Learning to Drive Like Human Beings: A Method Based on Deep Reinforcement Learning

Yantao Tian, Xuanhao Cao, Kai Huang, Cong Fei, Zhu Zheng, and Xuewu Ji

***Abstract*— In this paper, a new framework for path tracking is proposed through learning to drive like human beings. Firstly, the imitation algorithm (behavior cloning) is adopted to initialize the deep reinforcement learning (DRL) algorithm through learning the professional drivers' experience. Secondly, a continuous, deterministic, model free deep reinforcement learning algorithm is adopted to optimize our DRL model on line through trial and error. By combining behavior cloning and deep reinforcement learning algorithms, the DRL model can learn an effective policy quickly for path tracking using some easy-to-measure vehicle state parameters and environment information as inputs. Actor-Critic structure is adopted in the DRL algorithm. In order to speed up the convergence rate of the DRL model and improve the learning effect, we propose a dual actor networks structure for the two different action outputs (steering wheel angle and vehicle speed), and a chief critic network is built to guide the updating process of dual actor networks at the same time. Based on this dual actor networks structure, we can pick out some more important state information as state inputs for different action outputs. Besides, a kind of reward mechanism is also presented for autonomous driving. Finally, simulation training and experiment test are carried out, and the results confirm that the framework proposed in this paper is more than data efficient than the original algorithm, and the trained DRL model can track the reference path with accuracy and has the generalization ability for different roads.**

***Index Terms*— Autonomous driving, path tracking, imitation learning, reinforcement learning.**

I. Introduction

AUTONOMOUS driving is a hot topic that has attracted a great attention from both companies and research groups, due to its potential to provide a new solution to some traffic problems (vehicle jam, traffic accident, etc) [1]–[3]. And path tracking with accuracy is a prerequisite for realizing self-driving task of the vehicle.

The main task of autonomous driving is to enable the vehicle to immediately sense the environment information, and then make rapid judgment. In order to achieve this main task, the most important work is to make the vehicle learn a reasonable autonomous driving strategy which can generate control commands (steering wheel, gas pedal, and brake pedal) based on the information obtained from environment. Now, many different machine learning algorithms are used in the field of autonomous driving. Fully end-to-end supervised learning can be regarded as an effective solution to autonomous driving, which trains a neural network model to map environment information input to action outputs. Pioneering work by Pomerleau (1989) proved the possibility of learning to drive in an end-to-end manner, and a three-layered neural network was adopted to make the test vehicle with autonomous navigation follow the lane by the input information from a monocular camera and a laser range finder [5]. Lecun *et al.* (2005) trained a vision-based obstacle avoidance system by using the end-to-end supervised learning to map raw input images to steering angles for off-road mobile robots, and the network has 3.15 million connections and about 72,000 trainable parameters [6]. Chen C *et al.* (2015) proposed a direct perception approach to estimate the affordance for driving, which synthesized mediated perception approach and behavior reflex approach [7]. Mariusz *et al.* (2016) trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands, and the trained network was taken out for a road test [8]. More recently, Xu H *et al.* (2017) proposed a deep learning architecture for learning-to-drive from uncalibrated large-scale video data [9]. Codevilla *et al.* (2018) proposed a condition imitation learning method to map the camera inputs to the steering and acceleration outputs [10].

However, supervised learning usually requires a lot of data to train a model in order that the trained model can generalize to different environments. And it will expend a lot of time and need a large human work to collect the training data. In contrast, reinforcement learning (RL) can learn self-directed effectively through trial and error and need no supervision [28]. The applications of reinforcement learning have been infiltrated into many fields, such as robotics [11]–[13], games [14]–[16], autonomous driving [17], [18], etc. Recently, the reinforcement learning algorithm can be regarded as one

Manuscript received 27 November 2019; revised 22 September 2020 and 19 January 2021; accepted 27 January 2021. Date of publication 10 February 2021; date of current version 8 July 2022. This work was supported in part by the National Nature Science Foundation under Grant U19A2069 and Grant U1664263 and in part by the National Natural Science Foundation of China under Project 51975311. The Associate Editor for this article was L. Li. *(Corresponding author: Xuewu Ji.)*

Yantao Tian is with the Department of Control Science and Engineering, Jilin University, Changchun 130022, China, and also with the Key Laboratory of Bionic Engineering of Ministry of Education, Jilin University, Changchun 130022, China (e-mail: tianyt@jlu.edu.cn).

Xuanhao Cao and Kai Huang are with the Department of Control Science and Engineering, Jilin University, Changchun 130022, China (e-mail: caoxh17@mails.jlu.edu.cn; huangk17@mails.jlu.edu.cn).

Cong Fei and Xuewu Ji are with the School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China (e-mail: feicong2018@163.com; jixw@mail.tsinghua.edu.cn).

Zhu Zheng is with the School of Mechatronics and Vehicle Engineering, Chongqing Jiaotong University, Chongqing 400074, China (e-mail: zhengzhu2019@163.com).

Digital Object Identifier 10.1109/TITS.2021.3055899

of the promising methods to achieve autonomous driving technology.

Oh S Y *et al.* (2000) trained a RL model to solve the problem of high-speed road following on the high-curvature roads [4]. Riedmiller *et al.* (2007) designed a controller based on Reinforcement Learning (RL) called Neural Fitted Q Iteration (NFQ), which is able to learn a steering task in less than 20 minutes directly on the real car [19]. Isele *et al.* (2017) proposed an efficient strategy to navigate safely through unsigned intersections based on Deep Q-Networks (DQNs) [20]. Behzadan *et al.* (2018) applied the deep reinforcement learning to autonomous navigation, and a novel framework was proposed for benchmarking the behavior of collision avoidance mechanisms [21]. Pan *et al.* (2017) proposed a kind of virtual to real (VR) reinforcement learning method to make model trained in virtual environment workable in real world [22]. Ferdowsi *et al.* (2018) proposed a novel adversarial deep reinforcement learning algorithm framework to address security issues of autonomous driving [23]. For the unrestricted highway environments, Makantasis *et al.* (2019) proposed a driving policy based on the Double Deep Q-Network algorithm (DDQN) [24]. There are two problems with the deep reinforcement learning: one is lack of generalization capability to new goals, the other is data inefficiency. In order to solve these problems, Yuke *et al.* (2016) proposed an actor-critic model and an AI2-THOR framework [25].

The closest work to this paper is from Kendall *et al.* (2018) who trained an autonomous driving model using a continuous, model free deep reinforcement learning algorithm [26]. They demonstrated learning to drive with deep learning, from a single monocular image as input, using a reward function based on the distance travelled before exiting lane. Their work brings us a lot of inspiration. In this paper, we demonstrate learning to drive with deep learning, using some easy-to-measure vehicle state parameters and environment information as inputs, and proposed a reward function based on velocity and location information to realize the task of path tracking for autonomous driving under more complex road conditions.

In fact, it is not blind to learn to drive a car for a new driver. Usually, a new driver will learn some driving skills (such as how to start, how to turn, how to accelerate, etc) from his/her driving instructor, and then he/she will learn by continuous exploration under the guidance from driving instructor.

Based on this actual situation, in this paper, we propose a new framework for learning to drive like human beings. Firstly, behavior cloning algorithm is adopted to train our DRL model off-line, and through this process, our model learns some simple driver skills instead of generating some invalid action commands (such as turning around, astern running). In fact, it is an effective method to accelerate the learning process and diminish the exploratory space [27], [42]. After that, we use a continuous, deterministic, model-free deep reinforcement learning algorithm to optimize our model on line. Actor-Critic structure is adopted in the DRL algorithm. In order to speed up the convergence rate of the DRL model and improve the learning effect, we adopt dual actor networks structure for the two different action outputs (steering wheel angle and vehicle speed), and a chief critic network is built to guide the updating process of the dual actor networks at the same time. Besides, an effective reward mechanism is also presented for autonomous driving. Finally, by combining behavior cloning and reinforcement learning algorithm, our DRL model can learn an effective policy quickly for path tracking using some easy-to-measure vehicle state parameters and environment information as inputs. The contributions of this paper can be summarized as follows.

(1) A new framework of combining behavior cloning and reinforcement learning algorithm is proposed for autonomous driving work. And we demonstrate that the trained DRL control model based on the new framework can solve some simple driving tasks within 30 episodes and has the generalization ability for different road environments, that is, the framework is data efficient.

(2) Based on Deep Deterministic Policy Gradient (DDPG) algorithm and Actor-Critic structure, a dual actor networks with one chief critic network structure is proposed for the two different action outputs (steering wheel angle and vehicle speed). Based on this structure, we can pick out some more important state information as state inputs for different action outputs (for example, we find the information of the equivalent wheel speed is not so necessary for the generation of steering command by testing, but it is very important for the generation of vehicle speed command). So, it is effective to speed up the model convergence rate and improve the learning effect.

(3) We propose an indirect way to design the reward from tracking precision for autonomous driving. And based on the reward function, the chief critic network can learn how to guide the dual actor networks quickly.

This paper is organized as follows. In Section II, model design for autonomous driving is described. In Section III, the system architecture for autonomous driving, which combines behavior cloning (imitation learning) and reinforcement learning algorithm, is described. In Section IV, *Carsim/Simulink* simulation experiments are designed to train the DRL control model proposed in this paper. In Section V, the test experiment is designed to verify the effectiveness of the trained DRL control model with HIL system. And conclutions are drawn in Section VI.

II. Model Design

In this paper, a reinforcement learning algorithm called Deep Deterministic Policy Gradient, is applied to autonomous driving. Markov Decision Process (MDP) [43] is the theoretical basis of reinforcement learning. Formally, most of RL tasks can be modeled as MDP.

A. Markov Decision Process (MDP)

A Markov Decision Process is a quintuple: $\langle S, A, P, R, \gamma \rangle$. The state–value function $V^{\pi}(s)$ of an MDP is the expected return starting from state s, and its Bellman equation is:

$$V^{\pi}(s) = \sum_{a \in A} \pi\ (a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^{\pi}(s') \right) \tag{1}$$

where $\pi(s)$ is the policy, $P^a_{ss'}$ is the probability of next state being s' conditioned on the fact that the current state s and action taken a, and R^a_s is the mathematical expectation of the reward on the fact that the current state s and action taken a, and γ is the reward discount factor..

The action-value function $q^\pi(s,a)$ is the expected return staring from state s, taking action a, and its Bellman equation is:

$$q^\pi(s,a) = R^a_s + \gamma \sum_{s' \in S} P^a_{ss'} \sum_{a' \in A} \pi\left(a'|s'\right) q^\pi\left(s', a'\right) \quad (2)$$

The solution process of MDP is to find an optimal action policy based on the current state s. The optimal state-value function $V^*(s)$ is the maximum state-value function over all policies:

$$V^*(s) = \max_\pi V^\pi(s) \quad (3)$$

The optimal action-value function $q^*(s,a)$ is the maximum action-value function over all policies:

$$q^*(s,a) = \max_\pi q^\pi(s,a) \quad (4)$$

And policy can be updated as following:

$$\pi(s) = \arg\min_{a \in A} q^\pi(s,a) \quad (5)$$

Markov Decision Process
A Markov Decision Process is a tuple $\langle s, a, p, R, \gamma \rangle$
1. s is a finite set of states.
2. a is a finite set of actions.
3. P is a state transition probability matrix: $P^a_{ss'} = \mathbb{P}\left[s_{t+1} = s' | s_t = s, A_t = a\right]$
4. R is a reward function: $R^a_s = \mathbb{E}\left[R_{t+1} | s_t = s, a_t = a\right]$
5. γ is a reward discount factor $\gamma \in [0, 1]$

B. Autonomous Driving as an MDP

1) State Space s_t: DRL learns the behavior policy by interaction with environment, the selection of state space is the key to guaranteeing the validity of DRL. In this paper, some easy-to-measure vehicle state parameters and environment information are adopted as state space inputs. As we know, there is a lot of kinematic and dynamic information in the running process of a car, but some of the information is unused or has only minor impact on the driving algorithm. So, in this paper, by testing and adjusting, some kinematic and dynamic information is selected as observations for our DRL model (see Table I), and in fact, it is sufficient for simple driving task (see Section IV and Section V).

where L_DRV_i is the lateral distance from the drive target path to the four driver preview points, respectively, $i = 1, 2, 3, 4$(see Fig.1); v_j is equivalent wheel speed of the four wheels, respectively, $j = 1, 2, 3, 4$.

TABLE I
STATE SPACE

meanings	Parameters	Units
steering angle	δ	degree
velocity	v	km/h
lateral distance from the drive target path to the driver preview point	L_DRV_i	m
the equivalent wheel speed	v_j	km/h
vehicle lateral distance to path in CG	lat_veh	m

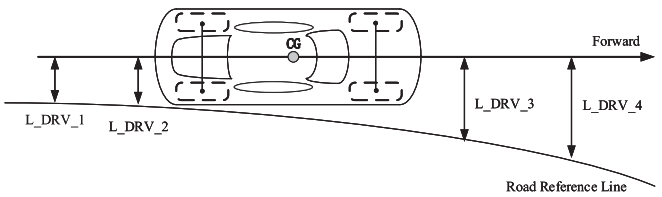

Fig. 1. The lateral distance from the drive target path to the driver preview points.

2) Action Space a_t: Steering angle and vehicle speed are chosen as control inputs of the autonomous vehicle. The most important reason to choose them is that the action space is on a lower dimension which is helpful for reducing the difficulty of training, but still can satisfy the upper control of the autonomous driving. Based on this reason, the two-dimensional action space [steering wheel angle, vehicle speed] is selected, and steering wheel angle is normalized in the range $[-1, 1]$, vehicle speed is normalized in the range $[0, 1]$, and the units are degree and km/h, respectively.

3) Reward Function R_t: The design of reward mechanism is the most important link for shaping the behavior of the autonomous driving policy. There are several methods to design the reward function for DRL, such as sparse reward [29], soft-step function of the form [30], SoID [31], curiosity reward [32], etc. But which is the best reward mechanism for autonomous driving? Some researchers adopted sparse reward for autonomous driving, and reward was obtained by the distance travelled before exiting lane. However, as they themselves said that it was incredibly sparse [26]. In this paper, an indirect way to design the reward function for autonomous driving is proposed. The reward function has two parts: reward from speed and reward from tracking precision. The reward fuction from speed toward the track axis is designed to obtain the positive reward (first term) and the reward fuction from speed deviating from the track axis is designed to obtain the negative reward (second term). And we propose an indirect way to design the reward from tracking precision. The tracking error is not used to design the reward function directly, and we adopt the difference of tracking error between the current state and the next state to design the reward function indirectly. And if the tracking error decreases between the next state and the current state, the reward function will obtain a positive reward, and if the tracking error increases, the reward function will obtain a

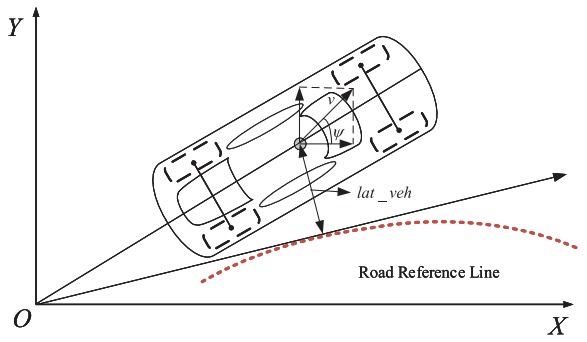

Fig. 2. Design of the rewards.

negative reward.

$$R_t = \omega_0(\underbrace{v|\cos\psi|}_{\text{first term}} - \underbrace{v|\sin\psi|}_{\text{second term}} - \omega_1(\underbrace{|lat\_veh_{t+1}| - |lat\_veh_t|}_{\text{third term}}) \tag{6}$$

where ψ is the heading angle of the vehicle; $\psi = \beta + r$; β is vehicle slip angle; r is the vehicle yaw angle; lat_veh_t and lat_veh_{t+1} are tracking error of the current state and the next state, respectively(see Fig.2).

Remark 1: ω_1 is used to regulate the proportion of reward from tracking precision in total rewards and balance the reward from speed and the reward from tracking precision. And the size of ω_1 is related to the sampling period. ω_0 is used to regulate the range of the reward, which, in fact, is an important parameter for DRL affecting the effectiveness of shaping the networks [39].

III. System Architecture

In this paper, a DRL model is trained for autonomous driving by combining behavior cloning and reinforcement learning algorithm. Actor-Critic structure is adopted in the DRL algorithm, but in order to speed up the model convergence rate and improve the learning effect, a dual actor networks structure is proposed for the two different action outputs (steering wheel angle and vehicle speed). A chief critic network is built to guide the updating process of dual actor networks at the same time. Fig.3 is the system architecture.

Behavior Cloning is a method of learning from demonstrations provided by experts [33]. Suppose we have a series of strategies from experts: $\langle \tau_1, \tau_2, \tau_3, \cdots \tau_n \rangle$, and each strategy contains a series of actions and states:

$$\tau_i = \left\langle s_1^i, a_1^i, s_2^i, a_2^i, \cdots s_n^i, a_n^i \right\rangle \tag{7}$$

Then all the" state-action" pairs can be extracted from the strategies to obtain a new data set:

$$D = \{(s_1, a_1), (s_2, a_2), (s_3, a_3), \cdots (s_n, a_n)\} \tag{8}$$

After that, the Behavior Cloning model can be trained from the new data set D. Finally, the trained model can solve the problems it faces according to imitation experts. But it is hard for direct imitation learning (such as behavior cloning) to obtain a good performance in the process of testing, so it is usually used as the initial strategy for DRL [40].

The algorithm flowchart of the behavior cloning algorithm can be seen in Algorithm 1.

Algorithm 1 (Behavior Cloning)

Learning by Demonstrations:
1. Dataset Aggregation: $\langle \tau_1, \tau_2, \tau_3, \cdots \tau_n \rangle$.
2. state-action: (s_n, a_n).
3. Training: Learning by expert Demonstrations (s_n, a_n).
4. Testing: (s, a) $(Actor) \sim \tau$ $(Expert)$.

A. Reinforcement Learning Algorithm – Deep Deterministic Policy Gradients

Deep Deterministic Policy Gradients (DDPG) is a continuous, model-free deep reinforcement learning algorithm [34], it incorporates the deep learning neural network into Deterministic Policy Gradient (DPG) [35]. DDPG is an off-policy learning algorithm, which means that actions performed during training come from a policy distinct from the learning optimal policy by the actor [26]. The advantages of off-policy are that it can make the agent explore more efficiently and obtain more training specimens without affecting the objective strategy.

Suppose the data is from the strategy χ, the optimization objective function of Off-Policy algorithm is as follows [30]:

$$J(w) = \min e_w E_\chi \left[\frac{1}{2} \left(R_t + \gamma\, Q^w(s_{t+1}, a_{t+1}) - Q^w(s_t, a_t) \right)^2 \right] \tag{9}$$

$$J(\theta) = \max e_\theta E_\chi \left[Q^w(s_t, \mu(s_t)) \right] \tag{10}$$

where Q^w is the Q network; $\mu(s_t)$ is the policy network; w is the parameter of Q^w; θ is the parameter of the policy network, and E is the mathematical expectation.

The purpose of the optimization objective functions is to estimate a Q-optimal policy $\pi(s_t) = argmax_{a_t} Q(s_t, a_t)$ and minimize the temporal difference (TD) error [36]. Here, (s_t, a_t, r_t, s_{t+1}) is an experience tuple. In order to reduce the correlation of the transition data in time series, the design of experience replay memory is applied to DDPG algorithm, and the training data is selected from experience replay memory randomly. More details about this algorithm can be seen in [35].

By solving the optimization objective functions (9)-(10), the following results can be obtained:

$$\Delta w = \alpha E_\chi \left[\left(r_t + \gamma\, Q^w(s_{t+1}, a_{t+1}) - Q^w(s_t, a_t) \right) \nabla_w Q^w(s_t, a_t) \right] \tag{11}$$

$$\Delta_\theta = \alpha E_\chi \left[\nabla_w Q^w(s_t, a_t)\,|_{a_t = \mu_\theta(s_t)}\, \nabla_\theta\, \mu_\theta(s_t) \right] \tag{12}$$

where α is the learning rate.

the DDPG algorithm can be seen in Algorithm 2.

DDPG is a sequential correlation model. In this paper, the design of exploration policy is based on the discrete Ornstein-Uhlenbeck process noise [33], which is a sequential correlation noise. So, at each step the actions noise is added as follows:

$$\hat{a}_t^i = a_t^i + \theta^i * \left(\mu^i - a_t^i \right) + \sigma^i * \epsilon_t^i \tag{13}$$

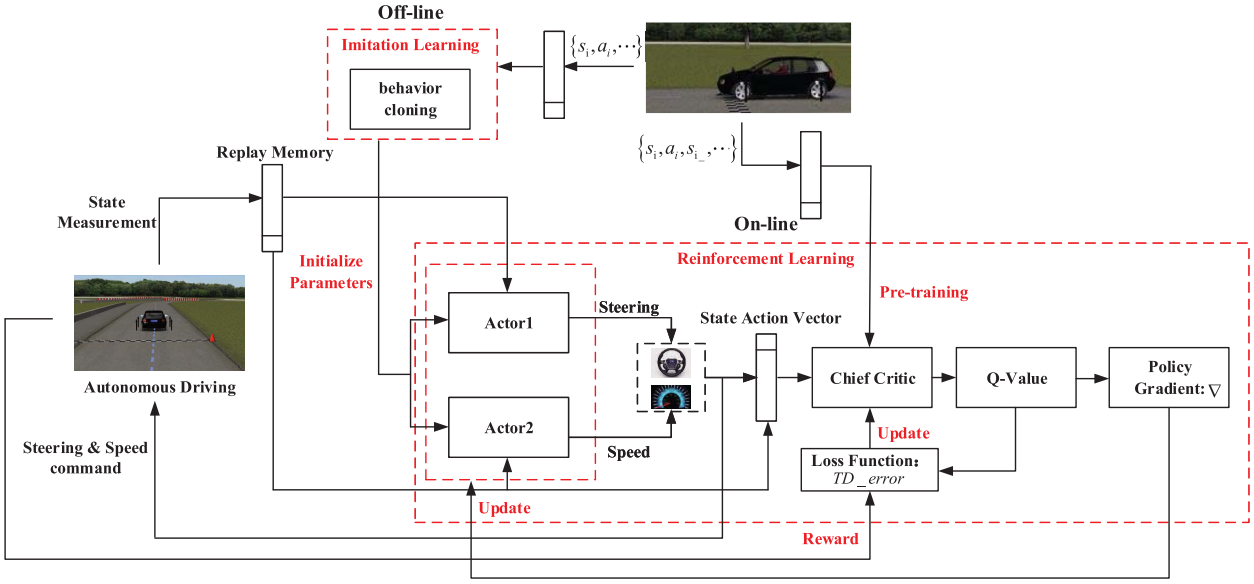

Fig. 3. The system structure diagram for autonomous driving like human beings.

Algorithm 2 (DDPG Algorithm)

Initialize critic network $Q\left(s, a|\theta^Q\right)$ and actor network $\mu\left(s|\theta^\mu\right)$

Initialize target network Q' and μ' with weights:

$\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$

Initialize replay buffer $memory\_batch$

Step 1: select action a_t based on actor network, current states s_t and exploration

Step 2: execute action a_t and observe reward r_t and observe new state s_{t+1}

Step 3: store transition (s_t, a_t, R_t, s_{t+1}) in $memory\_batch$

Training:

Step 4: sample a random $mini\_batch$ of N transitions (s_i, a_i, r_i, s_{i+1}) from $memory\_batch$

Step 5: set $y_i = R_i + \gamma\, Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right)$

Step 6: update critic by minimizing TD_error:

$TD\_error = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2$

Step 7: update the actor policy using the sampled gradient:

$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q\left(a, a|\theta^Q\right)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu\left(s|\theta^\mu\right)|_{s_i}$

Step 8: update the target networks:

$\theta^{Q'} = \tau\theta^Q + (1-\tau)\,\theta^{Q'}$

$\theta^{\mu'} = \tau\theta^\mu + (1-\tau)\,\theta^{\mu'}$

Step 9: $s_t = s_{t+1}$

Step 10: return to Step 1

where $\hat{a}_t$ is the action command which includes the action noise, a_t is the action command, θ, μ, σ are hyperparameters; ϵ_t is Winner Process [34], and i stands for the different action commands (steering wheel angle and vehicle speed), and $i = 1, 2$.

Remark 2: At the beginning of the training, the exploration is very important for providing better state-action space coverage. So, the adjustment of these parameters(θ, μ, σ) should be made very carefully.

IV. Simulation Experiments

A. Pre-Training

1) Implementation Details: In this paper, behavior cloning algorithm is adopted to initialize the Actor network parameters, and the pre-training is also carried out for the chief critic network by offline training at the same time. The training data is collected by the simulation environment built in *Carsim*, and it includes state information (Table I) and action commands. The road environment can be seen in Fig.4 (a), $length = 2327.97m$.

For Behavior Cloning (Actor), firstly, in order to obtain abundant and detailed expert's driving information, the sampling period is set to $T = 0.001s$ and the training dataset consists of a hundred thousand states-action pairs. And for Pre-Training (Critic), the sampling period is set to $T = 0.01s$ and the training dataset consists of a ten thousand states-action pairs. After that, different database design methods are adopted for different tasks. For Behavior Cloning (Actor), the training data is recorded in the form of states-action pairs $D = \langle s_t, a_t\rangle$. And for Pre-Training (Critic), the training data is also recorded in the form of MDP: $\langle s_t, a_t, r_t, s_{t+1}, a_{t+1}\rangle$.

The loss function of Pre-Training for Critic network is defined as:

$$L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2 \tag{14}$$

where $y_i = R_i + \gamma\, Q'\left(s_{i+1}, a_{i+1}\right)$, N is the random sampling data size from replay buffer, Q and Q' are the chief critic network and target chief critic network, respectively. And Q' can be obtained from Q by soft update:

$$\theta^{Q'} = \tau\theta^Q + (1-\tau)\,\theta^{Q'} \tag{15}$$

where τ is the soft replacement factor.

Remark 3: Based on the reward function, the Critic network is trained to judge the actions (a_t) by the difference between

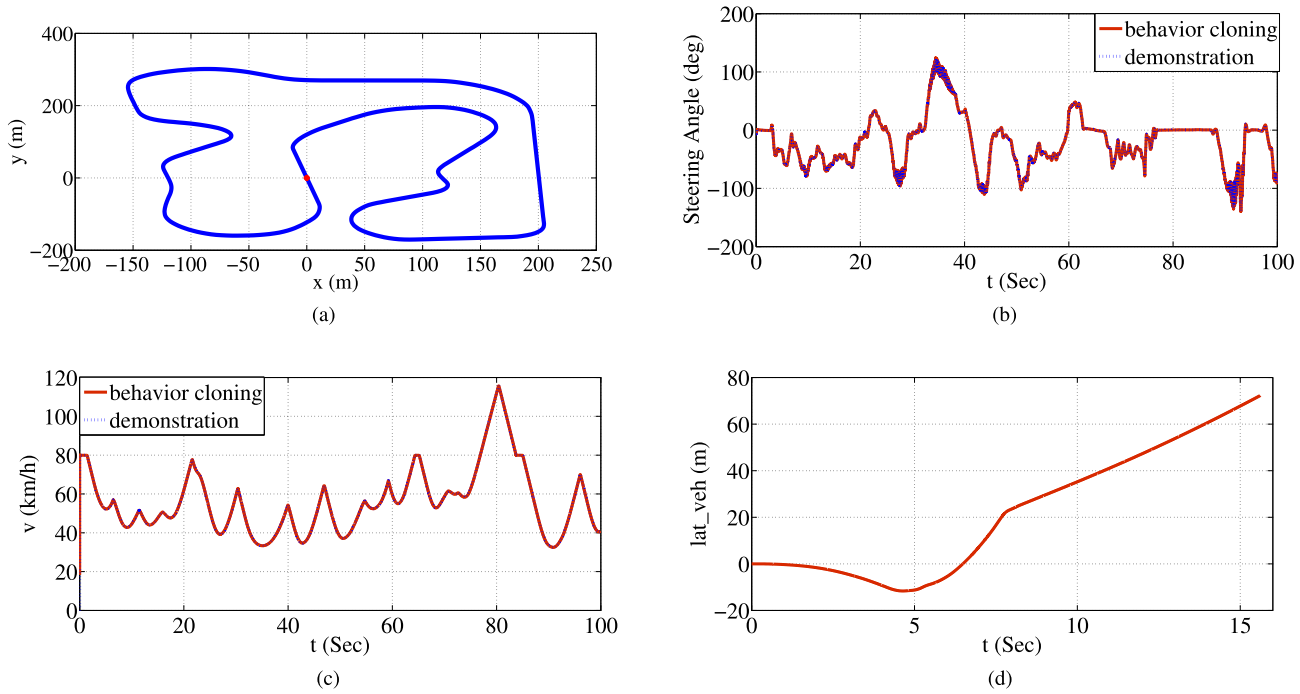

Fig. 4. Behavior cloning results of the Actor networks.(a) Road environments used to data collection in *Carsim*: $length = 2327.97$m, the red point is the start/end point, (b)Steering behavior cloning results for the Expert's Demonstrations, (c) Speed behavior cloning results for the Expert's Demonstrations, (d)Imitation Learning results in the test road environment.

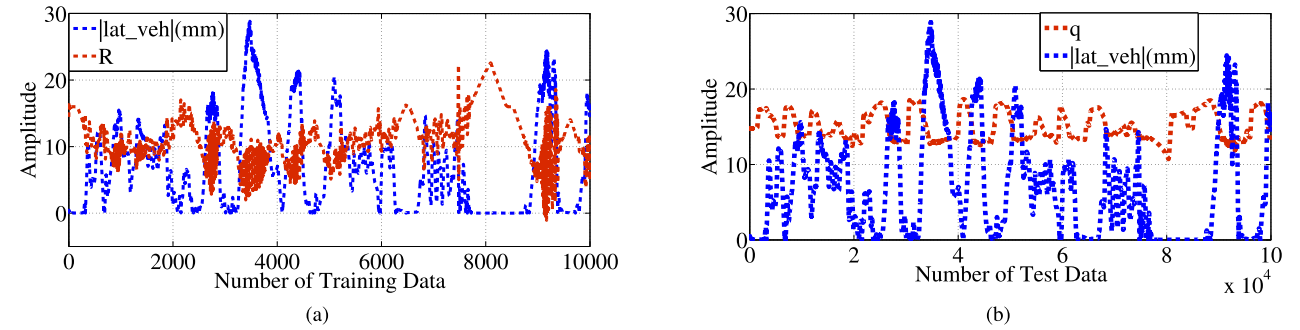

Fig. 5. Pre-Training results of the Chief Critic network. (a) Verification of the Reward function, (b) The test of Chief Critic training effect based on the Expert's Experience.

the current states (s_t) and the next states (s_{t+1}), and in fact, it is bad for the training if the difference is too small. So, the sampling period of data used to train the Critic network should not be too small in order to gain a better training performance.

2) Results: Fig.4 is the Behavior cloning results of the Actor networks. The Actor networks consist of steering network and speed network. Fig.4 (b) (c) is the behavior cloning results for the Expert's Demonstrations, and the learning errors are less than $1 * 10^{-5}$. Fig.4 (d) is the Imitation Learning effect in the test road environment, and from this figure, it can be seen that, the vehicle has gone off the road around 3 seconds, indicating that, adopting the behavior cloning algorithm alone cannot successfully finish this autonomous driving task. Fig.5 is Pre-Training results of the Chief Critic network. From Fig.5 (a) (b), it can be seen that, the rewards R_t from reward function and the q values from the chief critic network change in opposite to the cross track error (the lateral error between the desired trajectory and the actual trajectory). That is, the bigger the value of the cross track error, the smaller the values of the reward R and the q.

Remark 4: As reinforcement-learning based AI systems become more general and autonomous, the design of reward mechanism that produces desired behaviours becomes more important and difficult [39]. But there is not much theorization about the design of reward function. The most DRL algorithms design the reward function on the basis of trial and error. The purpose of this paper is to realize path tracking of autonomous driving with accuracy. So it is stand to reason to verify the efficiency of reward function on the basis of the cross track error.

B. Training by DDPG Algorithm Based on Pre-Training

1) Implementation Details: DDPG adopts the method of experience replay memory ($memory\_batch$) to train model, and the training data ($mimi\_batch$) is selected from experience replay memory randomly. In this paper, the size of

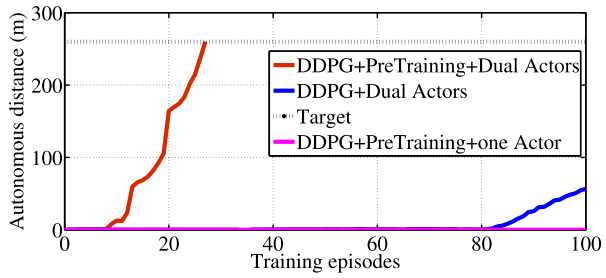

Fig. 6. The training process in the training road environment.

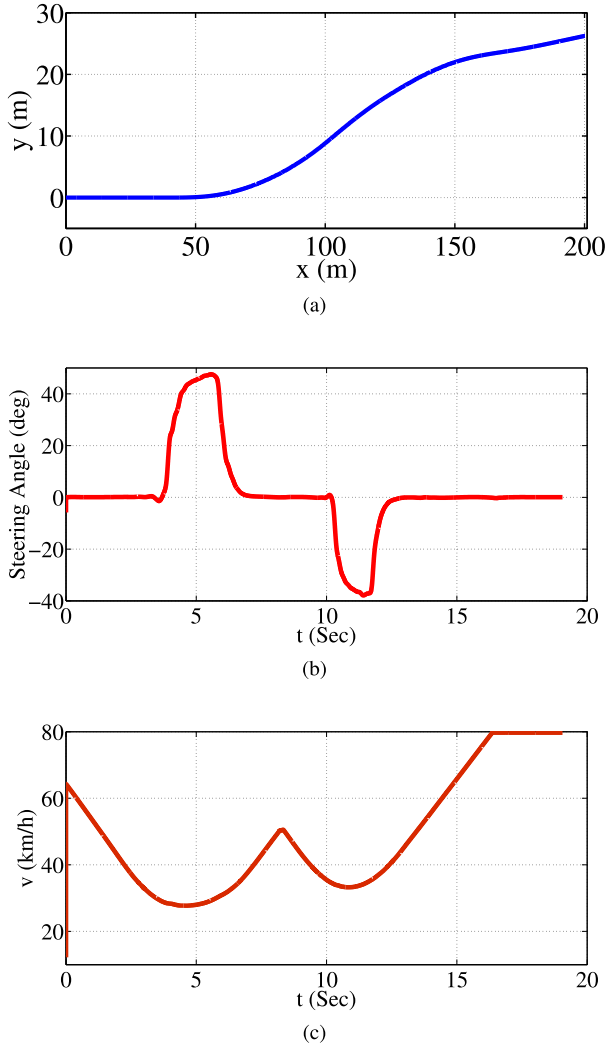

Fig. 7. The final training results in the training road environment.(a) training trajectory, (b) Steering action, (c) Speed action ($\max(Speed) = 80km/h$).

experience replay buffer is a hundred thousand, and the size of training data at each time is 64. And the network is not trained at the beginning, a wide variety of experience is accumulated to fill in the experience replay memory at first. In the process of accumulating experience, the process is terminated if the cross track error is greater than four meters, i.e. $cte > 4m$. In this case, the vehicle has gone off the road (road width: $width = 8m$), it is necessary to reset the vehicle to the position of the road centerline, and then continue the process of accumulating experience until the experience replay memory is full.

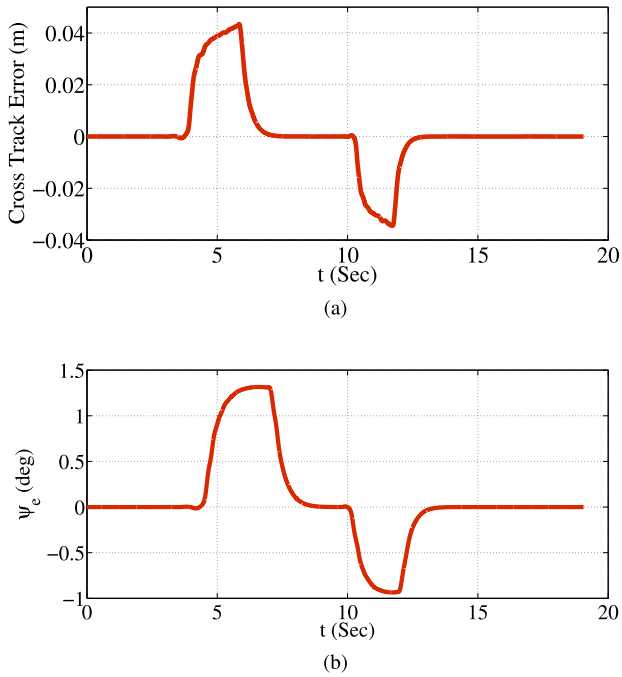

Fig. 8. The final training results in the training road environment.(a) Cross track error, (b) Heading error ($\max(Speed) = 80km/h$).

Finally, the DRL control model is trained online. In order to make the trained model follow the desired trajectory (road centerline) with accuracy, the DRL control model is considered as a failure if the cross track error is greater than 0.1m, i.e. $cte > 0.1m$. The car begins at the start of the road to commence training episodes. The episode is terminated, whenever the DRL control model fail, and then the vehicle is brought back to the road centerline and start a new episode. Besides, the model training is considered to have failed if the DRL model can't achieve the target within the maximum number of episodes ($\max(episodes) = 100$). In this case, it is necessary to adjust the training parameters, and retrain a new DRL model from the beginning.

2) Results: Fig.6 is the training process in the training road environment. In training, a 260-meter road is used and the details of it can be seen in Table II. From Fig.6 and Table III, it can be seen that, compared with the DDPG algorithm with random initialization or one actor, the improved DDPG algorithm with Pre-Training and dual actors proposed in this paper greatly improves the learning efficiency. The DRL control model with pre-training can be trained to realize the target within 30 episodes, and the DDPG with random initialization cannot be trained to realize the target even beyond 100 episodes. Fig.7 and Fig.8 are the results of the final trained DRL control model. From Fig.7(b) (c), it can be seen that the steering actions and the speed actions are very smooth. From Fig.7 (c) and Figs.8 (a)(b), it can be seen that, not only the cross track error and heading error are very small ($\max|cte| < 0.05m$, $\max|\psi_e| < 1.5°$), but also the vehicle maintains a reasonable speed. So, the target of the reward function is realized, which is path tracking of autonomous driving with accuracy, and the autonomous driving vehicle still can maintain a reasonable speed at the same time.

TABLE II
TRAINING AND TEST ROAD ENVIRONMENTS

Training					Test				
Direction	Size	Units	Radius/Curvature	Units	Direction	Size	Units	Radius/Curvature	Units
ahead	50	m	0	1/m	ahead	50	m	0	1/m
left	15	deg	60	m	left	15	deg	60	m
ahead	50	m	0	1/m	ahead	50	m	0	1/m
right	10	deg	80	m	left	10	deg	80	m
ahead	20	m	0	1/m	ahead	20	m	0	1/m
-	-	-	-	-	right	20	deg	120	m
-	-	-	-	-	ahead	20	m	0	1/m

TABLE III
DEEP REINFORCEMENT LEARNING TRAINING AND TEST RESULTS FOR AUTONOMOUS DRIVING

	Training			Test	
model	Episodes	Distance	Units	Station (Go off the road)	Units
Behavior Cloning Model	-	-	-	11.75	m
Deep RL Model+Dual Actors	100	56.57	m	53.63	m
Deep RL Model+PreTraining+Dual Actors	27	260	m	-	-
Deep RL Model+PreTraining+one Actor	100	0	m	0	m

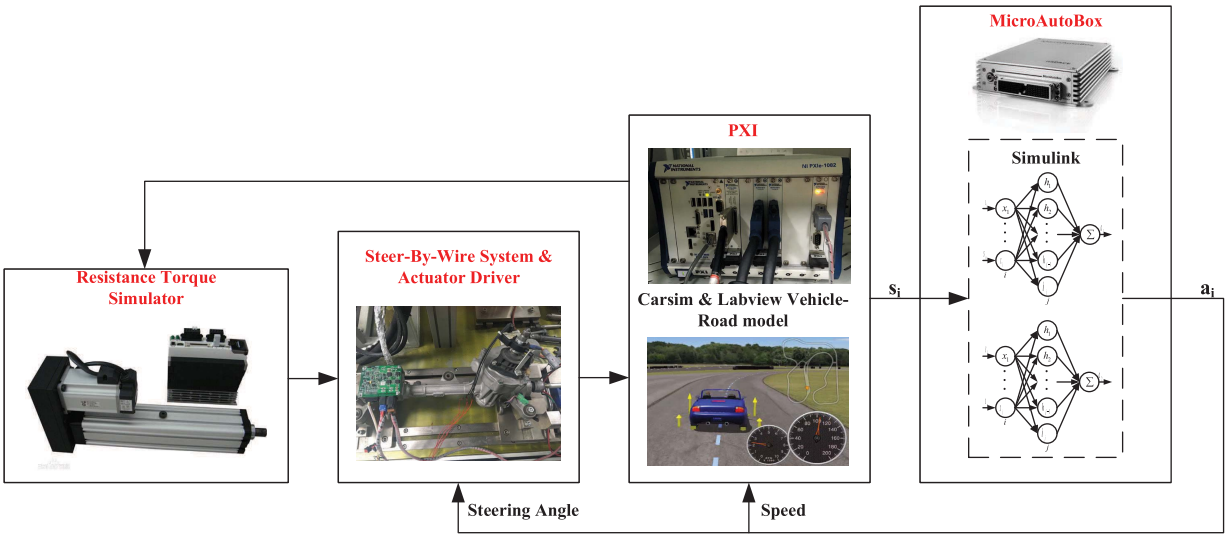

Fig. 9. Implementation architecture of the HIL system.

V. TEST WITH HIL SYSTEM

In this section, Hardware-In-Loop (HIL) simulation and actual measurement are carried out to test the generalization ability and real-time performance of the trained DRL control model.

A. HIL System

The HIL system can be seen in Fig.9. It mainly consists of a DS1501 MicroAutoBox from dSPACE company, a Freescale actuator driver, a steer-by-wire system, a EXLAR electric servo cylinder for steering resistance loading, a BOSCH steering angular sensor, a PXI using LabVIEW from NI company, a power supply, two host PCs and three displayers.

B. Implementation Details

The implementation architecture of the HIL system can be seen in Fig.10. The purpose of the testing is to verify the generalization ability [8] and realtime performance of the trained DRL control model in Section IV. The process of implementation details can be described as follows: firstly, vehicle–road system model from CarSim is encoded into PXI through LabView for real time testing. After that, the vehicle motion states and road information can be obtained from PXI. And based on this, the MicroAutoBox, which has loaded the trained DRL model, calculates the steering angle command for steer-by-wire system, and calculates the speed command for the drive system simulated in *Carsim* at the same time. Then the angle command is sent to the actuator drive to operate the steer-by-wire system, and the angular sensor feeds the values of corresponding steering column angle and angular speed back to nonlinear vehicle model of CarSim. The steering resistance torque, which is obtained from CarSim, acts on the rack of the steer-by-wire system by the EXLAR electric servo cylinder.

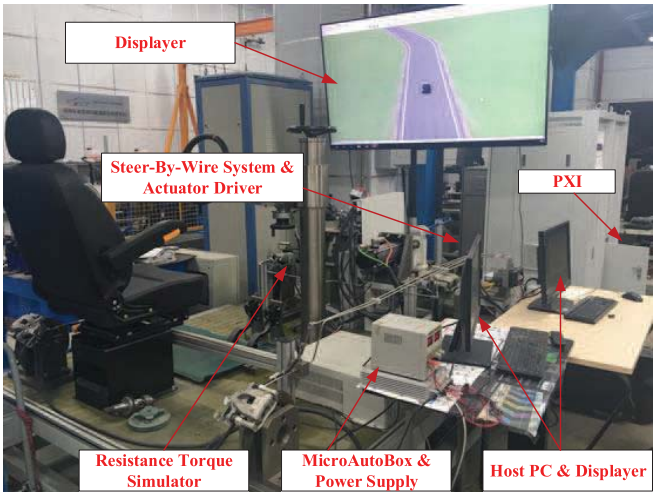

Fig. 10. HIL system for real-time testing.

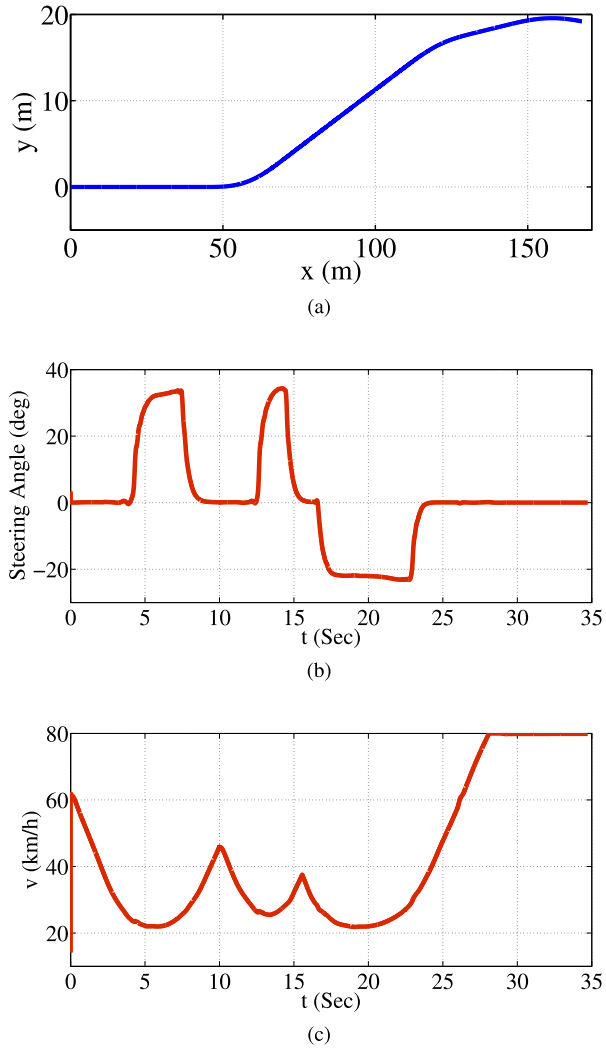

Fig. 11. The test results with HIL system in the test road environment. (a) test strategy, (b) Steering actions, (c) Speed actions ($\max(Speed) = 80km/h$).

C. Results

A more complex road environment is built to validate the generalization ability of the trained DRL control model for different road environments, and the details of this road can be seen in Table II. The test results of different algorithms can be seen in Table III. From Table III, it can be seen that, only the trained DRL control model proposed in this paper can complete the test and does not go off the road. Fig.11 and Fig.12 show the test results of the trained DRL control model proposed in this paper. From Fig.11 (b) (c), it can be seen that, the steering actions and the speed actions are still very smooth. From Fig.12 (a) (b), it can be seen that, the trained DRL control model still can control the vehicle to realize the path tracking with accuracy in the test road environment ($cte < 0.08m$, $\max|\psi_e| < 1\,^\circ$). Although trained and tested in two different road environment, our system still demonstrates reasonably good performance. So, the generalization ability and realtime performance of the trained DRL control model are validated.

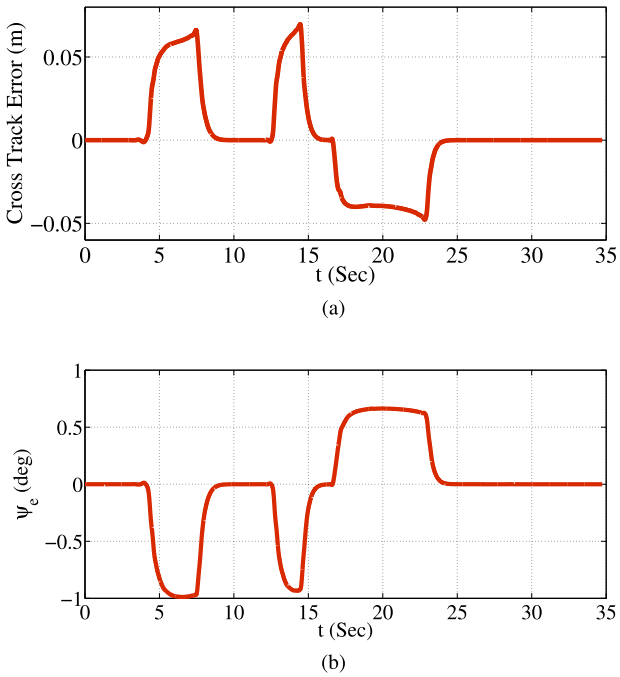

Fig. 12. The test results with HIL system in the test road environment. (a) Cross track error, (b) Heading error ($\max(Speed) = 80km/h$).

VI. Conclusion

In this paper, a new framework for learning to drive autonomously like human beings is proposed, which combines behavior cloning and reinforcement learning algorithms. Firstly, behavior cloning algorithm is adopted to initialize the model parameters. And pre-training is also carried out for the chief critic network by offline training method. After that, we use a continuous, deterministic, model-free deep reinforcement learning algorithm to optimize our model on line. Actor-Critic structure is adopted in the DRL algorithm. Our study proposed a dual actor networks structure for the two different action outputs (steering wheel angle and vehicle speed), and a chief critic network is built to guide the updating process of the dual actor networks. Finally, the DRL control model is trained in simulation experiment, and HIL implementation are used to validate the realtime performance and

TABLE IV
PARAMETERS ADOPTED IN THE DRL ALGORITHM

Name	Parameter	Value
learning rate for actor	lr_a	0.001
learning rate for chief critic	lr_c	0.001
reward discount factor	γ	0.9
noise discount factor	λ	0.9
soft replacement factor	τ	0.01
experience replay memory size	$memory\_batch$	100000
random sampling data size	$mini\_batch$	64
maximum episodes	$max\_episodes$	100
-	ω_0	0.25
-	ω_1	80

TABLE V
ACTIONS NOISE PARAMETERS

Action	θ	μ	σ
Steering Action noise parameters	0.6	0.0	0.3
Speed Action noise parameters	0.6	0.0	0.4

generalization ability of the trained DRL control model. The work we have done shows the following conclusions:

(a) This paper shows the successful application of DRL for the autonomous driving. The results of simulation and experiment show that the trained DRL control model can realize the path tracking control of autonomous driving with accuracy. Besides, the generalization ability and realtime performance of the trained DRL control model are also validated.

(b) Compared with the DDPG algorithm based on random initialization, the improved DDPG algorithm with pre-training proposed in this paper greatly improves the learning efficiency in the process of training. And the dual actor networks with one chief critic network structure is effective to speed up the model convergence rate and improve the learning effect. And the results of simulation and experiment show that the cross track error is perfectly kept close to zero, which validate the effectiveness of indirect way to design the reward from tracking precision for autonomous driving, at the same time, the vehicle can maintain a reasonable speed. Besides, the steering actions and the speed actions are very smooth.

Actually, we mainly focus on the problem of path following to verify the performance of the DRL framework proposed in this paper. In the future, the passenger comfort should be considered, otherwise, the problem of vehicle collision avoidance should be studied and the information of the unpredicted obstacle and the uncertainties on the road should be considered to train the DRL model.

APPENDIX A

The method of dual actor networks adopted in this paper is similar to Multi-agent RL (MARL)[41], but both of them share a common goal which is to gain bigger q_value from chief critic. In the structure of Actor-Critic, the Critic is called Chief Critic which is relative to the single actor networks. The DDPG algorithm's pivotal ideas can be divided into two parts: 1) the chief critic network learns to judge the pairs of State-Action $\langle s_i, a_i \rangle$ (a_i is from the dual actor networks) based on the reward function, and generates the corresponding q_value. 2) the dual actor networks are trained to gain bigger q_value from chief critic.

APPENDIX B

The parameters adopted in the DRL algorithm can be seen in Table IV, and noise parameters of the discrete Ornstein-Uhlenbeck process noise can be seen in Table V. And in the process of experience, these noise parameters θ, μ, δ can be adjusted until we get a reasonable exploration policy.

REFERENCES

[1] C. M. Martinez, M. Heucke, F.-Y. Wang, B. Gao, and D. Cao, "Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 666–676, Mar. 2018.

[2] Y. Rasekhipour, A. Khajepour, S.-K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1255–1267, May 2017.

[3] J. Yoo and R. Langari, "A predictive perception model and control strategy for collision-free autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 11, pp. 4078–4091, Nov. 2019.

[4] S.-Y. Oh, J.-H. Lee, and D.-H. Choi, "A new reinforcement learning vehicle control architecture for vision-based road following," *IEEE Trans. Veh. Technol.*, vol. 49, no. 3, pp. 997–1005, May 2000.

[5] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in aneural network," Comput. Sci. Dept., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. AIP-77, 1989.

[6] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-roadobstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2005.

[7] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2722–2730.

[8] M. Bojarski *et al.*, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*. [Online]. Available: http://arxiv.org/abs/1604.07316

[9] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," 2016, *arXiv:1612.01079*. [Online]. Available: http://arxiv.org/abs/1612.01079

[10] F. Codevilla, M. Muller, A. Lopez, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–9.

[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 39, pp. 1–40, 2016.

[12] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, nos. 4–5, pp. 421–436, Apr. 2018.

[13] D. Kalashnikov *et al.*, "QT-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," 2018, *arXiv:1806.10293*. [Online]. Available: http://arxiv.org/abs/1806.10293

[14] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[15] L. Espeholt *et al.*, "IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures," 2018, *arXiv:1802.01561*. [Online]. Available: http://arxiv.org/abs/1802.01561

[16] V. Zambaldi *et al.*, "Relational deep reinforcement learning," 2018, *arXiv:1806.01830*. [Online]. Available: http://arxiv.org/abs/1806.01830

[17] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.

[18] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *Comput. Sci.*, vol. 8, no. 6, pp. 1–14, Sep. 2015.

[19] M. Riedmiller, M. Montemerlo, and H. Dahlkamp, "Learning to drive a real car in 20 minutes," in *Proc. Frontiers Converg. Biosci. Inf. Technol.*, 2007, pp. 645–650.
[20] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2034–2039.
[21] V. Behzadan and A. Munir, "Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles," 2018, *arXiv:1806.01368*. [Online]. Available: http://arxiv.org/abs/1806.01368
[22] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," 2017, *arXiv:1704.03952*. [Online]. Available: http://arxiv.org/abs/1704.03952
[23] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," 2018, *arXiv:1805.00983*. [Online]. Available: http://arxiv.org/abs/1805.00983
[24] K. Makantasis, M. Kontorinaki, and I. Nikolos, "A deep reinforcement learning driving policy for autonomous road vehicles," 2019, *arXiv:1905.09046*. [Online]. Available: http://arxiv.org/abs/1905.09046
[25] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," 2016, *arXiv:1609.05143*. [Online]. Available: https://arxiv.org/abs/1609.05143
[26] A. Kendall *et al.*, "Learning to drive in a day," 2018, *arXiv:1807.00412*. [Online]. Available: http://arxiv.org/abs/1807.00412
[27] T. Hester *et al.*, "Deep Q-learning from demonstrations," 2017, *arXiv:1704.03732*. [Online]. Available: http://arxiv.org/abs/1704.03732
[28] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," 2017, *arXiv:1708.05866*. [Online]. Available: http://arxiv.org/abs/1708.05866
[29] M. Vecerik *et al.*, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017, *arXiv:1707.08817*. [Online]. Available: http://arxiv.org/abs/1707.08817
[30] P. Wawrzynski, "Learning to control a 6-degree-of-freedom walking robot," in *Proc. Int. Conf. Comput. Tool*, 2007, pp. 698–705.
[31] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," 2017, *arXiv:1707.05300*. [Online]. Available: http://arxiv.org/abs/1707.05300
[32] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," 2017, *arXiv:1705.05363*. [Online]. Available: http://arxiv.org/abs/1705.05363
[33] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Comput.*, vol. 3, no. 1, pp. 88–97, May 1991.
[34] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *Comput. Sci.*, vol. 8, no. 6, pp. 88–97, Jul. 2016.
[35] D. Silver *et al.*, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Int. Conf. Mach. Learn.* 2014, pp. 387–395.
[36] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1928–1937.
[37] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian-motion," *Phys. Rev.*, vol. 36, no. 3, pp. 823–841, Sep. 1930.
[38] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*. Hoboken, NJ, USA: Wiley, 1997.
[39] D. Dewey, "Reinforcement learning and the reward engineering principle," in *Proc. AAAI Spring Symp.*, 2014, pp. 13–16.
[40] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 293–321, May 1992.
[41] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.
[42] X. Zhang and H. Ma, "Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations," 2018, *arXiv:1801.10459*. [Online]. Available: http://arxiv.org/abs/1801.10459
[43] A. G. Joseph and S. Bhatnagar, "An incremental off-policy search in a model-free Markov decision process using a single sample path," *Mach. Learn.*, vol. 107, no. 6, 2018, pp. 969–1011, Feb. 2018.

Yantao Tian received the M.Sc. and Ph.D. degrees from the Jilin University of Technology, in 1987 and 1993, respectively. He is currently a Professor with the Department of Control Science and Engineering, Jilin University.

His research interests include complex systems, distributed intelligent system and network control, intelligent robot control systems, pattern recognition, and machine vision.

Xuanhao Cao received the B.S. degree from Jilin University in 2017, where he is currently pursuing the Ph.D. degree in control science and engineering.

His main research interest includes intelligent vehicle modeling and control.

Kai Huang received the B.S. and M.S. degrees from the College of Communication Engineering, Jilin University, in 2017 and 2020, respectively.

He is currently working with the China Electronics Technology Group Corporation. His research interest includes complex system modeling methods, analysis, and control.

Cong Fei received the B.S. degree from the College of Automotive Engineering, Jilin University, in 2018. He is currently pursuing the M.S. degree with the School of Vehicle and Mobility, Tsinghua University, China.

His research interests include vehicle dynamics and control, and advanced steering system technology.

Zhu Zheng received the M.S. degree in automotive engineering from the College of Mechatronics and Vehicle Engineering, Chongqing Jiaotong University, China, in 2020.

His research interest includes vehicle dynamics and control.

Xuewu Ji received the B.S., M.S., and Ph.D. degrees in automotive engineering from the College of Automotive Engineering, Jilin University, China, in 1987, 1990, and 1994, respectively.

He is currently an Associate Professor with the State Key Laboratory of Automotive Safety and Energy, Tsinghua University, China. His research interests include vehicle dynamics and control, and advanced steering system technology.

Dr. Ji received the National Science and Technology Progress Award for his achievements in the industrialization of electric power steering technology in 2014.