sensors

Article

Reinforcement Learning-Based End-to-End Parking for Automatic Parking System

Peizhi Zhang [1], Lu Xiong [1,*], Zhuoping Yu [1], Peiyuan Fang [1], Senwei Yan [2], Jie Yao [2] and Yi Zhou [2]

[1] School of Automotive Studies, Tongji University, Shanghai 201804, China; zhangpeizhi@tongji.edu.cn (P.Z.); yuzhuoping@tongji.edu.cn (Z.Y.); funpayyuan@foxmail.com (P.F.)

[2] SAIC Motor Corporation Limited, Shanghai 201800, China; adam_yan@foxmail.com (S.Y.); yaojie@saicmotor.com (J.Y.); zhouyi02@saicmotor.com (Y.Z.)

* Correspondence: xiong_lu@tongji.edu.cn

Received: 31 July 2019; Accepted: 12 September 2019; Published: 16 September 2019

Abstract: According to the existing mainstream automatic parking system (APS), a parking path is first planned based on the parking slot detected by the sensors. Subsequently, the path tracking module guides the vehicle to track the planned parking path. However, since the vehicle is non-linear dynamic, path tracking error inevitably occurs, leading to inclination and deviation of the parking. Accordingly, in this paper, a reinforcement learning-based end-to-end parking algorithm is proposed to achieve automatic parking. The vehicle can continuously learn and accumulate experience from numerous parking attempts and then learn the command of the optimal steering wheel angle at different parking slots. Based on this end-to-end parking, errors caused by path tracking can be avoided. Moreover, to ensure that the parking slot can be obtained continuously in the process of learning, a parking slot tracking algorithm is proposed based on the combination of vision and vehicle chassis information. Furthermore, given that the learning network output is hard to converge, and it is easy to fall into local optimum during the parking process, several reinforcement learning training methods in terms of parking conditions are developed. Lastly, by the real vehicle test, it is proved that using the proposed method can achieve a better parking attitude than using the path planning and path tracking-based method.

Keywords: automatic parking system (APS); end-to-end parking; reinforcement learning; parking slot tracking

1. Introduction

The average proportion of cars and parking slots in big cities is about 1:0.8, and that in small and medium-sized cities is nearly 1:0.5, according to the data released by the National Development and Reform Commission of China. The lack of parking space makes the designed parking slot increasingly narrower. Accordingly, parking environment is becoming complex progressively, and the increasingly higher requirement of the parking operation accuracy is raised, bringing great troubles to many drivers. Automatic parking system (APS) can increase parking safety and utilization rate of parking slot, so it has wide market application prospects.

However, the smaller size of the parking slot requires very high parking accuracy for APS. Take the perpendicular parking slot as an example; it raises a higher demand of parking attitude for its narrow width. The BS ISO 16787-2016 [1] stipulates that the perpendicular parking inclination angle of APS should be confined within $\pm 3^\circ$, imposing huge challenges to the performance of APS.

The current mainstream APS architecture is a path planning and path tracking-based method. To be specific, a parking path is first planned based on the parking slot detected by the sensors (e.g., camera and ultrasonic radar), and then the path tracking module controls the vehicle to track the planned parking path. However, since the vehicle is nonlinear dynamic, the control error of path tracking is inevitable during the path tracking, leading to inappropriate parking attitude.

How can we avoid the path tracking error to ensure the ideal parking attitude? Let us think about how humans park their cars. Actually, we directly turn the steering wheel according to the position of the parking slot, which is an end-to-end parking mode. Moreover, as the number of parking increases, we gain more experience, and parking is increasingly accurate. In fact, reinforcement learning is an algorithm in which the agent gets the greatest reward in the process of interactive learning with the environment, thus learning the optimal mapping from environment to action. Therefore, we try to apply reinforcement learning to APS to improve the parking attitude.

1.1. Related Work

1.1.1. Mainstream APS

The mainstream APS first plans a parking path according to the parking slot detected by sensors. Path planning can be split into geometric method, sampling method, and numerical optimization method. The geometric method adopts Reeds–Shepp (RS) curve [2], B-spline curve [3], η^3-splines [4], and arcs optimized by cyclotron curve [5,6] to plan parking path based on the non-holonomic constraints of the vehicle. The sampling method aims to spread the points evenly in the sampling space, filter the points by a certain method, and connect the selected points into the required path, covering Rapidly-exploring Random Tree (RRT) [7] and target bias RRT [8]. The numerical optimization method is to consider the parking process as a dynamic system, and the length [9,10] or curvature [11,12] of the parking path is the optimization goal of this dynamic system. The constraints of this dynamic system include the non-holonomic constraints of the vehicle, the starting point, and the target location of the parking.

After completing the path planning, the path tracking module of APS controls the vehicle to track the planned parking path. Path tracking can be divided into Ackerman steering model-based open loop control method and vehicle dynamics model-based closed loop control method. The Ackerman steering model-based open loop control method considers that there is no tire sideslip, and vehicle satisfies the non-holonomic constraints. The most typical one is the pure tracking control algorithm [13]. The vehicle dynamics model-based closed loop control method considers tire sideslip. Feedforward control is designed using the two-degree-of-freedom vehicle dynamics model, and closed-loop feedback control is implemented by proportional-integral-differential (PID) algorithm [14,15] or sliding mode control (SMC) algorithm [16,17]. In fact, no matter which control method is used, the control error of the path tracking is inevitable since the vehicle is nonlinear dynamic [18,19], which makes the vehicle unlikely to completely track the planned parking path. Though there have been studies to reduce the path tracking error [20,21], the path tracking error cannot be eliminated.

1.1.2. Reinforcement Learning

As mentioned above, path planning and path tracking-based method may result in poor parking attitude due to the inevitable control error (the experimental results in Section 3 also confirmed this). Accordingly, we take the "human-like" parking mode based on reinforcement learning, which cannot only avoid the error caused by path tracking through the end-to-end method of environment-to-action but also continuously learn and accumulate experience from considerable parking attempts, as well as learning the optimal steering wheel angle command at different parking slots relative to vehicle. How to choose a suitable reinforcement learning method for APS? To answer this question, different reinforcement learning methods are first reviewed.

Reinforcement learning mainly includes value-based method, policy-based method, and Actor-Critic method.

The value-based method evaluates the cumulative expectation reward by the value function after taking action and then chooses the action with the largest cumulative reward expectation [22]. Deep Q network (DQN) [23,24] is a typical value-based method. It is based on Q-learning and replaces Q-table with deep neural network (DNN) to solve the problem that Q-learning is prone to dimension disasters when state space is high-dimensional. However, value-based method makes value function continuous and chooses action based on the value function of each action, so it is not suitable for continuous action spaces (e.g., continuous steering wheel angle command for APS).

Compared with the value-based method, the policy-based method directly optimizes the policy based on the sampling method and constantly calculates the gradient of the policy expectation reward about the parameters of the policy network during the training process [25]. Though the policy-based method is applicable to high-dimensional continuous action spaces, each iterative step should sample a batch sequence to update the parameters, resulting in a large variance of the policy gradient estimation and making it easy to fall into local optimum.

The Actor-Critic method, which combines the value-based method and the policy-based method, adopts policy-based method to update the policy, and adopts the value function as the evaluation method of the policy [26–28]. By introducing the value function as the evaluation criterion in the policy search, the loss of sequential difference about the reward can be minimized, so that the variance of the policy gradient estimation can be reduced effectively. Although Actor-Critic method can realize the learning of continuous action space and can reduce the variance of the strategy gradient estimation, it only has one actor network and one critic network, which easily leads to unstable training. Deep Deterministic Policy Gradient (DDPG) algorithm [29,30] has made some improvements on the basis of Actor-Critic method. On one hand, it creates target networks for actor network and critic network, respectively, significantly enhancing the stability of learning. On the other hand, it uses experience pool-based replay caching technology to cut off the data correlation.

As mentioned above, we believe that DDPG is applicable to APS for the following reasons: first, Actor-Critic architecture can realize the learning of continuous action space (since the steering wheel angle for APS is a continuous action). Second, introducing the value function as the evaluation criterion in the policy search can reduce the variance of the policy gradient estimation, which is more efficient. Lastly, DDPG creates target networks for actor network and critic network, respectively, which makes it closer to the supervised learning and significantly enhance the stability of learning.

1.2. Objectives and Contributions

In brief, the current path planning and path tracking-based method cannot easily ensure the ideal parking attitude, especially the perpendicular parking slot, for its narrow width, which requires a higher demand for parking. To solve this problem, a reinforcement learning-based end-to-end parking algorithm is proposed in this paper for perpendicular parking. The main contributions are as follows:

- We innovatively apply DDPG to perpendicular parking so that the vehicle can continuously learn and accumulate experience from considerable parking attempts, learn the optimal steering wheel angle command at different parking slots relative to vehicle, as well as achieve the real "human-like" intelligent parking. Moreover, because it realizes the end-to-end control from the parking slot to the steering wheel angle command, the control errors caused by path tracking are fundamentally avoided;
- Since the parking slot needs to be continuously obtained in the course of learning, we propose a parking slot tracking algorithm, which uses extended Kalman filter (EKF) to fuse the parking slot information with vehicle chassis information to achieve continuous tracking of parking slot;
- Given that the learning network output is hard to converge and it is easy to fall into local optimum in the parking process, several reinforcement learning training methods in terms of parking conditions, e.g., manual guided exploration for accumulating initial experience sequence, control cycle phased setting, and training condition phased setting, are designed. Besides, the well-trained network in the simulation environment is migrated to the real vehicle training.

1.3. Paper Outline

The rest of this paper is organized as follows. In Section 2, our reinforcement learning-based end-to-end parking method is introduced. In Section 3, the experimental results are showed. In Section 4, some discussions are contained. Lastly, this paper is concluded in Section 5.

2. Method

The overview of the proposed method is shown in Figure 1. It primarily includes two modules, parking slot tracking and reinforcement learning-based planning. Parking slot tracking is used to provide continuous position of parking slot for reinforcement learning, and reinforcement learning is adopted to achieve end-to-end planning from the parking slot to steering wheel angle.

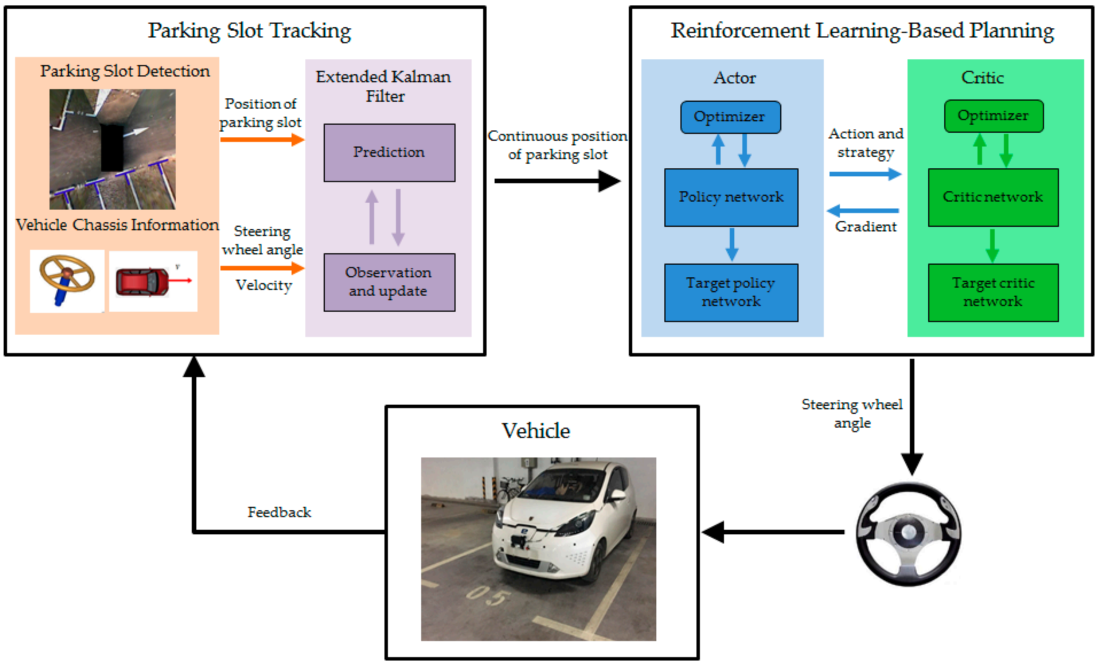


Figure 1. Overview of the reinforcement learning-based end-to-end parking method.

2.1. Parking Slot Tracking

In this section, the parking slot detection is first introduced, followed by the EKF-based parking slot tracking.

2.1.1. Parking Slot Detection

The sensors of surround view parking slot detection system are outfitted with four fisheye cameras in the front, rear, left, and right positions of the vehicle, respectively, with 180° of FOV horizontally and 140° of FOV vertically, as shown in Figure 2a.

The parking slot detection consists of two steps: one is to yield a surround view based on the images taken by the four fisheye cameras; the other is to detect the corner points of parking slots using the surround view. The flow chart is shown in Figure 2b.

For the generation of surround view, first, the distortion parameters of fisheye camera are calculated by Zhang Zhengyou's calibration method [31], and the mapping table T_{UF} from undistorted image coordinate system (CS) to fisheye image CS is yielded. Subsequently, based on the checkerboard calibration site, the homography matrix M_{VU} from vehicle CS to undistorted image CS is calculated using the least square method. Lastly, after confirming the scope and the image size of the surround view, the similarity transformation matrix M_{SV} of surround view CS to vehicle CS is calculated. Four fisheye perspectives are joined into one surround view by the comprehensive mapping table T_{BF} constructed above. The whole process is illustrated in Figure 3.

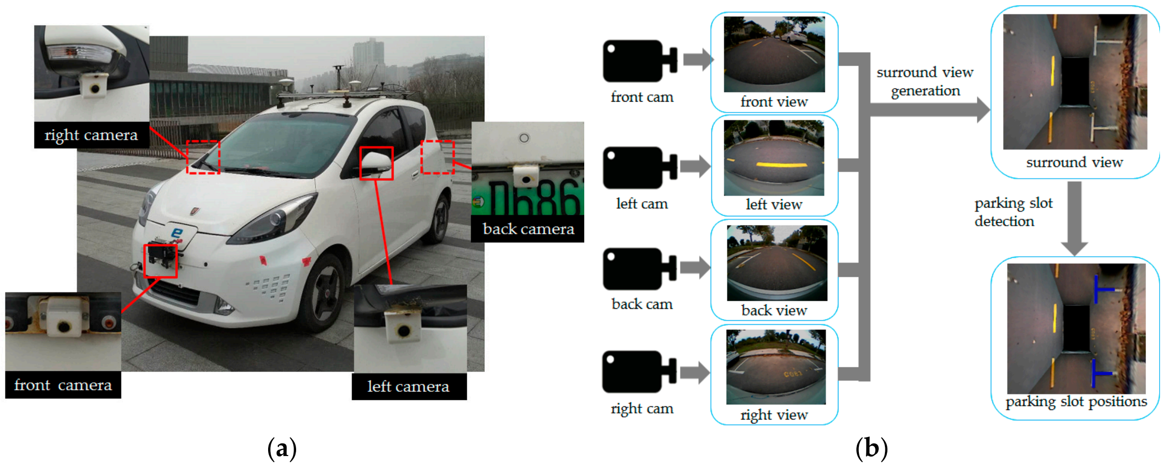


Figure 2. Surround view parking slot detection system. (**a**) Test vehicle and camera installation location. (**b**) Surround view generation and parking slot detection.

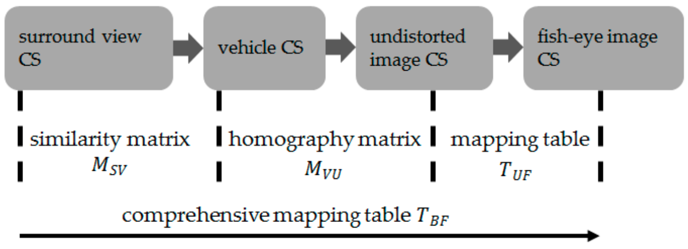

surround view CS
vehicle CS
undistorted image CS
fish-eye image CS
similarity matrix M_{SV}
homography matrix M_{VU}
mapping table T_{UF}
comprehensive mapping table T_{BF}

Figure 3. Surround view generation process.

The method proposed by Li et al. [32] is adopted to detect parking slot, which is an AdaBoost-based slot detection method that detects parking slots from surround view. This method is primarily used to detect common "L" and "T" corner points, as shown in Figure 4. The basic principle is to use Adaboost algorithm and decision tree to design a binary classifier to detect corner point patterns. The input of the classifier refers to an image patch, and the output is a Boolean value, indicating whether the input local block is a corner pattern. Because of the limited FOV of surround view, the length of the parking slot can be inferred following some prior rules after the detection of the corner points.

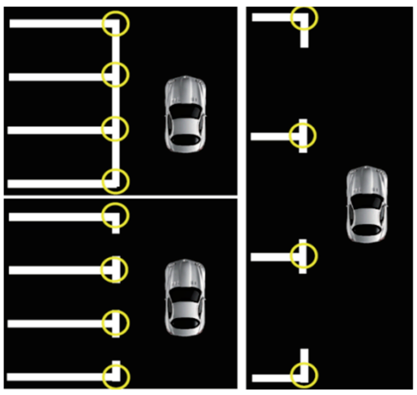


Figure 4. The "L" and "T" corner points.

The above analysis reveals that when detecting the corner points, the parking slot relative to the vehicle can be calculated through coordinate transformation. However, we find that the parking slot is difficult to continuously identify by relying solely on the vision. For instance, the corner points of the parking slot are sometimes not detected during parking due to image distortion, illumination change, occlusion, as well as the limited FOV, as shown in Figure 5.

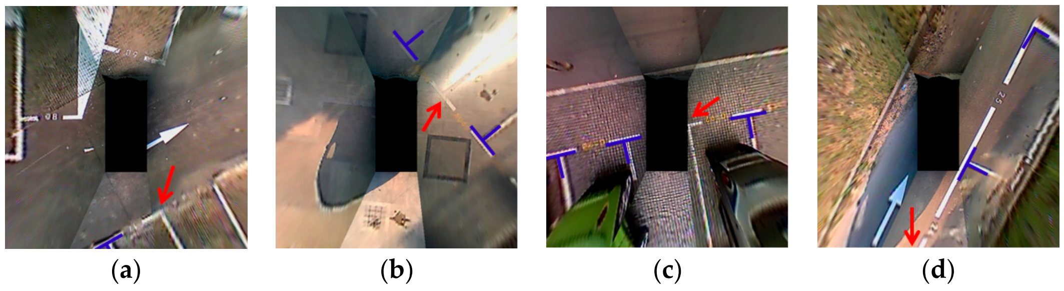


Figure 5. Missed parking slot detections (indicated by red arrows). (**a**) Image distortion. (**b**) Illumination change. (**c**) Occlusion. (**d**) Limited FOV.

2.1.2. EKF-Based Parking Slot Tracking

To track parking slot continuously and accurately, EKF is employed to achieve the fusion of vision and vehicle chassis information. First, we take the position of the corner point of parking slot relative to vehicle as the EKF's observation, building the constraint relationship between the position of vehicle and parking slot in the reference CS, i.e., the EKF's observation model. Second, the vehicle kinematics model is taken as the EKF's motion model, and the steering wheel angle and velocity obtained from vehicle chassis act as the EKF's control input. Lastly, based on EKF's "prediction" and "update" process, the fusion is completed to achieve the maximum posterior estimation of parking slot in the presence of noise.

The definition of CS and parameters is shown in Figure 6, where (x, y), φ and (x_i, y_i) are the vehicle coordinates, vehicle heading angle and the i th corner points (i = 1 and 2 represent the left and right corner point of parking slot, respectively) coordinates in the reference CS, respectively. (x_{vi}, y_{vi}) obtained by parking slot detection system in Section 2.1.1 denotes the coordinates of the i th corner point in the vehicle CS. Its distance from the center of the rear axle of the vehicle is r_i, and its angle relative to the axis of the vehicle CS is θ_i. r_i and θ_i can be derived from the Equation (1).

$$\begin{aligned} \theta_i(k) &= \arctan\left(\frac{y_{vi}(k)}{x_{vi}(k)}\right) \\ r_i(k) &= \sqrt{x_{vi}(k)^2 + y_{vi}(k)^2} \end{aligned} \quad (1)$$

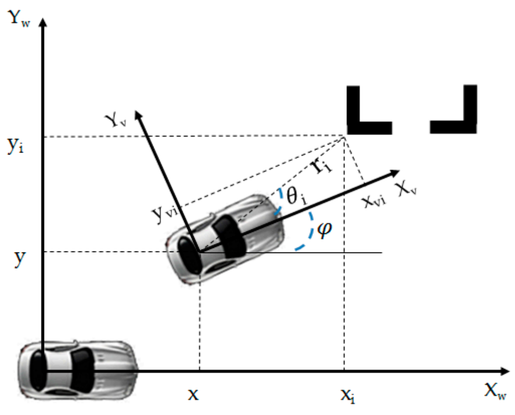


Figure 6. CS and parameter definition.

The state variable of EKF is $X = (x, y, \varphi, x_i, y_i)^T$, and its covariance matrix of the error is denoted as P. The observed variable of the system is expressed as $Z = (r_i, \theta_i)^T$.

The discrete observation model is expressed as Equation (2),

$$\begin{gathered} Z(k) = h(X(k)) + \vartheta(k) \\ \left[\sqrt{\begin{array}{c} (x_i(k) - x(k))^2 + (y_i(k) - y(k))^2 \\ \arctan\left(\frac{y_i(k) - y(k)}{x_i(k) - x(k)}\right) - \varphi(k) \end{array}} \right] + \vartheta(k) \end{gathered} \tag{2}$$

where $\vartheta(k)$ denotes the noise of parking slot detection system, assuming a Gaussian distribution; its covariance matrix is R.

According to the Ackerman steering model of the vehicle, the discrete motion model can be expressed as Equation (3),

$$\begin{aligned} X(k) &= f(X(k-1), U(k)) + w(k) \\ &= \begin{bmatrix} x(k-1) + Tv(k)cos\varphi(k) \\ y(k-1) + Tv(k)sin\varphi(k) \\ \varphi(k-1) + \frac{Tv(k)\tan(\delta(k)/i_0)}{L} \\ x_i(k-1) \\ y_i(k-1) \end{bmatrix} + w(k) \end{aligned} \tag{3}$$

where δ denotes the steer wheel angle; v the velocity; δ and v can be obtained directly from the vehicle chassis; i_0 the steering gear ratio; L the wheelbase; T the period; $w(k)$ the noise of the motion model, assumed to be Gaussian noise, and its covariance matrix is expressed as Q.

EKF can be split into two steps (prediction and update). First, the system state and its error covariance matrix at the k th iteration time are predicted, as expressed in Equation (4),

$$\begin{aligned} \hat{X}(k)^- &= f\left(\hat{X}(k-1), U(k)\right) \\ P(k)^- &= F(k)P(k-1)F(k)^T + Q \end{aligned} \tag{4}$$

where $F(k)$ denotes the Jacobian of function $f(X(k), U(k))$ with respect to $X(k)$.

Subsequently, it is the update process. First, the Kalman gain $K(k)$ is calculated, which is the key to the maximum posteriori estimation of $X(k)$ in the presence of noise, as shown in Equation (5). Second, $X(k)$ and $P(k)$ are updated by $K(k)$, as expressed in Equation (6),

$$K(k) = P(k)^- H(k)^T \left(H(k)P(k)^- H(k)^T + R\right)^{-1} \tag{5}$$

$$\begin{aligned} \hat{X}(k) &= \hat{X}(k)^- + K(k)\left[Z(k) - H(k)\hat{X}(k)^-\right] \\ P(k) &= (I - K(k)H(k))P(k)^- \end{aligned} \tag{6}$$

where $H(k)$ is the Jacobian of function $h(X(k))$ with respect to $X(k)$.

Equation (6) reveals that $K(k)$ helps fuse the vision and vehicle chassis information, and the updated $\hat{X}(k)$ is calculated by $K(k)$ in the presence of noise, satisfying the maximum posterior estimate of $X(k)$. Accordingly, EKF-based fusion is more accurate than relying solely on vision detection. Moreover, $X(k)$ is continuous because the vehicle chassis information continues to be inputted. Thus, the continuous and accurate position of the parking slot relative to the vehicle can be derived from the above.

2.2. Reinforcement Learning-Based Planning

In this section, reinforcement learning is adopted to achieve end-to-end planning from the parking slot to steering wheel angle. We first introduce the appropriate reinforcement learning model for APS, followed by the system settings and training process of DDPG, and finally the improved training measures applied in parking.

2.2.1. Appropriate Reinforcement Learning Model for APS

The basic process of reinforcement learning is a Markov decision-making process, which can be expressed by the quaternion $\{S, A, P, R\}$ composed of state S, action A, state transition probability P and reward R.

When a policy π is executed at time t, cumulative reward G_t can be calculated:

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots = \sum_{k=0} \gamma^k R_{t+k+1} \tag{7}$$

where γ is the discount factor, which is used to reduce the reward weight corresponding to the long-term decision.

The action value function $Q_\pi(s, a)$ is the expectation of the cumulative reward G_t after taking action a at the current state s, as expressed in Equation (8).

$$Q_\pi(s, a) = E_\pi[G_t \middle| S_t = s, A_t = a] \tag{8}$$

The action valued function $Q_\pi(s, a)$ satisfies the Bellman equation (Equation (9)), which transforms the solution of $Q_\pi(s, a)$ into an iterative process of dynamic programming.

$$Q_\pi(s, a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \middle| S_t = s, A_t = a] \tag{9}$$

The goal of reinforcement learning is to find an optimal policy for obtaining the maximum $Q_*(s, a)$, as shown in the following equation.

$$Q_*(s, a) = \underset{\pi}{max} Q_\pi(s, a) \tag{10}$$

According to the different optimization objects, reinforcement learning methods can be divided into value-based method, policy-based method, and Actor-Critic method.

- Value-based method

Q-learning is a basic value-based method. Q-learning first chooses action a according to Q value at the current state s in each step of the cycle (e.g., using $\varepsilon - \mathit{greedy}$ method: $1 - \varepsilon$ probability of selecting action $\underset{a}{\operatorname{argmax}} Q(s, a)$, ε probability of randomly selecting action). After the selected action is taken, the immediate reward R and the next state s' are observed, and then $Q(s, a)$ is updated, as expressed in Equation (11). Repeat the process until the final state is reached,

$$Q(s, a) \leftarrow Q(s, a) + \alpha\Big[R + \gamma \max_a Q(s', a) - Q(s, a)\Big] \tag{11}$$

where α is the learning rate.

DQN replaces Q-table with DNN with parameter w (Equation (12)) to solve the problem that Q-learning is prone to dimension disasters when state space is high-dimensional.

$$Q_w(s, a) \approx Q(s, a) \tag{12}$$

The updating objective of $Q_w(s, a)$ is to minimize the mean square deviation of the objective value $Q_\pi(s, a)$ and the actual value $Q_w(s, a)$, as shown in Equation (13). If gradient descent method is used, the gradient $\nabla_w J(w)$ of the objective function $J(w)$ relative to the parameter w is first calculated, and then the parameter w changes along the opposite direction of the gradient $\nabla_w J(w)$, as shown in Equation (14),

$$J(w) = E_\pi\Big[(Q_\pi(s,a) - Q_w(s,a))^2\Big] \tag{13}$$

$$w \leftarrow w + \alpha\nabla_w J(w) = w + \alpha[R + \gamma Q_w(s', a') - Q_w(s,a)]\nabla Q_w(s,a) \tag{14}$$

where $Q_\pi(s,a)$ can be solved by temporal-difference method, i.e., $Q_\pi(s,a) = R + \gamma Q_w(s', a')$.

Since the action space of the value-based method is discrete, it is not suitable for the continuous action space of parking control. Though the continuous action space can be discretized, too large discrete spacing will lead to the algorithm not getting the optimal action, and too small discrete spacing will lead to dimension disaster.

- Policy-based method

The policy-based method directly optimizes the policy based on the sampling method, and constantly calculates the gradient $\nabla_\theta J(\theta)$ of the policy expectation reward $J(\theta)$ about the policy parameter θ during the training process, as expressed in Equations (15) and (16),

$$J(\theta) = E_{\pi_\theta}[R] = \sum_{s\in S} d^\pi(s) \sum_{a\in A} \pi_\theta(s,a) R_{s,a} \tag{15}$$

$$\nabla_\theta J(\theta) = \sum_{s\in S} d^\pi(s) \sum_{a\in A} \pi_\theta(s,a)\nabla_\theta log\pi_\theta(s,a) R_{s,a} = E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a) R_{s,a}] \tag{16}$$

where $\pi_\theta(s,a)$ is the policy of action selection, which represents the probability of choosing action a under the state s; $d^\pi(s)$ is the static distribution of the state s under the policy π.

If the Monte Carlo policy gradient algorithm is used, the iteration equation of the policy parameter θ is as follows:

$$\theta \leftarrow \theta + \alpha\nabla_\theta\, log\pi_\theta(s,a)v \tag{17}$$

where v is equal to the cumulative reward G_t of the current step minus the average cumulative reward $(1/T)\sum_{t=1}^{T} G_t$, i.e., if the action gets better evaluation than before and v is positive, it will increase the probability of this action being selected.

Though the policy-based method is applicable to high-dimensional continuous action spaces, each iterative step should sample a batch sequence to update the parameters, resulting in a large variance of the policy gradient estimation and making it easy to fall into local optimum.

- Actor-Critic method

The Actor-Critic method, which combines the value-based method and the policy-based method, consists of two updating processes: The critic network is responsible for updating the network parameters of the action value function, observing the action and reward, and evaluating the policy. The actor network is responsible for updating the actor network parameters according to the guidance of the critic networks. By introducing the value function as the evaluation criterion in the policy search, the loss of sequential difference about the reward can be minimized so that the variance of the policy gradient estimation can be reduced effectively.

Although Actor-Critic method can realize the learning of continuous action space and can reduce the variance of the policy gradient estimation, it only has one policy network and one critic network, which easily leads to unstable training.

In order to solve this problem, DDPG constructs the target network with parameter θ', which is used to calculate the target value. The target network is adopted to track the actor network and critic network slowly to update the parameter θ', as expressed in Equation (18). This means that the target value is limited to slow change, which greatly improves the stability of learning. This improvement brings the reinforcement learning closer to supervised learning.

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta', \tau \leq 1 \tag{18}$$

In addition, a challenge in reinforcement learning using neural networks is that most optimization algorithms assume that the samples are independent and identically distributed. Obviously, this assumption is no longer valid when the samples are sequentially explored in the environment. DDPG uses a finite size experience pool to cut off the data correlation. The experience sequence is sampled from the environment according to the exploratory strategy, and the tuples are stored in the experience pool. When the experience pool is full, discard the oldest sample. At each time step, the actor network and critic network are updated by uniformly sampling small batches from the experience pool.

As mentioned above, we believe that DDPG is applicable to APS for the following reasons: First, Actor-Critic architecture can realize the learning of continuous action space (since the steering wheel angle for APS is a continuous action). Second, introducing the value function as the evaluation criterion in the policy search can reduce the variance of the policy gradient estimation, which is more efficient. Third, DDPG creates target networks for actor network and critic network, respectively, which makes it closer to the supervised learning and significantly enhances the stability of learning. Lastly, the experience pool is adopted to cut off the data correlation.

2.2.2. System Settings of DDPG

In this section, we mainly introduce the system settings of DDPG, including input and output, reward and network.

- Input and output

The input state s of DDPG refers to the parking slot relative to the vehicle, i.e., the coordinates of the four corner points in the vehicle CS. The output action a of DDPG is the steering wheel angle, capable of controlling the vehicle backing into the parking slot.

- Reward

At present, the reward of reinforcement learning mainly depends on expert experience. The goal of proposed algorithm is to make the vehicle parked in the middle of the parking slot, avoiding inclination, deviation, and line-pressing. We take these factors into consideration and through a large number of simulation training get a better reward setting as shown in Equations (19) to (24).

The total reward R consists of three parts, as expressed in Equation (19). The first part R_{cp} considers the reward that the vehicle tends to the center of the parking slot, and vehicle longitudinal axis parallels to the parking slot. The second part P_l and the last part P_d consider the punishment of line-pressing and the punishment of the vehicle's deviation to one side of the parking slot, respectively.

$$R = R_{cp} + P_l + P_d \tag{19}$$

As shown in Figure 7a, when the rear axle center of the vehicle is outside the outer line of the parking slot, P_{cp} is defined as:

$$\begin{gathered} P_{cp} = P_c + P_p \\ = \left(5 - 5\left(\tfrac{1}{2}abs\left(Y_{p_0} + Y_{p_1}\right) + \tfrac{1}{2}abs\left(Y_{p_2} + Y_{p_3}\right)\right)\right) + \left(5 - 5abs\left(\tfrac{Y_{p_0} - Y_{p_3}}{X_{p_0} - X_{p_3}}\right)\right) \end{gathered} \tag{20}$$

where P_c denotes the reward for the vehicle to be close to the center of the parking slot; P_p the reward for the vehicle's longitudinal axis parallel to the parking slot; (X, Y) the coordinates of the corner points ($P_0 - P_3$ in Figure 7) of the parking slot in the vehicle CS.

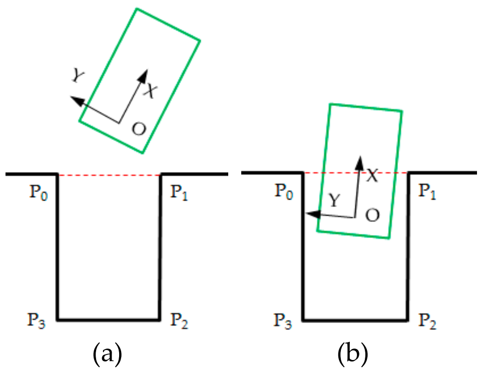


Figure 7. Different parking stages. (**a**)The rear axle center of the vehicle is outside the outer line of the parking slot. (**b**) The rear axle center of the vehicle crosses the outer line of the parking slot.

As shown in Figure 7b, when the rear axle center of the vehicle crosses the outer line of the parking slot, R_{cp} is defined as:

$$R_{cp} = min\left\{R_c, R_p\right\} + \frac{1}{2}max\left\{R_c, R_p\right\} + R_n \tag{21}$$

The reason why the larger values in R_c and R_p are reduced is to prevent falling into one better and conceal the worse performance of the other, so more attention is paid to the worse performance of the two. Besides, when the vehicle enters the parking slot, we pay more attention to the parallelism between the vehicle and the parking slot. Accordingly, a reward R_n is set, which is expressed in Equation (22). It is limited to a value of less than 10 to avoid covering other rewards.

$$R_n = abs(\frac{1}{10}min\left\{1/abs\left(\frac{Y_{p_0} - Y_{p_3}}{X_{p_0} - X_{p_3}}\right) + eps\right), 100\})) \tag{22}$$

If any outer contour boundary of the vehicle intersects with the parking slot lines, it is considered a line-pressing, P_l is defined as:

$$P_l = -10 \tag{23}$$

If the vehicle is biased towards one side of the parking slot, P_d is defined as:

$$P_d = -10 \tag{24}$$

- Network

The input of our network is not the image but the result of the parking slot detection. Thus, the deep neural networks are not necessarily required to be used. For actor network and critic network of DDPG, we just use back propagation neural network in this paper. Besides, we build the target network with an identical structure but different parameters for actor network and critic network and the relationship between network parameter θ and its target network parameters θ' is $\theta' \leftarrow \tau\theta + (1 - \tau)\theta', \tau \ll 1$, significantly enhancing the stability of learning.

The structure of actor network and target actor network is illustrated in Figure 8. The number of nodes for the coordinates of four corner points is 8, and the number of nodes in hidden layer la_1 and hidden layer la_2 is 100 and 200, respectively. Then, the number of nodes for steering wheel angle is 1. All activation functions are Rectified Linear Unit (ReLU).

The structure of critic network and target critic network is shown in Figure 9. The number of nodes for the coordinates of four corner points is 8, and the number of nodes in hidden layer ls_1 and hidden layer ls_2 are both 100. The number of nodes for steering wheel angle is 1, and the number of nodes in hidden layer lc_1 is 200. Subsequently, the number of nodes in hidden layer l_1 and hidden layer l_2 is 300 and 200, respectively. Lastly, the number of nodes for reward is l. All activation functions are also ReLU.

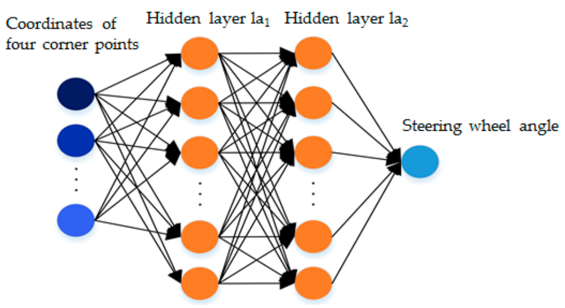


Figure 8. The structure of actor network and target actor network.

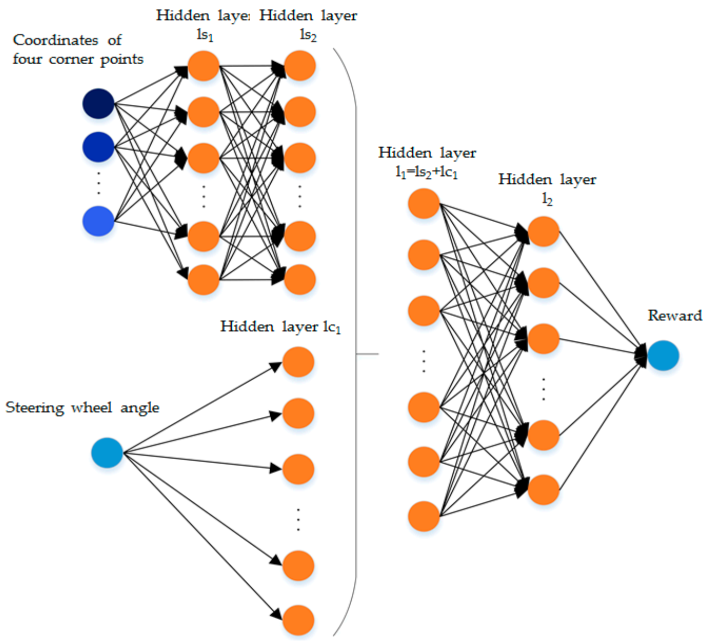


Figure 9. The structure of critic network and target critic network.

2.2.3. Training Process of DDPG

First, the training is conducted in the simulation environment. The simulation platform is shown in Figure 10. We use PreScan, MATLAB/Simulink, and Python in sequence to build the parking environment, build the vehicle model, and then run our algorithm, respectively. After the simulation training, the well-trained network migrates to the real vehicle training. In Section 3, the real vehicle platform will be introduced.

The training architecture of DDPG is shown in Figure 11, and the corresponding training process is shown in Algorithm 1.

Algorithm 1: DDPG Algorithm

```
Randomly initialize critic network $Q(s, a/\theta^Q)$ and actor network $\mu(s/\theta^\mu)$ with parameters $\theta^Q$ and $\theta^\mu$
Initialize target critic network $Q'$ and target actor network $\mu'$ with parameters $\theta^{Q'}$ and $\theta^{\mu'}$
Set up a replay memory buffer (experience pool) for the sampling experience sequence with the total number of buffers $M$
for each episode:
Initialize a random process for action exploration
Receive initial state $s_1$
for $t = 1, T$:

  1. Select action $a_t$ according to the current policy and exploration noise:
       $a_t = \mu(s_t/\theta^\mu) + N_t$
     where $N_t$ denotes Gaussian noise.
  2. Execute action $a_t$ and obtain the reward $r_t$ and the next state $s_{t+1}$
  3. Store the transition $(s_t, a_t, r_t, s_{t+1})$ in the experience pool
  4. Randomly sample $N$ experience sequences from experience pool as a mini-batch training data for the critic network and actor network
  5. This step is adopted to update the parameters of the critic network. With a method similar to supervised learning, loss is defined as:
       $L = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i/\theta^Q)\right)^2$
     where $y_i$ is calculated based on $\mu'$ and $Q'$:
       $y_i = R_i + \gamma Q'\left(s_{i+1}, \mu'(s_{i+1}/\theta^{\mu'})/\theta^{Q'}\right)$
     Calculate the gradient $\nabla_{\theta^Q} L$, and then update $\theta^Q$ with gradient descent method:
       $\theta^Q = \theta^Q + \alpha\, \nabla_{\theta^Q} L$
     where $\alpha$ is the learning rate.
  6. After the critic network is updated, the actor network is updated using the policy gradient method:
       $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a/\theta^Q)/_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s/\theta^\mu)/_{s_i}$
     Update $\theta^\mu$ with $\nabla_{\theta^\mu} J$ based on gradient descent method:
       $\theta^\mu = \theta^\mu + \alpha\nabla_{\theta^\mu} J$
  7. Update the target networks:
       $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
       $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$

  end for
end for
```

2.2.4. Improved Training Measures Applied in Parking

Given that the learning network output is hard to converge and it is easy to fall into local optimum in the parking process, several reinforcement learning training methods in terms of parking conditions are designed.

- Manual guided exploration for accumulating initial experience sequence

Before the training of network, exploration should be conducted to gain the initial experience sequence database. In the initialization stage, instead of random exploration, we conduct manual guidance on exploration, which is realized by setting a series of control commands for the initial parking slot relative to the vehicle (the driver's control sequence is collected in the simulation or real vehicle test). Based on the manual control commands, the appropriate noise is added to give the model a better space for policy exploration and trial-and-error. In such a way, compared with random exploration, considerable experience sequences will receive higher rewards, which can make the training converge to excellent policy faster. The reward can converge eventually with manual guided exploration, as shown in Figure 12.

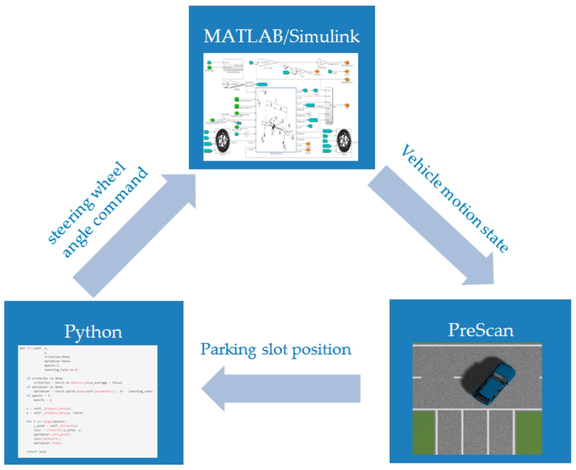


Figure 10. Simulation platform.

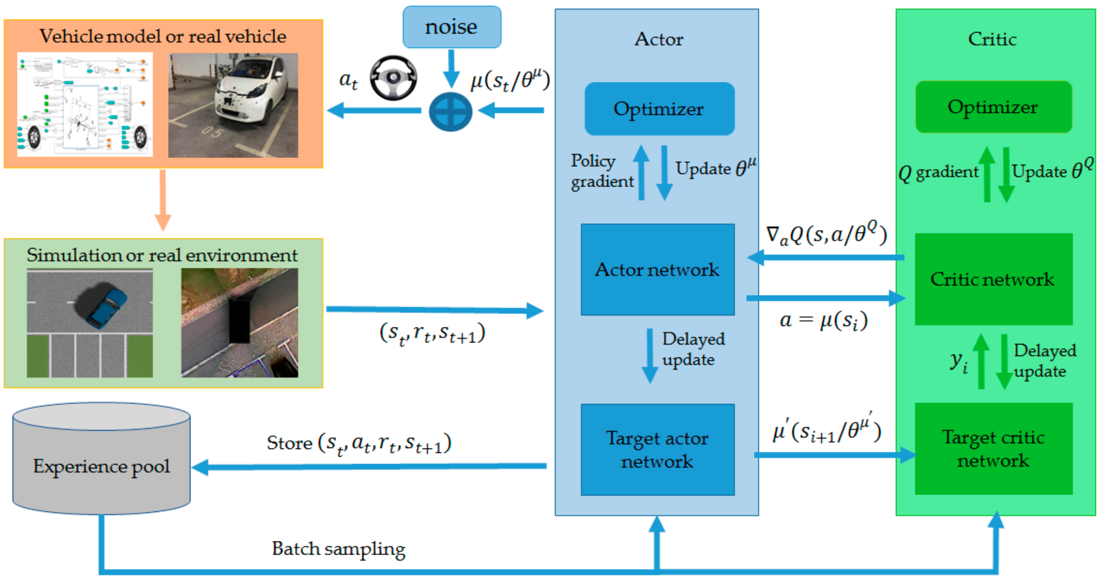
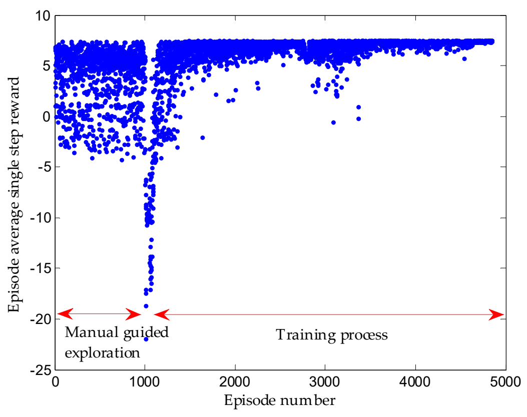
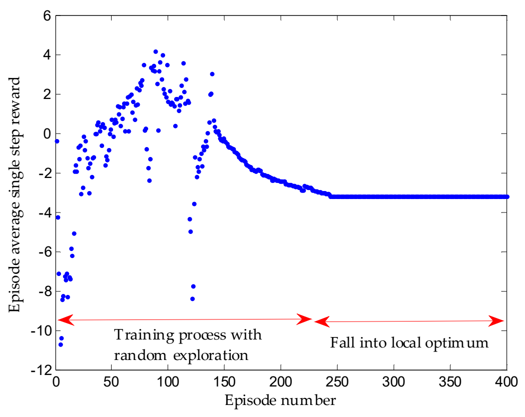


Figure 11. The training architecture of DDPG.



(**a**)

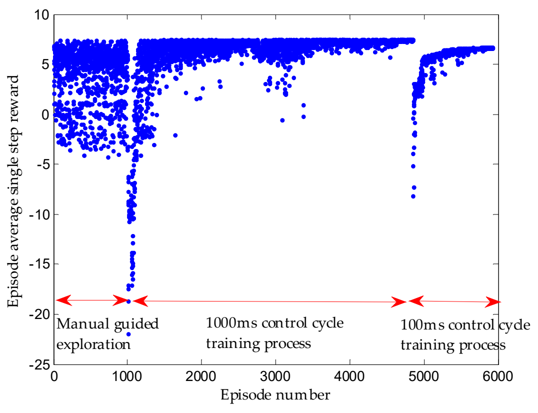


(**b**)

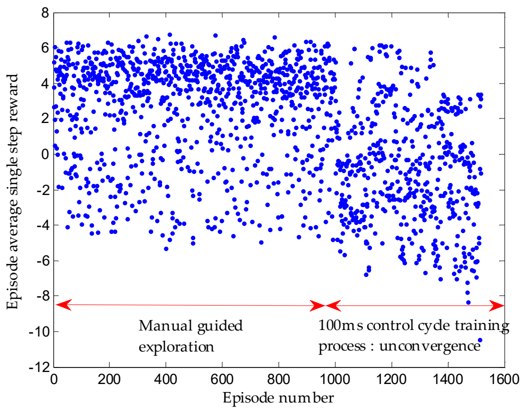
Figure 12. (**a**) Training process by manual guided exploration. (**b**) Training process by random exploration.

- Control cycle phased setting

Given that the vehicle model has inertia delay characteristics, it is found that if the period of steering wheel angle change is too small, it will cause the loss of Markov characteristics of some collected experience sequences. The state of the current cycle of the vehicle depends on both the state of the previous cycle and the action taken. To weaken this adverse effect, the first round of training sets the control cycle to 1000 ms. In such a way, the actions executed in the current cycle will retain sufficient execution time, which will be the major factor affecting the state of the next cycle and can be approximated to Markov decision-making process. When the network converges to the optimum, the training control cycle of the following training can be reduced, which can make the control cycle closer to the actual situation and achieve better results. Figure 13a shows that the 1000 ms control cycle is first trained, then the 100 ms control cycle is trained, and lastly the reward lastly converges. Figure 13b suggests that the reward does not converge if we start with a 100 ms control cycle training directly.



(**a**)



(**b**)

Figure 13. (**a**) Training process first by 1000 ms control cycle and followed by 100 ms control cycle. (**b**) Training process only by 100 ms control cycle.

- Training condition phased setting

Usually the perpendicular parking can be split into two steps, as shown in Figure 14. Just like human parking, the "step two" plays a major role in the final parking attitude. Thus, we currently primarily apply reinforcement learning to the "step two".

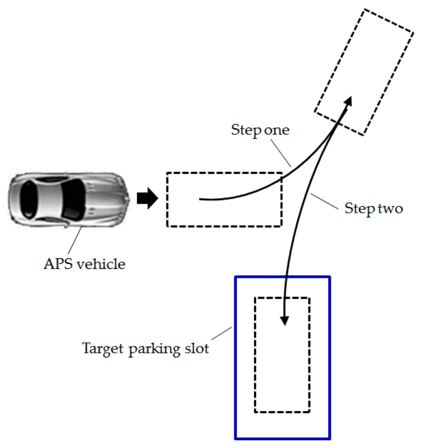


Figure 14. Common perpendicular parking process.

Since there will be different initial angles of "step two" between the vehicle and the parking slot, we first train at 30° and then expand the initial angle to 0° to 90° to continue training. Figure 15 suggests that based on 30° well-trained network, the networks between 0° to 90° can converge quickly, i.e., the 30° well-trained network has ideal generalization ability. Since the initial angle of the vehicle relative to the parking slot is different in different episodes, the sequence of states experienced in each episode is different, so the average single step reward is also different.

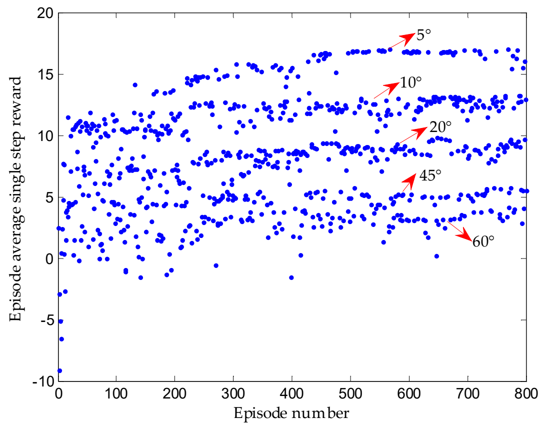


Figure 15. Extended training of other initial angles based on 30° well-trained network.

- Real vehicle training migration

Because the real vehicle training takes a lot of manpower, time and resources, it is better to train in the simulation environment and then transfer it to the real vehicle. Since the sensor model and vehicle model used in simulation will differ from the real vehicle, the same control command may produce different observation results. Accordingly, the real vehicle should be continuously trained based on well-trained network in simulation. Figure 16 reveals that the result of real vehicle migration training is ideal.

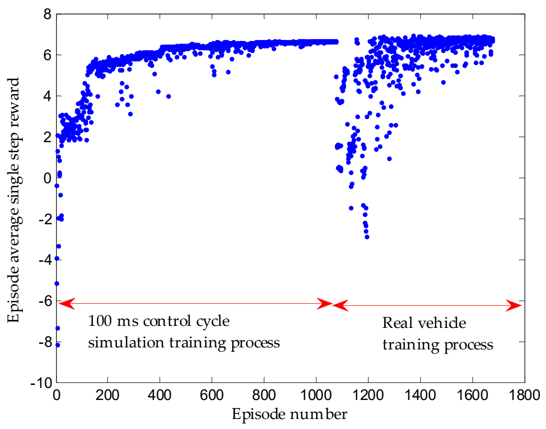


Figure 16. Real vehicle training migration.

3. Experimental Results

After the above training, we can ascertain the performance of the trained algorithm. This section shows the experimental platform, experimental scenes, and results.

3.1. Experimental Platform

The experimental platform is refitted from Rongwei E50 pure electric vehicle (Figure 17). Four fisheye cameras act as sensors for parking slot detection. The algorithm running platform is an industrial computer (i5 processor, 8G memory, 128G solid-state hard disk). Chassis control and information exchange is performed in the vehicle control unit, i.e., the controller of vehicle chassis. The RT3000 navigation system is employed to acquire the position information of the vehicle. During the test, notebook computer is employed to record data.

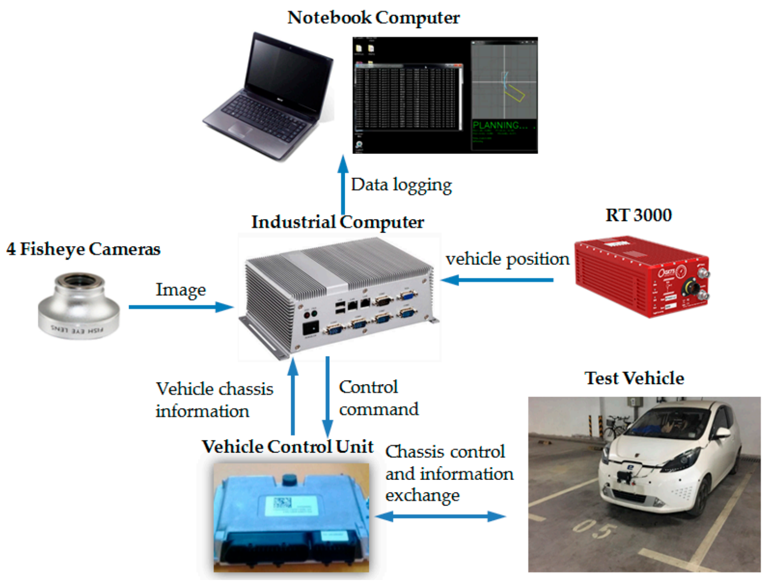


Figure 17. Experimental platform.

3.2. Experimental Scenes

To ascertain the performance of the proposed algorithm in the “step two” perpendicular parking, we choose three parking scenes with initial angles of 60°, 45°, and 30° between the vehicle and the parking slot, which are common “step two” scenes. Figure 18 illustrates the experimental scenes expressed in the surround view. The blue marking points represent the target parking slots; the width of these parking slots ranges from 2.4 m to 2.44 m and the length is between 5.6 m and 5.8 m, which basically meets the test requirements of BS ISO 16787-2016. As described in Section 2.1.1, since the FOV of surround view cannot cover the entire parking slot, only the nearby corner points can be detected, i.e., the width of the parking slot can be detected, and the length can only be inferred from priori rules.

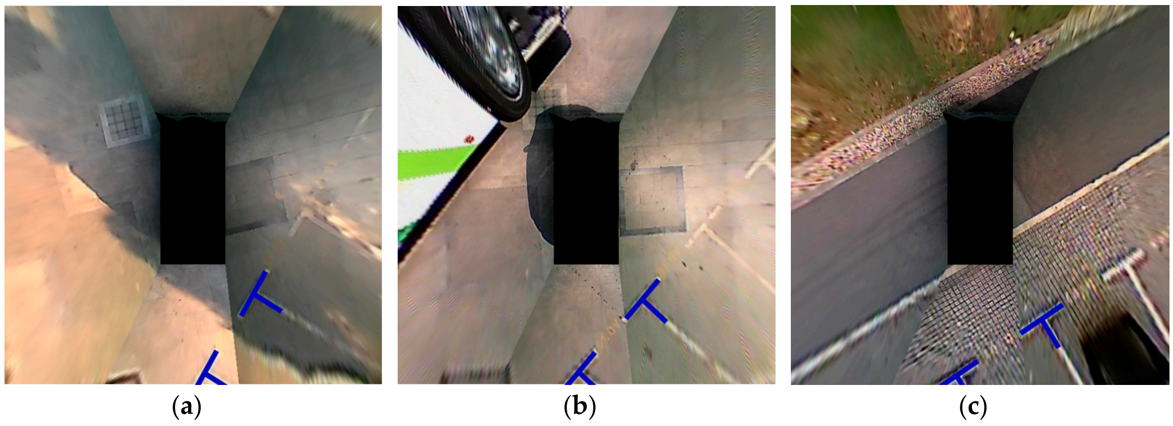


Figure 18. Experimental scenes. (**a**), (**b**) and (**c**) present parking scenes with initial angles of 60°, 45°, and 30°, respectively.

Three parking methods are adopted in the experiment: geometric method-based path planning with PID-based path tracking, geometric method-based path planning with SMC-based path tracking, and reinforcement learning-based end-to-end parking. The first two represent the current mainstream parking methods, and the last one represents the method used in this paper. Each parking method is at the same starting point and reversed at the same speed (4km/h).

Subsequently, the parking performances of different parking methods are compared. According to BS ISO 16787-2016, the inclination angle of the vehicle with respect to the parking slot, the deviation between the four tire contact points of the vehicle and the parking slot, and the deviation between the rear of the vehicle and the parking slot are measured. The measurement parameters are presented in Figure 19.

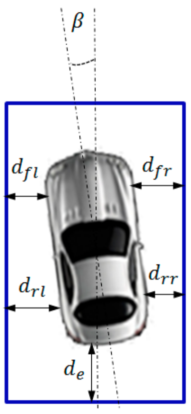


Figure 19. Measurement of parking attitude.

3.3. Results

The experimental results of 60° perpendicular parking are presented in Figure 20. Figure 20b shows that only planned path can ensure the ideal parking attitude, whereas the two path tracking methods (PID and SMC) cannot completely track the planned parking path. The existing control error causes the final vehicle to deviate from the ideal parking attitude, as shown in Figure 20b,c. Figure 20b also shows that the parking performance of reinforcement learning is better than those of the other two methods. Besides, the changes of the parking slot in the vehicle CS are recorded in the case of only visual detection and parking slot tracking in the experiment of reinforcement learning, as shown in Figure 20d. It is suggested that visual detection has missed detection, and it cannot provide continuous parking slot for reinforcement learning. Thus, the test cannot be performed normally. However, the parking slot tracking has not missed detection.

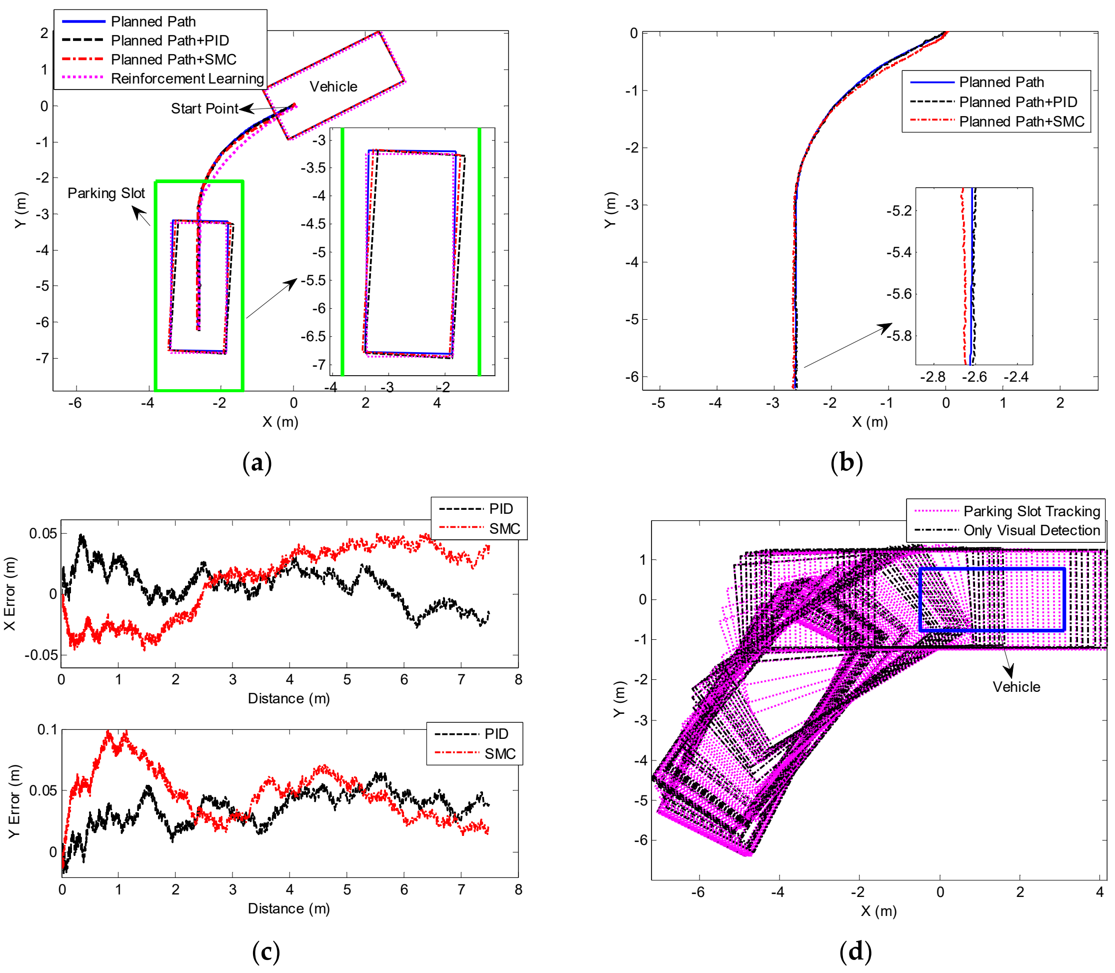


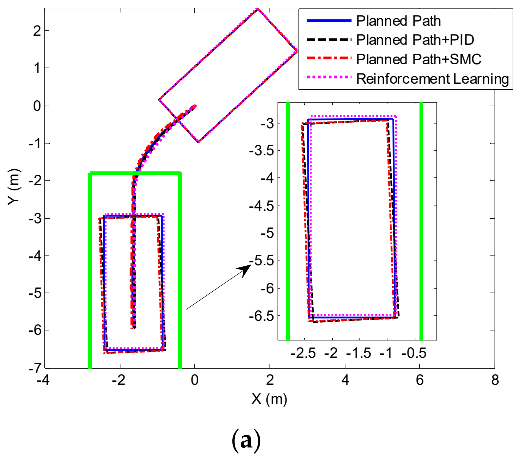
Figure 20. Experimental results of 60° perpendicular parking. (**a**) Parking performance of different methods; (**b**) and (**c**) present the control effect and error of different path tracking methods, respectively. (**d**) Parking slot detection and tracking in the parking process.

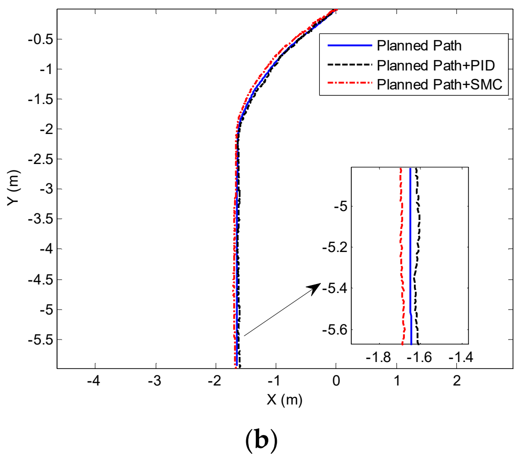
The experimental data corresponding to Figure 20 is listed in Table 1. This table suggests that reinforcement learning can achieve an inclination angle of –0.747°, satisfying the requirements of the BS ISO 16787-2016 (≤±3°). Moreover, these deviations are relatively uniform, satisfying the requirements of the BS ISO 16787-2016 (>0.1 m). As mentioned above, only planned path can ensure that the ideal parking and inclination angle and deviation meet the requirements of the standard. However, when path tracking is practically performed, these deviations of path planning with PID and path planning with SMC are not uniform, and the inclination angles are –3.638° and –3.126°, respectively, which do not satisfy the requirements. These two path tracking methods have errors of more than 0.02 m in both X and Y directions. Lastly, it is suggested that the loss rate of visual detection is 37.35%, and that of the parking slot tracking reaches 0%.

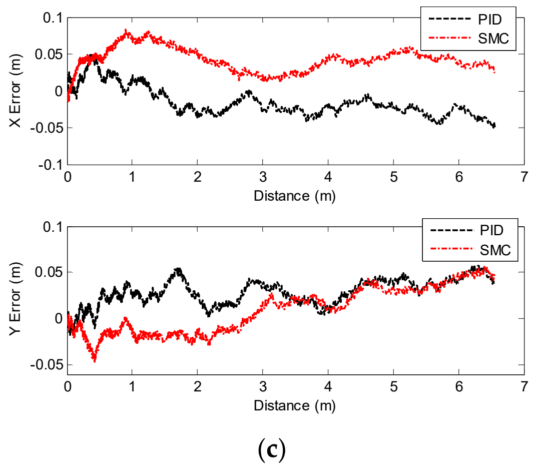
Table 1. Experimental data of 60° perpendicular parking.

	Planned Path	Planned Path+PID	Planned Path+SMC	Reinforcement Learning
Inclination angle β (°)	−1.051	−3.638	−3.126	−0.747
Deviation d_{fr} (m)	0.423	0.304	0.379	0.457
Deviation d_{f1} (m)	0.468	0.589	0.514	0.463
Deviation d_{rr} (m)	0.465	0.45	0.504	0.487
Deviation d_{rl} (m)	0.425	0.443	0.388	0.434
Deviation d_{e} (m)	1.087	1.016	1.047	1.053
X average error (m)	\	0.021	0.028	\
Y average error (m)	\	0.033	0.048	\
Loss rate of visual detection (%)	\	\	\	37.35
Loss rate of parking slot tracking (%)	\	\	\	0

The experimental results of perpendicular parking at 45° and 30° initial angle are shown in Figures 21 and 22. On the whole, the results are consistent with the 60° test. The parking performance of reinforcement learning is obviously superior over those of the other two methods, suggesting that our algorithm can adapt to parking scenario with different initial angles. Likewise, both PID and SMC have control errors, making it unlikely for the vehicle to track the parking path accurately, and eventually the vehicle has inclination angle and uniform deviation.







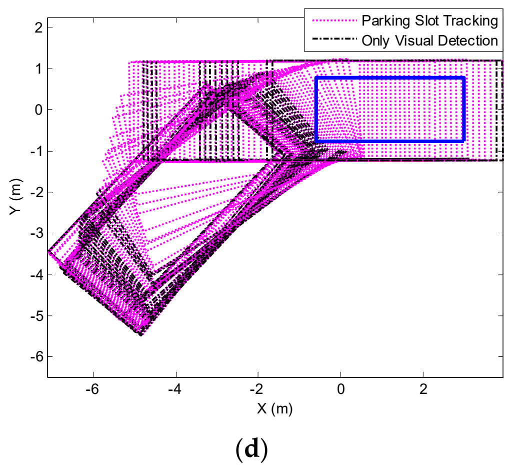


Figure 21. Experimental results of 45° perpendicular parking. (**a**) Parking performance of different methods; (**b**) and (**c**) represent the control effect and error of different path tracking methods, respectively. (**d**) Parking slot detection and tracking in the parking process.

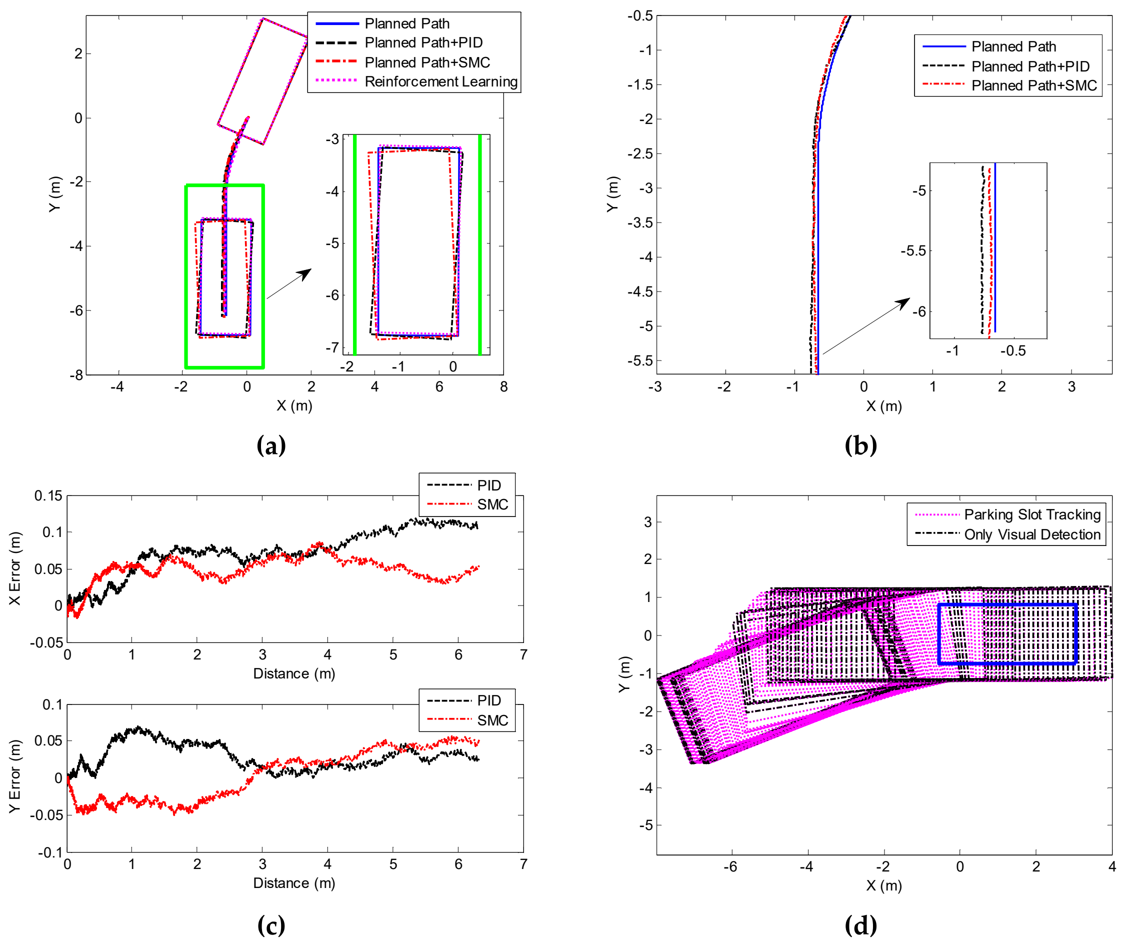


Figure 22. Experimental results of 30° perpendicular parking. (**a**) Parking performance of different methods; (**b**) and (**c**) present the control effect and error of different path tracking methods, respectively. (**d**) Parking slot detection and tracking in the parking process.

Tables 2 and 3 suggest that reinforcement learning can achieve an inclination angle below 1°, satisfying the standard requirements. The other two methods have large inclination angle due to control error, especially the PID exceeding 3°. The path tracking errors of these two methods in X and Y directions are basically above 0.02 m. Besides, the loss rate of visual detection in these two scenarios reaches over 30%, and the parking slot tracking ensures that the position of the target parking slot can be continuously achieved.

Table 2. Experimental data of 45° perpendicular parking.

	Planned Path	**Planned Path+PID**	**Planned Path+SMC**	**Reinforcement Learning**
Inclination angle β (°)	0.313	3.088	2.011	−0.573
Deviation d_{fr} (m)	0.493	0.557	0.59	0.438
Deviation d_{fl} (m)	0.377	0.315	0.281	0.436
Deviation d_{rr} (m)	0.48	0.433	0.509	0.461
Deviation d_{rl} (m)	0.391	0.439	0.361	0.413
Deviation d_{e} (m)	0.872	0.788	0.795	0.918
X average error (m)	\	0.032	0.042	\
Y average error (m)	\	0.024	0.019	\
Loss rate of visual detection (%)	\	\	\	43.68
Loss rate of parking slot tracking (%)	\	\	\	0

Table 3. Experimental data of 30° perpendicular parking.

	Planned Path	Planned Path+PID	Planned Path+SMC	Reinforcement Learning
Inclination angle β (°)	−0.223	−3.782	2.416	−1.02
Deviation d_{fr} (m)	0.394	0.363	0.552	0.376
Deviation d_{fl} (m)	0.456	0.49	0.299	0.474
Deviation d_{rr} (m)	0.403	0.515	0.455	0.417
Deviation d_{rl} (m)	0.447	0.339	0.396	0.434
Deviation d_e (m)	1.031	0.952	0.948	1.056
X average error (m)	\	0.056	0.043	\
Y average error (m)	\	0.028	0.031	\
Loss rate of visual detection (%)	\	\	\	31.48
Loss rate of parking slot tracking (%)	\	\	\	0

4. Discussion

The above experimental results reveal that the existing mainstream parking methods of path planning with path tracking can basically park the vehicle into the parking slot, whereas the final inclination angle of the vehicle does not meet the strict requirements of the standard. This method is feasible for some wide parking slots. However, with the increasing number of vehicles, the design of parking slot will become narrower and narrower. Thus, the accuracy of parking should be enhanced. Besides, we can also see that it is not difficult to plan an ideal parking path according to the parking slot. However, due to the nonlinear dynamic characteristics of the vehicle, path tracking will inevitably produce control errors that cause the vehicle to deviate from the planned path, thereby resulting in inclination angle and uniform deviation of the parking attitude.

The reinforcement learning-based end-to-end planning method can not only achieve the end-to-end parking from parking slot to steering wheel angle, avoiding errors caused by path tracking, but also learn the best steering wheel angle through a lot of training. Thus, the reinforcement learning-based end-to-end planning can achieve better parking attitude. Besides, because we have fused the vision and vehicle chassis information, we can continuously get the position of parking slot to ensure the normal training and testing of reinforcement learning.

However, future research can still make some improvements: (1) The reward setting of this article is obtained by artificial setting and experimental adjustment. Though the final effect converges to an ideal level, it cannot be proved that it is the optimal reward setting. Accordingly, we will consider the method of inverse reinforcement learning [33,34] to optimize the reward. (2) In this paper, the reinforcement learning-based parking only has the function of reversing (e.g., "step two" in Figure 14), and it cannot automatically adjust the gear forward and backward. If the vehicle needs to judge the gear, we will consider selecting the Long Short-Term Memory (LSTM) network [35].

5. Conclusions

In this study, we innovatively adopt reinforcement learning to perpendicular parking so that the vehicle can continuously learn and accumulate experience from considerable parking attempts, learn the command of the optimal steering wheel angle at different parking slots relative to vehicle, as well as achieve real, "human-like" intelligent parking. Moreover, such end-to-end planning can avoid errors caused by path tracking. Besides, to ensure that the parking slot can be obtained continuously in the course of learning, a parking slot tracking algorithm is proposed based on fusion of vision and vehicle chassis information. Besides, since the learning network output is hard to converge and it is easy to fall into local optimum in the parking process, several reinforcement learning training methods in terms of parking conditions are designed (e.g., manual guided exploration for accumulating initial experience sequence, control cycle phased setting, and training condition phased setting). Lastly, the well-trained network in the simulation environment is migrated to the real vehicle training.

In the subsequent study, on one hand, inverse reinforcement learning will be used to set rewards to ensure optimal reward settings; on the other hand, the LSTM network will be used to achieve gear adjustment in the parking process.

Author Contributions: Conceptualization, L.X., P.Z. and Z.Y.; funding acquisition, Z.Y.; investigation, P.Z. and P.F.; methodology, P.Z. and L.X.; software, P.Z., S.Y.; validation, P.Z., P.F., S.Y., J.Y. and Y.Z.; resources, L.X. and Z.Y.; supervision, X.L. and Z.Y.; writing—original draft, P.Z.; writing—review and editing, P.Z. and L.X.

Funding: This research is supported by the National Key Research and Development Program of China (grant no.2016YFB0100901) and Shanghai Scientific Research Project (grant no. 17DZ1100202).

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

References

1. British Standards Institution. BS ISO 16787:2016. Intelligent transport systems-Assisted Parking System (APS)-Performance requirements and test procedures. BSI Standards Limited: UK, 2016. Available online: https://www.iso.org/standard/63626.html (accessed on 10 September 2019).
2. Fraichard, T.; Scheuer, A. From reeds and shepp's to continuous-curvature paths. *IEEE Trans. Robot.* **2004**, *6*, 1025–1035. [CrossRef]
3. Gómez-Bravo, F.; Cuesta, F.; Ollero, A. Parallel and diagonal parking in nonholonomic autonomous vehicles. *Eng. Appl. Artif. Intell.* **2001**, *14*, 419–434. [CrossRef]
4. Lini, G.; Piazzi, A.; Consolini, L. Multi-optimization of η3-splines for autonomous parking. In Proceedings of the 50th IEEE Conference of Decision and Control/European Control Conference, Orlando, FL, USA, 12–15 December 2011; pp. 6367–6372.
5. Vorobieva, H.; Minoiu-Enache, N.; Glaser, S.; Mammar, S. Geometric continuous-curvature path planning for automatic parallel parking. In Proceedings of the 2013 IEEE 10th International Conference on Networking, Sensing and Control, Evry, France, 10–12 April 2013; pp. 418–423.
6. Vorobieva, H.; Glaser, S.; Minoiu-Enache, N.; Mammar, S. Automatic parallel parking in tiny spots: Path planning and control. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 396–410. [CrossRef]
7. Han, L.; Do, Q.H.; Mita, S. Unified path planner for parking an autonomous vehicle based on RRT. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 5622–5627.
8. Zheng, K.; Liu, S. RRT based path planning for autonomous parking of vehicle. In Proceedings of the 2018 IEEE 7th Data Driven Control and Learning Systems Conference, Enshi, China, 25–27 May 2018; pp. 627–632.
9. Li, B.; Shao, Z. A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. *Knowledge-Based Syst.* **2015**, *86*, 11–20. [CrossRef]
10. Takei, R.; Tsai, R. Optimal trajectories of curvature constrained motion in the Hamilton-jacobi formulation. *J. Sci. Comput.* **2013**, *54*, 622–644. [CrossRef]
11. Micelli, P.; Consolini, L.; Locatelli, M. Path planning with limited numbers of maneuvers for automatic guided vehicles: An optimization-based approach. In Proceedings of the 2017 25th Mediterranean Conference on Control and Automation, Valletta, Malta, 3–6 July 2017; pp. 204–209.
12. Roald, A.L. Path planning for vehicle motion control using numerical optimization methods. Master's Thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2015.
13. Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D. Stanley: The robot that won the DARPA Grand Challenge. *J. Field Robot.* **2006**, *23*, 661–692. [CrossRef]
14. Park, H.; Ahn, K.; Park, M.; Lee, S. Study on robust lateral controller for differential GPS-based autonomous vehicles. *Int. J. Precis. Eng. Manuf.* **2018**, *19*, 367–376. [CrossRef]
15. Mvemba, P.; Lay-Ekuakille, A.; Kidiamboko, S.; Rahman, M. An embedded beamformer for a PID-based trajectory sensing for an autonomous vehicle. *Metrol. Meas. Syst.* **2018**, *25*, 561–575.
16. He, X.; Liu, Y.; Lv, C.; Ji, X.; Liu, Y. Emergency steering control of autonomous vehicle for collision avoidance and stabilization. *Veh. Syst. Dyn.* **2019**, *57*, 1163–1187. [CrossRef]
17. Wu, Y.; Wang, L.; Zhang, J.; Li, F. Path following control of autonomous ground vehicle based on nonsingular terminal sliding mode and active disturbance rejection control. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6379–6390. [CrossRef]

18. Samuel, M.; Maziah, M.; Hussien, M.; Godi, N. Control of autonomous vehicle using path tracking: A review. In Proceedings of the International Conference on Science, Engineering, Management and Social Sciences, Johor Bahru, Malaysia, 6–8 October 2016; pp. 3877–3879.
19. Cui, Q.; Ding, R.; Zhou, B.; Wu, X. Path-tracking of an autonomous vehicle via model predictive control and nonlinear filtering. *Proc. Inst. Mech. Eng. Part D-J. Automob. Eng.* **2018**, *232*, 1237–1252. [CrossRef]
20. Yu, Z.; Zhang, R.; Xiong, L.; Fu, Z. Robust hierarchical controller with conditional integrator based on small gain theorem for reference trajectory tracking of autonomous vehicles. *Veh. Syst. Dyn.* **2019**, *57*, 1143–1162. [CrossRef]
21. Lidberg, M.; Muller, S. Special issue on motion control for automated driving and autonomous functions on road vehicles. *Veh. Syst. Dyn.* **2019**, *57*, 1087–1089. [CrossRef]
22. Sun, R.; Silver, D.; Tesauro, G.; Huang, G. Introduction to the special issue on deep reinforcement learning: An editorial. *Neural Netw.* **2018**, *107*, 1–2. [CrossRef] [PubMed]
23. Van, H.; Guez, A.; Silver, D. Deep reinforcement learning with double Q-Learning. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 2094–2100.
24. Sharifzadeh, S.; Chiotellis, I.; Triebel, R.; Cremers, D. Learning to drive using inverse reinforcement learning and Deep Q-Networks. *arXiv* **2016**, arXiv:1612.03653.
25. Gu, S.; Lillicrap, T.; Ghahramani, Z.; Turner, R.; Levine, S. Q-Prop: Sample-efficient policy gradient with an off-policy critic. *arXiv* **2016**, arXiv:1611.02247.
26. Jagodnik, K.; Thomas, P.; Branicky, M.; Kirsch, R. Training an Actor-Critic reinforcement learning controller for arm movement using human-generated rewards. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2017**, *25*, 1892–1905. [CrossRef] [PubMed]
27. Fan, Q.; Yang, G.; Ye, D. Quantization-Based Adaptive Actor-Critic Tracking Control With Tracking Error Constraints. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 970–980. [CrossRef] [PubMed]
28. Skach, J.; Kiumarsi, B.; Lewis, F.; Straka, O. Actor-Critic off-policy learning for optimal control of multiple-model discrete-time systems. *IEEE T. Cybern.* **2018**, *48*, 29–40. [CrossRef]
29. Lillicrap, T.; Hunt, J.; Pritzel, A.; Heess, N.; Erez, T. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
30. Xu, J.; Hou, Z.; Wang, W.; Xu, B.; Zhang, K.; Chen, K. Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks. *IEEE Trans. Ind. Inform.* **2019**, *15*, 1658–1667. [CrossRef]
31. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
32. Li, L.; Zhang, L.; Li, X.; Liu, X.; Shen, Y.; Xiong, L. Vision-based parking-slot detection: A benchmark and a learning-based approach. In Proceedings of the 2017 IEEE International Conference on Multimedia and Expo, Hong Kong, China, 10–14 July 2017; pp. 649–654.
33. Imani, M.; Braga-Neto, U. Control of gene regulatory networks using bayesian inverse reinforcement learning. *IEEE-ACM Trans. Comput. Biol. Bioinform.* **2019**, *16*, 1250–1261. [CrossRef] [PubMed]
34. Pan, W.; Qu, R.; Hwang, K.; Lin, H. An ensemble fuzzy approach for inverse reinforcement learning. *Int. J. Fuzzy Syst.* **2019**, *21*, 95–103. [CrossRef]
35. Gao, C.; Yan, J.; Zhou, S.; Chen, B.; Liu, H. Long short-term memory-based recurrent neural networks for nonlinear target tracking. *Signal Process.* **2019**, *164*, 67–73. [CrossRef]