints so as to obtain the longitudinal distance s and lateral distance l of the ego vehicle relative to the reference line.

Therefore, under the scene without map constraints, the ego vehicle's and target's relative state information can be utilized. However, when there are map constraints, and the state input should be considered, this paper selects the relative state of the ego vehicle and the target vehicle as $S_t^{ego\_obj}$, as well as the relative state of the ego vehicle and the reference waypoint set list as $S_t^{ego\_wps}$.

$$S_t = \left\{ S_t^{ego\_obj}, S_t^{ego\_wps} \right\} \tag{4}$$

When considering the relative relationship between the ego vehicle and the target vehicle, different from the reinforcement learning method without map constraints, it needs to be calculated in the Frenet coordinate system as follows:

$$\begin{aligned} S_t^{ego\_obj\_1} &= long\_dis_{obj} - long\_dis_{ego} \\ S_t^{ego\_obj\_2} &= clip\left(\frac{vel_{ego\_long}}{clip\left(vel_{obj\_long}, min, max\right)}, min, max \right) \\ S_t^{ego\_obj} &= S_t^{ego\_obj\_1} + S_t^{ego\_obj\_2} \end{aligned} \tag{5}$$

Among them, $S_t^{ego\_obj\_1}$ represents the distance between the ego vehicle and the target vehicle, i.e., the integral arc length between the projection points of the ego vehicle and the preceding vehicle on the lane. Under the constraints of the map lane, this state can more accurately reflect the relative position of the ego car and the preceding target vehicle than the line segment distance between the ego and target vehicles. $S_t^{ego\_obj\_2}$ is the ratio of the projection of the speed of the ego car and the target car in the Frenet coordinate system in the s direction. The closer the ratio $S_t^{ego\_obj\_2}$ is to 1, the closer the speed components of the ego car and the target vehicle are. The function of $clip()$ in the formula is to intercept the value, which can prevent the denominator from being zero or the ratio from being too large and blocking within the boundary range.

When considering the relative relationship between the ego vehicle and the set of waypoints, the lateral distance between the ego vehicle, and the lane center line, the yaw angle deviation between the ego vehicle and the corresponding waypoint are mainly analyzed, as shown in the following formula

$$\begin{aligned} S_t^{ego\_wps\_1} &= lateral\_dis_{obj\_wps} \\ S_t^{ego\_wps\_2} &= yaw_{ego} - yaw_{wp} \end{aligned} \tag{6}$$

In addition, the state function also introduces preview points, i.e., three waypoints in the direction of the vehicle's forward direction in the lane center line. The difference between these three preview waypoints and the physical quantities related to the vehicle is expressed as follows:

$$S_t^{ego\_wps\_3} = \begin{bmatrix} wp\_1_x - ego_x & wp\_1_y - ego_y & wp\_1_{yaw} - ego_{yaw} \\ wp\_2_x - ego_x & wp\_2_y - ego_y & wp\_1_{yaw} - ego_{yaw} \\ wp\_3_x - ego_x & wp\_3_y - ego_y & wp\_1_{yaw} - ego_{yaw} \end{bmatrix} \tag{7}$$

Therefore, the relative state of the ego vehicle and the set of reference waypoints $S_t^{ego\_wps}$ can be expressed as follows:

$$S_t^{ego\_wps} = S_t^{ego\_wps\_1} + S_t^{ego\_wps\_2} + S_t^{ego\_wps\_3} \tag{8}$$

In the external interface of the CARLA platform, three variable parameters control the driving of the vehicle: the straight-forward parameter, the steering parameter, and the braking parameter. For the scene in this section, the braking parameters are not considered for the time being. Therefore, the outputs of the entire neural network contain the straight direction control command and the steering direction control command, which are consistent with the model without map constraints. To make the action value acting in the simulation environment smoother, the two variables directly outputted from the neural network are adjusted as follows and then directly act on the CARLA simulation platform:

$$\begin{aligned} a_{throttle}^{t+1} &= a_{throttle}^{t} + \gamma_1 \left(a_{throttle}^{t+1} - a_{throttle}^{t} \right) \\ a_{steer}^{t+1} &= a_{steer}^{t} + \gamma_2 \left(a_{steer}^{t+1} - a_{steer}^{t} \right) \end{aligned} \tag{9}$$

γ_1 and γ_2 are weight values from 0 to 1. It can be seen that the smoothed current time variable is the weighted value of the variable directly outputted by the neural network at the last time and the current time.

In the episode-based training process of reinforcement learning, if some policy solutions with relatively poor effects cannot be terminated in time, the invalid training may waste resources and time. Therefore, the reinforcement learning episode exit condition in this paper involves multiple termination trigger conditions.

(1) Among the trigger conditions, the most important and intuitive termination condition is the occurrence of a collision event. Safety is undoubtedly the most important thing for self-driving vehicles. A collision detector can be added in CARLA. When the ego vehicle collides with other objects, a collision signal is generated. Within a period of the round, once the collision signal is detected, the current round will exit immediately. The relevant scenario is shown on the left side of Figure 5.

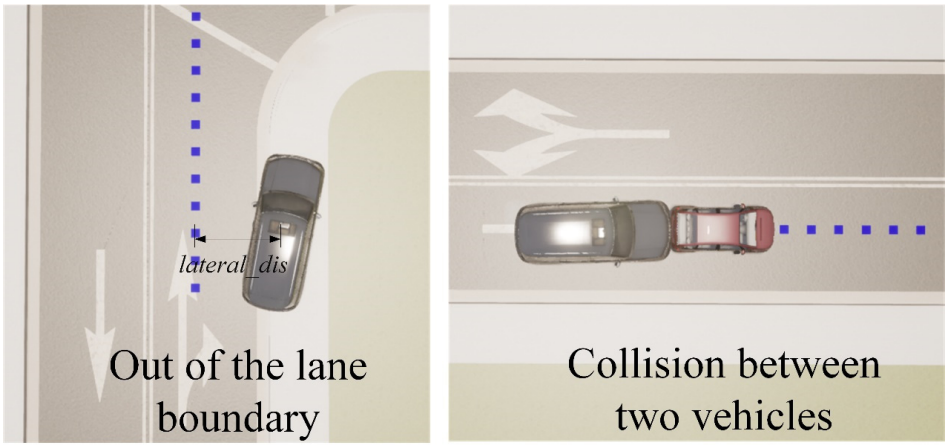

Figure 5. Two scenarios for exiting training episodes.

(2) The lane departure event is also a relatively intuitive termination condition. In the early stage of training, the neural network training is not stable, and the trajectory of the vehicle sometimes deviates too far from the correct lane; thus, a restriction needs to be added. Combined with the physical quantity of the lateral offset of the vehicle lane center line point set in the above state factors, the maximum value of the lateral offset is specified. When the lateral offset exceeds the threshold, the termination condition will be triggered, and the current round will exit immediately. This scenario is shown on the right side of Figure 5.

(3) The neural network outputs two control commands, namely the accelerator command and the steering wheel command. In the early stage of training, the speed of the ego vehicle may be too low or even close to zero. Therefore, a minimum speed threshold is specified. When the speed of the ego vehicle is lower than the threshold for several consecutive steps, the termination condition will be triggered and the current episode will exit.

(4) Ideally, during an episode of training, the ego vehicle will not meet the above-mentioned exit conditions, but it cannot be stuck in a certain episode forever and cannot exit. Therefore, in this paper, the maximum number of training steps is added in the episode training process. When the number of training steps in the episode reaches the maximum number of steps, the termination condition will be triggered and the current episode will exit.

2.2.4. The Design of the Reward Function

As mentioned above, in addition to the influence of the relative position between two vehicles with regard to the reward, when designing the reward function, the influence of the set of lane center line points in the map and the relative attitude of the ego vehicle with regard to the reward function should also be considered. The design of the reward function corresponds to the above state function. For the set of waypoints, the reward function is as follows:

$$\begin{aligned} r_t^{ego\_wps\_1} &= k_1 \left| lateral\_dis_{obj\_wps} \right| + b_1 \\ r_t^{ego\_wps\_2} &= k_2 \left| yaw_{ego} - yaw_{wp} \right| + b_2 \\ r_t^{ego\_wps} &= r_t^{ego\_wps\_1} + r_t^{ego\_wps\_2} \end{aligned} \tag{10}$$

Among them, k_1, k_2, and b_1, b_2 are the artificially designed algorithm parameters. In actual training, the farther the ego vehicle deviates from the center of the lane, the smaller the reward value; the larger the yaw difference angle between the ego car and the center line of the lane, the smaller the reward value. That is, the vehicle will have a higher reward when driving along the reference center line value. For the positions and attitudes of the ego vehicle and the preceding/target vehicle, similar to the formula, the reward function is as follows, and the parameters in the formula are similar to those too:

$$r_t^{ego\_obj\_1} = \begin{cases} b, S_t^{ego\_obj\_1} \in \left[d_{threshold}^{-}, d_{threshold}^{+} \right] \\ -k_1 \cdot S_t^{ego\_obj\_1}, S_t^{ego\_obj\_1} > d_{threshold}^{+} \\ -k_2 / S_t^{ego\_obj\_1}, S_t^{ego\_obj\_1} < d_{threshold}^{+} \end{cases} \tag{11}$$

In the above formula, the predetermined tracking distance range of the ego vehicle is $\left[d_{threshold}^{-}, d_{threshold}^{+} \right]$. Similar to the actual vehicle driving process, the following distance will be enlarged along with the increase in the ego vehicle speed. After the introduction of the Frenet coordinate system, with the aim of maintaining a relatively stable attitude of both ego and target vehicles, the reward function also needs to be related to the projection ratio of the two velocities in the s direction.

$$\begin{aligned} \| \vec{v}_{ego\_s} \| &= clip\left(\| \vec{v}_{ego\_s} \|, v_{min}, v_{max} \right) \\ \| \vec{v}_{obj\_s} \| &= clip\left(\| \vec{v}_{obj\_s} \|, v_{min}, v_{max} \right) \\ r_t^{ego\_obj\_2} &= k_3 - clip\left(max\left(\frac{\| \vec{v}_{ego\_s} \|}{\| \vec{v}_{obj\_s} \|}, \frac{\| \vec{v}_{obj\_s} \|}{\| \vec{v}_{ego\_s} \|} \right), value_{min}, value_{max} \right) \\ r_t^{ego\_obj} &= r_t^{ego\_obj\_1} + r_t^{ego\_obj\_2} \end{aligned} \tag{12}$$

In the formula, the projection of the speed of the ego vehicle and the target vehicle in the s direction are calculated, respectively, the maximum and minimum values are specified for the interception, and the relevant reward function design uses the ratio of the speeds of the ego vehicle and target vehicle. It can be seen from the above formula that when the discrepancy between the two is significant, the reward function is more minor, and only when the projection values of the ego vehicle and the target vehicle in the s direction are close or equal, the reward function is close to, or even reaches, the maximum value. In addition, as shown in the above state function, when the ego vehicle triggers the round

termination condition in the round and does not reach the maximum number of training steps in the round, that is, when it exits the training under abnormal conditions, a large negative value needs to be added.

$$r_t^{terminate} = \begin{cases} c, \text{trigger termination condition} \\ 0, \text{other} \end{cases} \tag{13}$$

In addition to the above three reward functions, the discussion is also carried out according to the vehicle's driving characteristics on straights and curves. On the straight road, the vehicle is supposed to eliminate the lateral offset, while on the curved road, especially when at a corner, the vehicle is expected to perform a large steering behavior in order to achieve cornering. Thence, the steer reward function can be expressed as follows:

$$r_t^{steer} = \begin{cases} k_4 steer_{ego}, \text{In the junction area} \\ 0, \text{other} \end{cases} \tag{14}$$

So far, the reward function based on map constraints can be expressed as follows

$$r_t = r_t^{ego\_wps} + r_t^{ego\_obj} + r_t^{terminate} + r_t^{steer} \tag{15}$$

2.2.5. Simulation Results and Analysis

In the simulation process, a curve scene is intercepted for analysis. The trajectory and lateral distance are illustrated in the following figure.

PID is the abbreviation of "proportional–integral–derivative", and PID is the most commonly used control method. Figure 6 shows the simulation results of the target vehicle controlled by PID and the ego vehicle controlled by reinforcement learning during the steering process. The target vehicle is driven by the traditional trajectory tracking control method, and the PID method is used in both the longitudinal and lateral directions of the target vehicle. The center line of the lane at the turning point is used as the trajectory and the preview point is extracted from it. In Figure 6, the black curve is the trajectory of the lane's center line on the curve, while the red one represents the trajectory of the vehicle controlled by the reinforcement learning model at the turning point. The blue one represents the driving curve of the target vehicle using the PID method. According to the comparison curve of the simulation results, it is found that the reinforcement learning method achieved a more stable steering effect at the curving area than the target vehicle even though the former does not input the preview points like the PID method does.

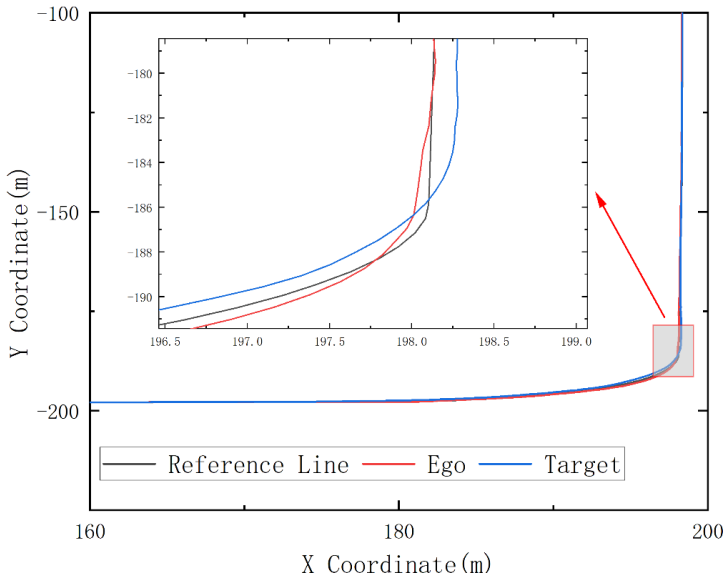

Figure 6. Turning process trajectory comparison chart.

In Figure 7, the black curve represents the ego vehicle controlled by reinforcement learning, and the red one refers to the target vehicle controlled by the traditional PID method. The x-coordinate is the number of time periods experienced in the steering process, and the y-coordinate is the lateral distance between the driving trajectories of the two vehicles and the lane center reference line. It can be seen from the curve results that the ego vehicle car controlled by reinforcement learning can also reach a relatively stable state at the turning point during the following process, and can fit the reference line well.

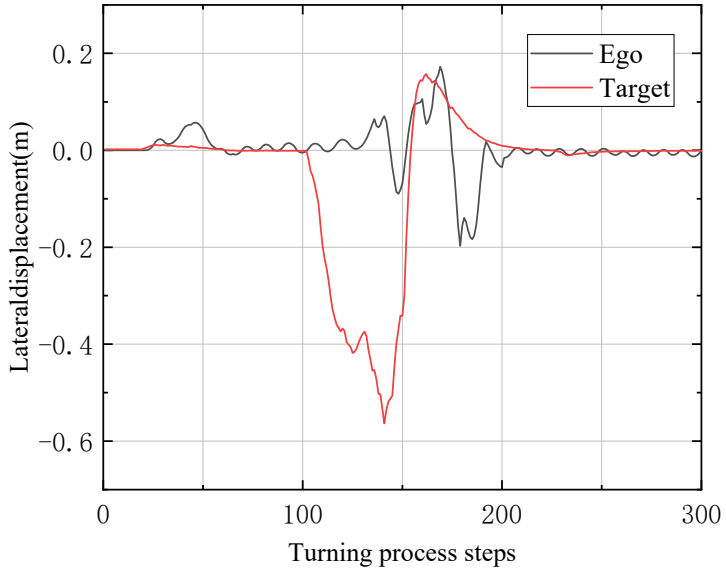

Figure 7. Comparison between the lateral displacement of the ego and target in the turning process.

Figure 8 demonstrates the comparison of the throttle scalar input value during the turning process of the ego vehicle controlled by reinforcement learning and the target vehicle controlled by PID trajectory tracking. From the comparison of the figures, it can be seen that the throttle fluctuation range of the vehicle controlled by reinforcement learning is small, and the fluctuation range is relatively smooth. The trolley controlled by the PID method has a larger fluctuation range.

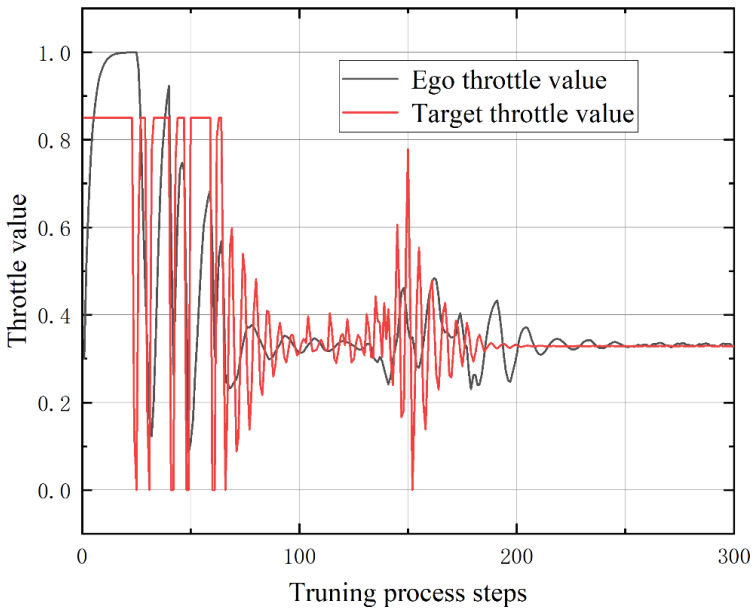

Figure 8. Comparison between throttle value of the ego and target in the turning process.

Figure 9 exhibits the projection of the distance between the ego vehicle and the target vehicle on the lane during the turning process. From the curve results, it is obvious that the following distance can be stabilized at 16 m on the straight road, and the distance at the turn has increased, and when it returns to the straight road again, the distance can also be restabilized at 16 m.

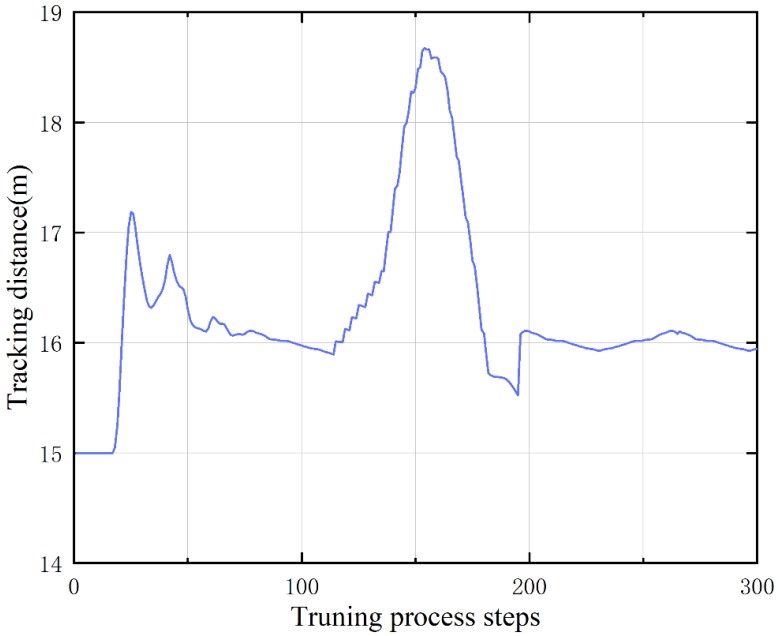

Figure 9. Tracking distance between the ego and target.

3. Obstacle Avoidance Using Deep Reinforcement Learning Approach

3.1. Improved Obstacle Representation

For the obstacle avoidance, owing to the fact that the number of external obstacles is not fixed, some researchers have adopted the rasterization processing method to rasterize the observation range. A rectangle grid or a sectorial grid is used to encode obstacles to determine the location and size of the obstacles. Moreover, the obstacle information is dimensionally reduced to address the issue that the input dimension is not fixed. For rasterization methods, a question that needs to be considered is the accuracy of the grid. The smaller the grid size, the higher the accuracy of its representation, and the larger the input dimension of the reinforcement learning neural network, which is more inconducive to episode training. However, if the grid size is enlarged, the accuracy will be worsened and the passable area will be shrunk, making it more difficult for the vehicle to get through the passable narrow area in complex obstacle avoidance scenarios. In a way, the grid map method will hinder the performance of the vehicle when passing through the narrow areas.

The perceptual information that the ego vehicle receives at a certain moment is unknown globally, and the perceptual results constantly change from time to time. In the case of uncertain external obstacles, taking the external obstacle input as the research object, it is difficult for us to utilize the fixed-dimensional data input to represent the obstacle information. Inspired by this limitation, it is thought that the shape of the ego vehicle is determined, and the ego vehicle's position, attitude, and speed information can also be accurately obtained through the simulation platform or sensors. Suppose that a polygon is fixed on the vehicle body coordinate system. The polygon follows the movement of the vehicle, and the shape can change according to the speed of the vehicle. When the speed of the vehicle is slow, the polygon range is small. The extent of the polygon also expands in accordance with the increase in the vehicle's speed. When the ego vehicle is driving in an obstacle avoidance scene, the state function of the obstacle input part only considers the part that enters the polygonal area. For the reinforcement learning neural network, this

method can screen and sort out the complex obstacle scenes with unfixed numbers and the location distribution, but only a part of external obstacles within a certain range will be quantitatively calculated.

The improved state input of the self-vehicle obstacle avoidance process is shown in Figure 10. The input is divided into two areas, the sectorial area with the center of the ego vehicle as the dot and the extended rectangular area of the ego vehicle in the forward direction. The angle of the sector is determined by the maximum slip angle of the front wheels on both sides of the vehicle, while the width of the extended rectangular area depends on the bounding box length of the vehicle. Both areas take the center line of the vehicle's forward direction as the axis. Among them, the sectorial area is divided by a group of equally angularly spaced line segments with the length of the sector radius. After encountering obstacles and lane boundary lines within the fan-shaped range, the corresponding inner-sector line segments will intersect with them. The data of line segments intersecting with the obstacles and lane boundary lines are stored as a list to represent the obstacle status within the sector. Similarly, the rectangular extension area of the yellow part is also divided by a cluster of equally spaced line segments. When obstacles and lane boundary lines are encountered and intersected with each other, the intersecting data of different interval line segments and obstacles are stored in a list to represent the status of obstacles in a rectangular area. The specific algorithm is shown in Algorithm 2:

Algorithm 2. Obstacle input representation algorithm

Initialize the sectorial and rectangular areas according to the maximum slip angle of the front wheel, the size of the ego vehicle frame, and the speed of the ego vehicle.
Obtain the number N_{sector} and N_{rec} of the sectors and rectangles, and determine the angular interval of the sector β and the distance interval of the rectangle ω.
Combine multiple obstacle vehicles and lane boundaries into one obstacle set.
Initialize the sector area representation $S_{sector} = [[\beta_1, r], \cdots, [\beta_2, r]]$, and the rectangle area representation within the extended area $S_{rectangle} = [[\omega_1, l], \cdots, [\omega_j, l]]$, in which r represents the radius of the sector and l represents the length of the rectangle.
Calculate sector representation.
For N_{sector} in cycles **do**:
  if The current line segment intersects the set of obstacles:
    Confirm the intersection of the current line segment and the obstacle, and calculate the distance from the intersection to the origin, and update the corresponding S_{sector}
  else:
    The current S_{sector} corresponding section is not updated
end for
for $N_{rectangle}$ in cycles do:
  if The current line segment intersects the set of obstacles:
    Confirm the intersection of the current line segment and the obstacle, and calculate the distance from the intersection to the corresponding horizontal reference line of the vehicle, and update the corresponding $S_{rectangle}$.
  else:
    The current $S_{rectangle}$ corresponding section is not updated.
end for
  Normalize the updated S_{sector} and $S_{rectangle}$.

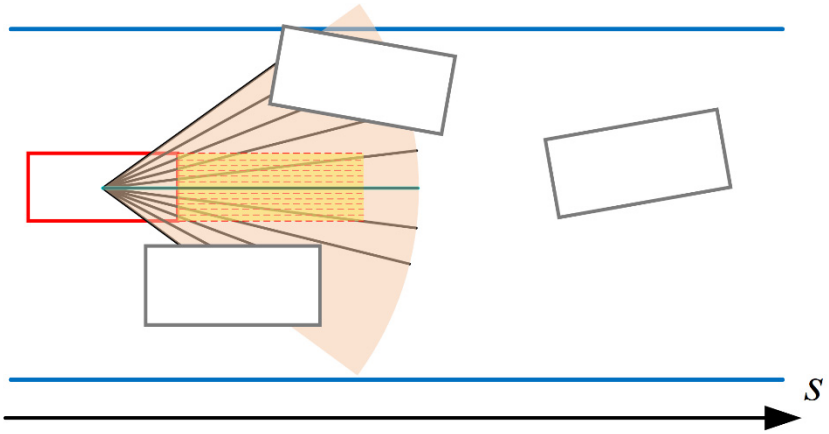

Figure 10. Schematic diagram of self-vehicle obstacle avoidance state input.

In the actual training process, the number of divisions of sectors and rectangles is set in advance by the parameter file, and the intervals are the same size, so the characterization of sectors and rectangles can be simplified as follows:

$$\begin{aligned} S_{sector} &= [r_1, r_2, \cdots, r_{N_{sector}}] \\ S_{rectangle} &= \left[d_1, d_2, \cdots, d_{Nrectangle}\right] \end{aligned} \tag{16}$$

As mentioned above, the lengths of the sectorial area and the rectangular area in the process are proportional to the speed of the ego vehicle:

$$\begin{aligned} r &= k_3 \cdot v_{ego} \\ l &= k_4 \cdot v_{ego} \end{aligned} \tag{17}$$

To represent the obstacle state at different ego vehicle speeds and calculate the reward function in the relevant state more accurately, the representation list in the region needs to be normalized, and the elements in S_{sector} and S_{sector} can be expressed as follows:

$$\begin{aligned} r_i &= r_i / r \\ d_i &= d_i / l \end{aligned} \tag{18}$$

Take Figure 10 as an example; the number of segmented rectangular areas is 10. After the algorithm and various optimizations, the representations in the sectorial area and the rectangular area are shown as

$$\begin{aligned} S_{sector} &= [0.35,\ 0.39,\ 0.52,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0.52,\ 0.39,\ 0.35] \\ S_{rectangle} &= [1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1] \end{aligned} \tag{19}$$

On the left side and right side of the sectorial area, there is an intersection with the obstacle, and the simplified distance from the intersection to the center of the vehicle is as above. If the area near the center of the sectorial symmetry axis has no intersection with the obstacle, the corresponding state list element is 1. In the rectangular area, there is no intersection between the rectangular area and the obstacle in the current round, so the elements of the state list are all 1. From the current scene graph and this representation, it can be directly inferred in the current round that the ego vehicle can go straight forward from the current attitude.

3.2. Corresponding Reward Function Design

The sectorial areas and rectangular areas mentioned above are all forward detection areas during the driving process of the ego vehicle. The obstacles or lane boundaries in the detection area might affect the driving state of the ego vehicle in the following process.

According to the schematic diagram of the obstacle avoidance scene and the corresponding state representation of obstacles and lane boundaries, it is found that the larger the element value, the farther the direction is from the obstacle or the lane boundary. When the element value is 1, it means that within the current detection range, there are no obstacles or lane boundaries in this direction. For the obstacles and lane boundaries within the detection range, the closer they are to the ego vehicle, the greater the possibility that the ego vehicle will collide with the obstacles or exceed the lane boundary; the closer the obstacles or lane boundary is to the forward direction of the ego vehicle, the higher the possibility that the ego vehicle will collide with an obstacle or exceed the lane boundary.

In order to describe the reward function of obstacles at different positions at each moment relative to the ego vehicle, the state bias vector and weight vector can be introduced. B_{sector} represents the sectorial state bias vector and $B_{rectangle}$ represents the rectangle state bias vector. The role of bias vectors is to make the state vectors symmetrically distributed. During the calculation process of reward function in this part, the two-state bias vectors and the obstacle and lane boundary state functions are superimposed to generate new state vectors. The inner product of the new state vectors and the weight vectors obtained by this part of the operation are used as the benchmarks for the reward function of the obstacle and lane boundary parts, as described in the following formula,

$$\begin{aligned} B_{sector} &= \begin{bmatrix} b_{s\_1} & b_{s\_2} & \cdots & b_{s\_N_1} \end{bmatrix} \\ W_{sector} &= \begin{bmatrix} w_{s\_1} & w_{s\_2} & \cdots & w_{s\_N_2} \end{bmatrix} \end{aligned} \tag{20}$$

$$r_{sector} = (S_{sector} - B_{sector}) \cdot W_{sector}^T = \begin{bmatrix} x_{s_1} & x_{s_2} & \cdots & x_{s_{N_1}} \end{bmatrix} \cdot \begin{bmatrix} w_{s_1} \\ w_{s_2} \\ \cdots \\ w_{s_3} \end{bmatrix} = \sum_{i=1}^{N_1} x_{s\_i} \cdot w_{s\_i} \tag{21}$$

Among them, W_{sector} is the weight vector of the sectorial area mentioned above. With the weight values, the closer to the center, the greater the weight, which means that the obstacle or boundary distance corresponding to the position has a greater impact on the driving of the ego vehicle. To obtain a large reward value, the ego vehicle tends to approach the driving state in which there are no obstacles and boundaries in the forward direction as much as possible. Similarly, the reward function in the rectangular area is as follows: $W_{rectangle}$ represents the weight vector of the rectangular area mentioned above. Different from W_{sector}, $W_{rectangle}$ represents the weight vector of the ego vehicle's forward direction. As long as an external obstacle appears anywhere within the area, the vehicle will collide with the obstacle or exceed the lane boundary if it continues to go straight forward. Therefore, the weight of this area does now show much difference. There is little difference between the weights in this area.

$$r_{rectangle} = (S_{sector} - B_{sector}) \cdot W_{sector}^T = \begin{bmatrix} x_{r_1} & x_{r_2} & \cdots & x_{r_{N_1}} \end{bmatrix} \cdot \begin{bmatrix} w_{r_1} \\ w_{r_2} \\ \cdots \\ w_{r_3} \end{bmatrix} = \sum_{i=1}^{N_1} x_{r\_i} \cdot w_{r\_i} \tag{22}$$

In this obstacle representation method, the detection range is designed for different numbers of obstacles, and the input state of obstacles is unified, which is convenient to generate more complex data in the training process. Additionally, this method is used to unify the obstacle vehicles that need to be considered at each moment and the lane boundary lines on both sides into an obstacle set, which simplifies the separate discussion of obstacles and lane boundary modules in the obstacle avoidance process. This part of the

reward replaces the reward section $r_{ego\_boundary}$ and r_{ego_obs}, and the rest is the same as the previous section, which can be described as follows,

$$\begin{aligned} r_{ego\_obs} &= r_{sector} + r_{rectangle} \\ r_t &= r_{ego\_obs} + r_{ego\_other} \end{aligned} \tag{23}$$

3.3. Simulation Results and Analysis

In the training process of the improved obstacle avoidance method, we enhance the environment complexity in the training process of the reinforcement learning neural network, which is embodied as follows: We have increased the randomness of the obstacle position and the randomness of the obstacle vehicle direction. In the obstacle environment of each episode, the scene of two vehicles side by side often occurs, resulting in a narrow traversable area at this location, and higher requirements for the attitudes of passing vehicles near this location. As shown in Figure 11, in these scenarios, under the model of the obstacle input state and reward function designed with the traditional method, the success rate of the ego vehicle passing through the area is low.

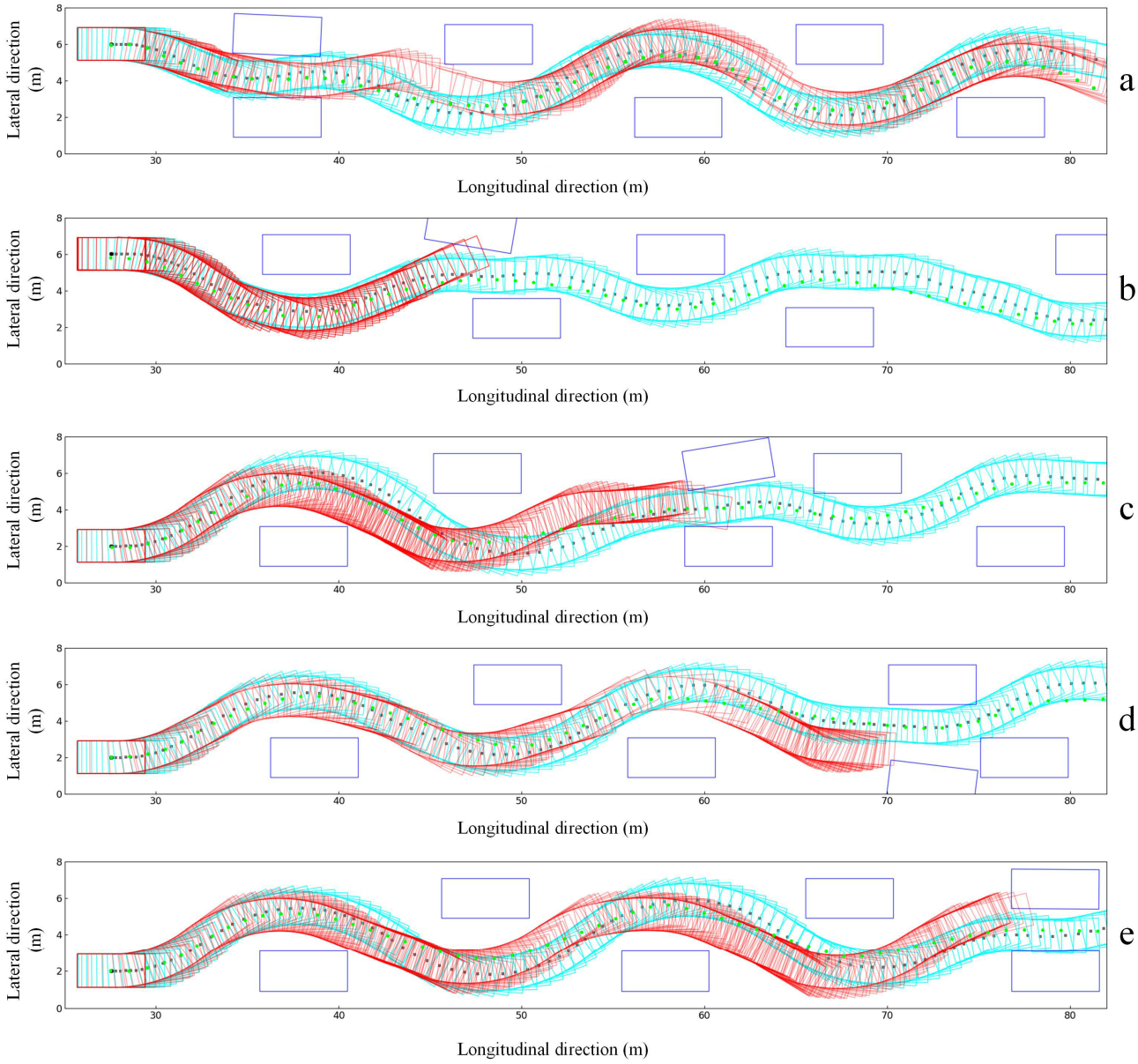

Figure 11. Trajectory comparison of three obstacle avoidance methods (**a**–**e**).

In order to compare with traditional classical methods, we also add the trajectory of the lattice planner into the same scenario. The lattice planner is a graph-based approach to the path planning issue that reduces the search space into a uniform discretization of vertices corresponding to positions and headings. It generates multiple obstacle avoidance trajectories, and then selects the most suitable one according to the cost ranking. The algorithm has good performance in both real vehicle and simulated obstacle avoidance scenes.

The trajectory comparison of the three obstacle avoidance methods is presented in Figure 11. The dark blue rectangle is obstacle vehicles on the road, and they are distributed in a random state within a specific range. The cyan one refers to the driving trajectory of the ego vehicle frame using the multi-obstacle input method, and the black points are the center locations of the ego vehicle during the whole episode. The green points represent the final trajectory points generated by the lattice planner. The red rectangle is the bounding trajectories of the ego vehicle using the reinforcement learning method with the traditional obstacle representation method. Specifically, in the first obstacle scene in the series, the side-by-side vehicles appear next to the first vehicle. Although the three obstacle avoidance methods all have passed the obstacle field in the simulation verification, the driving trajectories in the narrow area are slightly different. Similar to the traditional method lattice planner, the trajectory using the obstacle avoidance method with the obstacle state representation in Section 3 is smoother when passing through narrow areas, and the obstacle avoidance effect is more stable when compared with the ordinary obstacle avoidance method. It can be seen from the obstacle scenes in the series diagram that the ego vehicle, based on the common obstacle avoidance RL method, collided with obstacle vehicles. However, the obstacle avoidance method in Section 3 can still pass in a relatively stable state even though the traversable area is narrow.

Table 1 displays the statistics of the obstacle avoidance performance parameters of three methods. It is found that both our work and the lattice planner have better performance, but the lattice planner consumes a longer time per step and is more unstable. Although the common RL algorithm takes the shortest time and is relatively stable, it has limited ability of passing through complex obstacles. Our improved algorithm has the best and most stable passing performance, taking a relatively short time with the highest success rate.

Table 1. Statistics of obstacle avoidance performance parameters of three methods under random difficulty conditions.

Algorithm	Our Work	Common RL	Lattice Planner
Average success rate (%)	95	70	92
Average calculation time (ms)	3.42	2.96	35.41
Average time standard variance	0.37	0.31	10.48

From the results and obstacle avoidance trajectories of the above three obstacle avoidance methods, it can be seen that the improved multi-obstacle environment state and reward function design method can uniformly process the external obstacles with random numbers and attitudes so that the ego vehicle can better pass obstacle areas in complex obstacle scenarios.

4. Conclusions

In this paper, we first adopted the reinforcement learning method TD3 as the primary network, which was convenient for us to further adjust the relevant reward function in time according to the simulation training performance. The task scenarios of autonomous driving following and avoiding obstacles using the map constraints were investigated. For the following scenarios with turning roads, the state function and the corresponding reward function were designed in combination with the waypoint information in the map in the Frenet coordinate system. After 10,000 episodes of simulation training, the ego vehicle's following performances on straights and corners reached a stable state.

Aiming at the multi-obstacle avoidance scene under the map constraints, we then proposed a representation method that could represent obstacles of different numbers and shapes. Combined with the obstacle representation method and the reinforcement learning model in the vehicle following scenario, the relevant reward function of the obstacle avoidance scene was designed. After a period of training, some obstacle scenes with narrow passable areas were constructed on the map. To verify the effect of two reinforcement learning methods, we compared them with the classic lattice planner. The results indicate that the obstacle representation method proposed by this paper has a better performance. In future work, we consider integrating the two single-task models into a multi-task reinforcement learning model.

In addition, multi-agent collaboration is also another research endpoint. We plan to focus on the reinforcement learning network model based on multi-agent collaboration using the map constraints. The cooperation among multi-agents is more in line with the future direction of intelligent networked vehicles.

Author Contributions: Validation, X.Y.; Writing—original draft, Z.L.; Writing—review & editing, S.Y.; Supervision, S.T.; Project administration, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by Natural Science Foundation of Chongqing, China under cstc2021jcyj-msxmX0792.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data are not publicly available due to the nature of our laboratory for defense.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Moser, D.; Schmied, R.; Waschl, H.; del Re, L. Flexible Spacing Adaptive Cruise Control Using Stochastic Model Predictive Control. *IEEE Trans. Control. Syst. Technol.* **2017**, *26*, 114–127. [CrossRef]
2. Ioannou, P.; Xu, Z.; Eckert, S.; Clemons, D.; Sieja, T. Intelligent cruise control theory and experiment. In Proceedings of the 32nd IEEE Conference on Decision and Control, San Antonio, TX, USA, 15–17 December 1993.
3. Kitazono, S.; Ohmori, H. Semi-Autonomous Adaptive Cruise Control in Mixed Traffic. In Proceedings of the 2006 SICE-ICASE International Joint Conference, Busan, Republic of Korea, 18–21 October 2006; pp. 3240–3245. [CrossRef]
4. Choi, S.B.; Hedrick, J.K. Vehicle longitudinal control using an adaptive observer for automated highway systems. In Proceedings of the 1995 American Control Conference—ACC'95, Seattle, WA, USA, 21–23 June 1995.
5. Luo, J.; Wang, Z.-X.; Pan, K.-L. Reliable Path Planning Algorithm Based on Improved Artificial Potential Field Method. *IEEE Access* **2022**, *10*, 108276–108284. [CrossRef]
6. Xie, Z.; Wu, Y.; Gao, J.; Song, C.; Chai, W.; Xi, J. Emergency obstacle avoidance system of driverless vehicle based on model predictive control. In Proceedings of the 2021 International Conference on Advanced Mechatronic Systems (ICAMechS), Tokyo, Japan, 9–12 December 2021; pp. 22–27. [CrossRef]
7. Zhang, X.; Zhang, W.; Zhao, Y.; Wang, H.; Lin, F.; Cai, Y. Personalized Motion Planning and Tracking Control for Autonomous Vehicles Obstacle Avoidance. *IEEE Trans. Veh. Technol.* **2022**, *71*, 4733–4747. [CrossRef]
8. Yang, S.; Lin, Y. Development of an Improved Rapidly Exploring Random Trees Algorithm for Static Obstacle Avoidance in Autonomous Vehicles. *Sensors* **2021**, *21*, 2244. [CrossRef] [PubMed]
9. Zhang, X.; Zhu, T.; Du, L.; Hu, Y.; Liu, H. Local Path Planning of Autonomous Vehicle Based on an Improved Heuristic Bi-RRT Algorithm in Dynamic Obstacle Avoidance Environment. *Sensors* **2022**, *22*, 7968. [CrossRef] [PubMed]
10. Sallab, A.E.; Abdou, M.; Perot, E.; Yogamani, S. Deep Reinforcement Learning framework for Autonomous Driving. *Electron. Imaging* **2017**, *2017*, 70–76. [CrossRef]
11. Chen, J.; Yuan, B.; Tomizuka, M. Model-free Deep Reinforcement Learning for Urban Autonomous Driving. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019.
12. Gao, H.; Su, H.; Cai, Y.; Wu, R.; Hao, Z.; Xu, Y.; Wu, W.; Wang, J.; Li, Z.; Kan, Z. Trajectory prediction of cyclist based on dynamic Bayesian network and long short-term memory model at unsignalized intersections. *Sci. China Inf. Sci.* **2021**, *64*, 1–13. [CrossRef]
13. Nishitani, I.; Yang, H.; Guo, R.; Keshavamurthy, S.; Oguchi, K. Deep Merging: Vehicle Merging Controller Based on Deep Reinforcement Learning with Embedding Network. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May 2020–31 August 2020.

14. Desjardins, C.; Chaib-Draa, B. Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* **2011**, *12*, 1248–1260. [CrossRef]
15. Gao, A.; Wang, Q.; Liang, W.; Ding, Z. Game Combined Multi-Agent Reinforcement Learning Approach for UAV Assisted Offloading. *IEEE Trans. Veh. Technol.* **2021**, *70*, 12888–12901. [CrossRef]
16. Li, Z. A Hierarchical Autonomous Driving Framework Combining Reinforcement Learning and Imitation Learning. In Proceedings of the 2021 International Conference on Computer Engineering and Application (ICCEA), Kunming, China, 25–27 June 2021; pp. 395–400.
17. Jamshidi, F.; Zhang, L.; Nezhadalinaei, F. Autonomous Driving Systems: Developing an Approach based on A and Double Q-Learning. In Proceedings of the 2021 7th International Conference on Web Research (ICWR), Tehran, Iran, 19–20 May 2021; pp. 82–85.
18. Josef, S.; Degani, A. Deep Reinforcement Learning for Safe Local Planning of a Ground Vehicle in Unknown Rough Terrain. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6748–6755. [CrossRef]
19. Meyer, E.; Robinson, H.; Rasheed, A.; San, O. Taming an Autonomous Surface Vehicle for Path Following and Collision Avoidance Using Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 41466–41481. [CrossRef]
20. Liu, X.; Liu, Y.; Chen, Y.; Hanzo, L. Enhancing the Fuel-Economy of V2I-Assisted Autonomous Driving: A Reinforcement Learning Approach. *IEEE Trans. Veh. Technol.* **2020**, *69*, 8329–8342. [CrossRef]
21. Zhang, Q.; Pan, W.; Reppa, V. Model-Reference Reinforcement Learning for Collision-Free Tracking Control of Autonomous Surface Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 8770–8781. [CrossRef]
22. Wang, H.; Gao, H.; Yuan, S.; Zhao, H.; Wang, K.; Wang, X.; Li, K.; Li, D. Interpretable Decision-Making for Autonomous Vehicles at Highway On-Ramps With Latent Space Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2021**, *70*, 8707–8719. [CrossRef]
23. Li, Z.; Zhou, J.; Li, X.; Du, X.; Wang, L.; Wang, Y. Continuous Control for Moving Object Tracking of Unmanned Skid-Steered Vehicle Based on Reinforcement Learning. In Proceedings of the 2020 3rd IEEE International Conference on Unmanned Systems (ICUS) and the organizing committee, Harbin, China, 27–28 November 2020.