Sensors and Materials, Vol. 37, No. 6 (2025) 2241–2256
MYU Tokyo

S & M 4053

Design of Deep-reinforcement-learning-based Automatic Vehicle Parking Algorithm

Sijie Qiu,[1] Chi-Hsin Yang,[1*] Hao Gao,[1] Zhonghu Huang,[2] and Lina Xue[2]

[1]School of Mechanical and Electric Engineering, Sanming University, Sanming, Fujian Province 365004, China
[2]School of Big Data and Artificial Intelligence, Maanshan University, Maanshan, Anhui Province 243100, China

(Received August 5, 2024; accepted May 23, 2025)

Keywords: actor–critic network, deep reinforcement learning, reward function, automatic vehicle parking

Automatic vehicle parking is an advanced technology that significantly improves road traffic efficiency and reduces congestion compared with manual driving. On the basis of machine learning technologies, in this study, we introduce a new deep-reinforcement-learning-based automatic vehicle parking algorithm in designated free spaces, using the framework of the actor–critic network. The proposed study's principal features are as follows. (1) The provided system uses a single front camera of the vehicle as the primary sensor, simplifying the necessary vehicle sensing systems. A pretrained operator network receives the acquired picture data and uses it to create an action plan based on the vehicle's present condition. (2) A reinforcement learning approach based on both critic and actor networks for training autonomous parking schemes is developed. (3) Formulation rules for an autonomous parking task-specific reinforcement learning environment, including creating parking environments, defining states, designing action spaces, and creating reward functions, are provided. It also incorporates a neural network structure suitable for automatic parking with a residual network module as its main feature to meet safety and reliability requirements.

1. Introduction

Autonomous driving epitomizes an advanced technology with the potential to markedly enhance road traffic efficiency and mitigate congestion as opposed to manual driving. Furthermore, it offers unparalleled convenience for individuals unable to operate a vehicle. The primary aim of autonomous driving is to enable vehicles to function with minimal or no human intervention. Given the dynamic and intricate nature of road environments, the implementation of an intelligent autonomous driving decision system is imperative.

The current decision-making methods for autonomous driving are grounded in a rule-based approach that necessitates the manual formulation of driving policies.[1,2] However, this deterministic methodology is inclined to producing less than optimal decisions owing to the inherent bias of pre-established models within highly stochastic environments and their restricted adaptability to varied scenarios. On the other hand, many researchers have employed

*Corresponding author: e-mail: 20190207@fjsmu.edu.cn
https://doi.org/10.18494/SAM5272

ISSN 0914-4935 © MYU K.K.

an imitation learning technique to develop a driving policy based on data acquired from an expert driver.[3,4] Nonetheless, imitation learning is confronted with several constraints as the acquired policy is limited to the expertise of the expert driver, who typically only exhibits commendable driving behavior. As a result, the model may encounter challenges in addressing risky situations. Additionally, implementing imitation learning in a real-world setting entails significant costs and time investment.

The car planning system encounters an intricate challenge in multi-objective planning owing to its expansive state space, which encompasses an array of potential scenarios. To tackle this complexity, it becomes imperative to leverage reinforcement learning for optimizing the system's performance. This approach enables the determination of the most proficient combination of strategies by employing an advanced search strategy[5] that systematically explores the vast state space and identifies the best action to take in each situation. Furthermore, the application of reinforcement learning in the development of a car transfer planning system for parking facilities adds value to the solution. This system not only learns from the current state of the environment but also anticipates future states, allowing it to make informed decisions that facilitate efficient parking lot utilization. By implementing such a reinforcement-learning-based system, the issue of automatic vehicle parking can be effectively addressed, streamlining the parking process and enhancing user experience.

In Ref. 6, the tracking of an unmanned surface vehicle in a complex system is optimized using a reinforcement learning control algorithm, enhancing performance and accuracy. Wang *et al*.[7] presented a self-learning model-free solution for the optimal control of an unmanned surface vehicle. Using directly trained spiking neural networks, Liu *et al*.[8] demonstrated best-in-class performance on several Atari games, suggesting that deep reinforcement learning can be used to achieve human-level control. Additionally, in Ref. 9, the behaviorally critical augmented learning mechanisms are combined with finite time control techniques to optimize the tracking of an unmanned surface vehicle. Additionally, Li *et al*.[10] reported an advanced autonomous driving integration method based on end-to-end multi-agent deep reinforcement learning, which is capable of autonomously learning complex and realistic traffic dynamics. Lastly, a deep-reinforcement-learning-based approach described in Ref. 11 allows unmanned aerial vehicles to move randomly and dynamically through multi-obstacle situations.

Recently, Guo *et al*.[12] described the development of a system for planning car transfers in parking facilities using reinforcement learning methodologies. This advanced parking management system incorporates independent decision-making, intricate path planning, and efficient resource allocation through a Markov decision process and a dynamic planning-based reinforcement learning algorithm. A hierarchical trajectory planning method[13] using deep reinforcement learning in the federated learning scheme was developed to rapidly and accurately generate collision-free automated parking trajectories in multiple narrow spaces while safeguarding the privacy of vehicle data during model parameter aggregation. In Ref. 14, a framework was introduced for autonomous parking maneuvers in complex scenarios, involving trajectory planning and collision avoidance. The planned parking maneuvers were tracked using innovative controllers to operate a 14-degree-of-freedom vehicle simulator.

In this study, we introduced a novel deep reinforcement learning algorithm that utilizes the framework of the actor–critic network[15,16] to enable automatic parking in designated free

parking spaces. When applied to the task, the algorithm captures image data from the vehicle's front camera at predefined intervals, which serves as descriptive input for the current vehicle state. This information is then fed into a pretrained operator network, which formulates an action execution strategy based on the vehicle's present state. The vehicle subsequently executes operation actions in accordance with this strategy and iterates through the control program until successfully completing the automatic parking task, as depicted in Fig. 1.

The trained actor network, as depicted in Fig. 1, underwent training using a reinforcement learning algorithm that integrates both critic and actor components. This approach leverages two deep learning neural networks, with input data comprising images captured by the vehicle's front camera. However, there is a disparity in the output of these networks, as illustrated in Fig. 2 during the overall training process. The critic network primarily provides an estimation of the evaluation index for current vehicle execution actions to assess the automatic parking efficacy, while the actor network devises action execution strategies based on the vehicle's present state. Following this strategy from the actor network, actions are executed by the vehicle and relevant environmental interaction information is recorded to form corresponding training sample sequences. These sequences are then randomly selected through experience replay methods. Then, based on the critic network's feedback, parameter changes are made to train both actor–critic networks concurrently until the automatic parking tasks are successfully completed, which signifies the end of our algorithm's training phase.

The reinforcement learning algorithm, which combines the critic and actor networks, primarily captures changes in vehicle state through image information including, but not limited to, velocity, acceleration, and orientation. Subsequently, it determines an optimal action

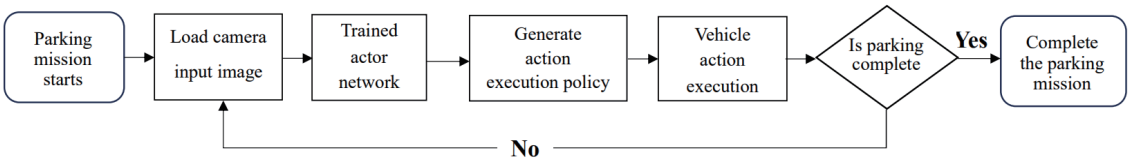


Fig. 1. Execution process of automatic vehicle parking algorithm.

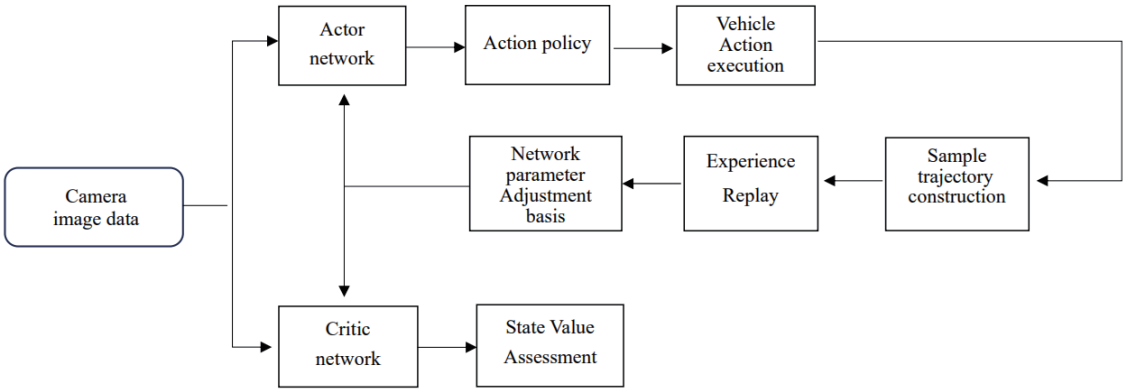


Fig. 2. Network training of automatic vehicle parking algorithm.

execution strategy based on the current vehicle states, taking into account the vehicle's surroundings, such as obstacles and available parking spaces. This enables dynamic planning of the vehicle's parking path, ensuring efficient and safe parking, and facilitates motion control through the action execution strategy, which includes braking, steering, and throttle adjustments.

Consequently, in this study, we offer significant contributions and innovations across multiple dimensions. The algorithm's characteristics are primarily evident in three key points.

(1) First, it utilizes a single front camera as the primary sensor, significantly reducing the complexity of the required vehicle sensing system compared with conventional automatic parking systems based on radar and camera multisensor fusion.
(2) Second, it introduces a reinforcement learning algorithm based on the critic network combined with the actor network for automatic parking scheme training, eliminating the need for the manual extraction of image features and overcoming the limitations of deep learning methods that require extensive dataset establishment beforehand by enabling autonomous end-to-end neural network training.
(3) Third, it formulates rules for a reinforcement learning environment specific to automatic parking, encompassing the construction of the parking environment, state definition, action space design, and reward function formulation. It also incorporates a neural network structure suitable for automatic parking tasks with a residual network module as its main structural feature to meet safety and reliability requirements.

This paper is organized into the following sections. In Sect. 2, we provide an overview of the development of a vehicle-driving simulation environment tailored for the deep-reinforcement-learning-based automatic parking algorithm, as well as an extensive examination of the training process for automatic parking using actor–critic networks. In Sect. 3, a comprehensive presentation of experiments and validation of trained-network performance is followed by a detailed discussion on the limitations of the proposed method. Finally, a concise summary encapsulates the principal findings of this study.

2. Design of Algorithm Based on Deep Reinforcement Learning

In this section, we delve into the intricate design nuances of an automatic parking algorithm, which is crafted using deep reinforcement learning within the Car Learning to Act (CARLA) vehicle driving simulator.[17] As illustrated in Figs. 3 and 4, CARLA's efficacy is showcased through various simulations. The CARLA vehicle driving simulator not only facilitates the development, training, and validation of highly complex urban autonomous driving systems but also offers adaptable configuration options for sensing suites and environmental conditions, thereby enhancing the robustness and versatility of the simulated systems.

The initial phase involves the execution of automatic parking environment design, which includes the construction of a virtual environment, the delineation of action space, the determination of a reward function, and the development of a deep neural network. Subsequently, the training methodology for the actor network within the reinforcement learning framework is expounded upon, ultimately culminating in a comprehensive automatic parking algorithm flow.

Fig. 3. (Color online) CARLA vehicle driving simulator.

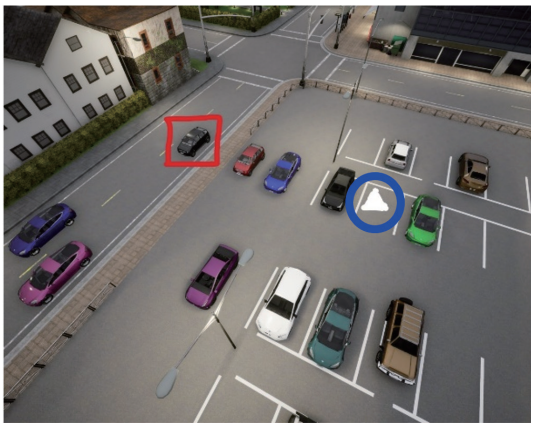


Fig. 4. (Color online) Initial environment of automatic parking.

2.1 Automatic parking environment design

In this section, we present the reinforcement learning environment design for the deep-reinforcement-learning-based automatic parking algorithm. The environment utilizes CARLA to simulate a virtual driving environment and will provide detailed insights into the action space, reward function, and adopted deep neural network structure. Initially, the vehicle's surrounding environment is established in the CARLA vehicle driving simulator using a preformulated parking lot map as the training ground. The vehicle's initial position is on the main road adjacent to the parking lot, indicated by a red rectangle in Fig. 4. The objective is to maneuver into an available parking space marked with a white triangle, as depicted in Fig. 4.

We introduced a diverse array of vehicles, varying in color and model, onto the designated route for vehicular travel. The rationale behind this will be expounded upon in the discussion of experimental results. The algorithm employs a discrete action space, encompassing not only the standard forward action but also a reverse action tailored for automatic parking tasks. This includes two driving directions (forward and backward) as well as five steering wheel angles (90 degrees to the left, 45 degrees to the left, neutral, 45 degrees to the right, and 90 degrees to the right), as shown in Fig. 5.

Each action is represented as a tuple,

$$action = \left(orientation, angle\right), \tag{1}$$

where *orientation* denotes the driving direction (either forward or backward) and *angle* represents the steering wheel angle using values.

$$angle = \{-90 + 45 * i \mid i = 0,1,2,3,4\} \tag{2}$$

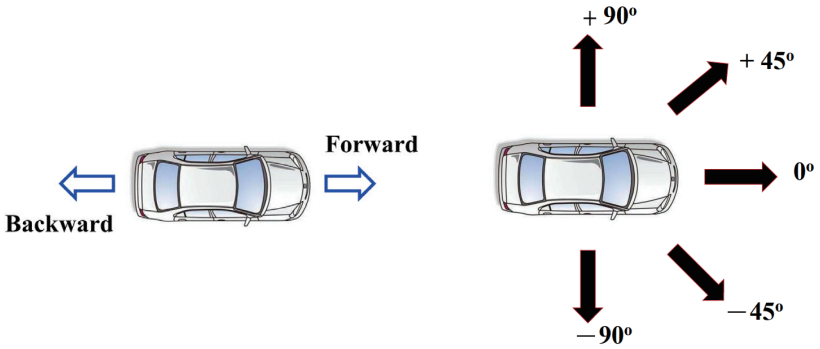


Fig. 5. (Color online) Actions for automatic parking task. (a) Two driving directions. (b) Five steering wheel angles.

Braking actions are intentionally excluded from the action space owing to their potential interference with training processes. If they are included during training sessions, vehicles may become stationary prematurely, which would impede data collection and delay progress.

The algorithm employs the image captured by the front RGB camera as the observation information. The image data is input into the actor network to determine the probability of selecting different actions under the current vehicle state, and action selection is based on the strategy output by the network. Additionally, the global navigation satellite system (GNSS) sensor is utilized to provide real-time longitude and latitude coordinates of the vehicle in a virtual map for describing its pose. This feature description does not affect input data of the deep neural network. It serves solely as a basis for determining immediate action reward and whether to complete a parking task during training.

Since autonomous parking necessitates the vehicle to be precisely positioned within the demarcated lines of the parking space, two GNSS sensors are integrated into the vehicle. The vehicle's map coordinates returned by these sensors upon achieving the specified pose in the parking space are recorded as (LA_1, LO_1) and (LA_2, LO_2). Here, LA_i denotes the latitude coordinates of the i-th sensor at its final position, whereas LO_i represents its longitude coordinates. When each sensor's feedback aligns closely with its corresponding coordinate pair, it signifies the successful completion of autonomous parking. Given that collision-free driving is imperative for autonomous parking operations, a reward function should encompass penalties for collisions and incentives for attaining parking objectives.

In this study, the difference in the state-based potential function is utilized as the criterion for designing the reward function. The state-based potential function is defined as the Euclidean distance between the geodetic coordinates obtained from GNSS measurements at the current position of the vehicle and those at its end position.

$$\Phi(s_t) = (la_1^t - LA_1)^2 + (lo_1^t - LO_1)^2 + (la_2^t - LA_2)^2 + (lo_2^t - LO_2)^2 \tag{3}$$

Here, la_i^t denotes dimensional information provided by the i-th GNSS sensor at time t, whereas lo_i^t represents longitude information from that sensor at time t. LA_i and LO_i denote latitude and

longitude coordinates of that sensor at its end position, respectively. It can be observed that a larger positional disparity between the vehicle and its end point results in a correspondingly augmented state-based potential function. Therefore, we define our basic reward function as being equal to the difference between the state-based potential function at time t and the next one at time $t + 1$.

The reward function R_{Φ}^{t} is defined as the discrepancy of the potential function at time t by

$$R_{\Phi}^{t} = \Phi\left(s_t\right) - \Phi\left(s_{t+1}\right). \tag{4}$$

It encompasses penalties for collisions and rewards for task completion, resulting in the final reward function expressed as

$$r_t = kR_{\Phi}^{t} + R_{collision} + R_{goal}, \tag{5}$$

where $R_{collision}$ represents the penalty for collisions and takes a value of 0 when no collision occurs and –2 when a collision occurs. R_{goal} denotes the reward for completing automatic parking, which is a value of +5 upon task completion. The proportional coefficient k is utilized to scale R_{Φ}^{t} to an appropriate range.

In this algorithm, a deep neural network is employed as both the actor and critic networks. The network architecture is depicted in Figs. 6 and 7, where '*s*' represents the stride of the operation and '*p*' denotes the padding operation. The output of the actor network is a one-dimensional action strategy vector with dimensions of 10 × 1 × 1, wherein each element signifies the probability distribution for selecting its corresponding action. The initial five actions correspond to forward vehicle movement, while the subsequent five actions pertain to backward vehicle motion. Each set of actions includes steering wheel angles at 90 degrees left, 45 degrees left, neutral position, 45 degrees right, and 90 degrees right.

The network structure shown in Fig. 7 represents the critic network, with its output being a specific value that signifies the current state value of the vehicle. The deep neural network utilized in this algorithm continues to employ the residual network structure for preserving shallow features while also incorporating the leaky rectified linear unit (ReLU) activation function. As for the actor network, its output denotes the preference for different actions in the current state, which is then normalized by the softmax function to derive the action policy in the said state. In contrast, a fully connected layer is employed by the critic network to produce a singular value representing the value function of the current state.

The algorithm initializes the parameters of the neural network to ensure that the initial output of the network is a small value. The weight coefficients in the convolution kernel and fully connected layer are generated from a normal random distribution with a mean of 0 and a variance of 0.05, while the corresponding bias term coefficients are all initialized to zero. This initialization method results in an initial action execution strategy that approximates equal probability for action selection, and an approximate state value estimate of 0 for the vehicle.

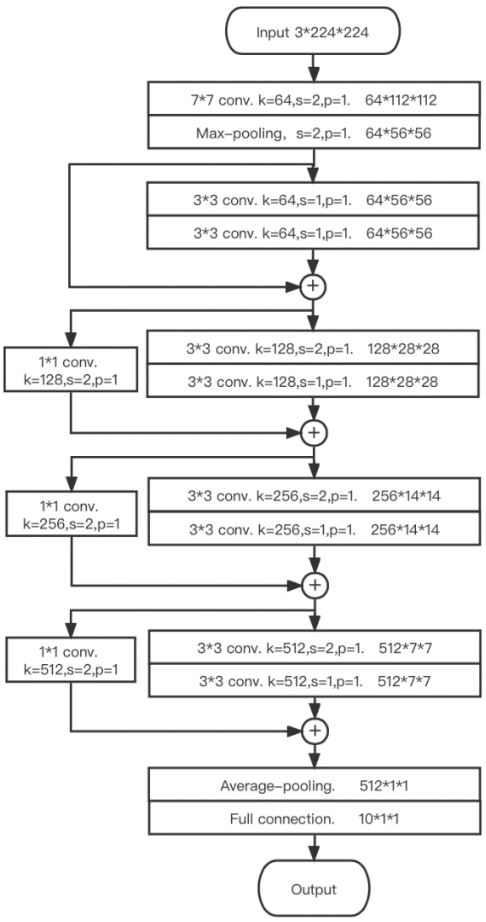


Fig. 6. (Color online) Architecture of actor network.

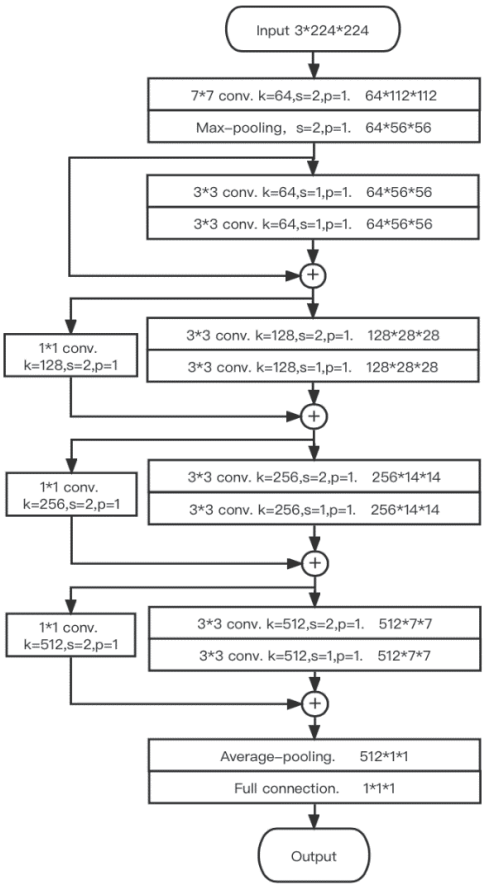


Fig. 7. (Color online) Architecture of critic network.

Consequently, every action selection during vehicle training improves, and state estimation does not unduly impact network convergence.

Up to this point, we have established the vehicle parking environment, the design of the action space, the formulation of the reward function, and the determination of the deep neural network structure. These elements correspond to fundamental components within a reinforcement learning environment. A comprehensive explanation of the overall algorithm training process will be provided in the subsequent section.

2.2 Training of actor–critic network

The actor–critic network seamlessly integrates the value-function-based and policy-based reinforcement learning algorithms. It enables the concurrent generation of action policies while completing the estimation of the state value function. In this subsection, we outline the actor–critic-based deep reinforcement learning algorithm flow employed in the automatic parking task. To mitigate variance in the training iteration process, this algorithm introduces a baseline

concept that is exclusively tied to the current vehicle state and remains invariant with respect to selected vehicle actions within that state. The merits and demerits of actions are further elucidated through comparison between estimations of vehicle action value functions and baselines. In accordance with the policy gradient theorem,[18,19] adjusting network parameters θ along specific gradient directions augments expected returns from network generation policies.

$$\begin{aligned}\nabla E_{\pi(\theta)}[G_0] &= E\left[\nabla v_{\pi(\theta)}(s_0)\right] = \sum_{t=0}^{+\infty} E\left[\sum_a \gamma^t q_{\pi(\theta)}(s_t, a)\nabla\pi(a|s_t;\theta)\right] \\ &= \sum_{t=0}^{+\infty} E\left[\sum_a \gamma^t (q_{\pi(\theta)}(s_t, a) - b(s_t))\nabla\pi(a|s_t;\theta)\right]\end{aligned} \tag{6}$$

The value baseline $b(s_t)$ pertains solely to the current state. The discount rate γ is applied to action rewards. The action a_t is executed by the vehicle at time t, whereas s_t denotes the vehicle's state at that time. G_t represents the cumulative benefit of actions available to the vehicle at time t. The policy $\pi(*;\theta)$ outputs an execution policy for actions through the executor network. $v_{\pi(\theta)}(s_0)$ signifies the state value function of the vehicle in state s_0 obtained by executing policy $\pi(\theta)$. $q_{\pi(\theta)}(s, a)$ represents the corresponding action value function.

When updating the algorithm parameters with the state baseline, the network parameters are adjusted in accordance with the following equation:

$$\theta_{t+1} \leftarrow \theta_t + \alpha(G_t - \mathrm{b}(\mathrm{s_t}))\nabla\ln(\pi(a_t \mid s_t;\theta)), \tag{7}$$

where θ_t represents the actor network parameters at time t and α is the learning rate. The introduction of the state baseline transforms the parameter adjustment from being solely based on vehicle action benefit to considering the relative advantage of actions compared with the baseline. Consequently, the vehicle state value function is typically designated as the corresponding state baseline, and network parameters are adapted accordingly.

This algorithm follows a coherent training process, where action execution is carried out synchronously with network training without any corresponding switch between them. The formulation of the reward function adheres to previously described rules for reward functions. The complete sample data is stored in the sample pool using the experience replay method. A subset of samples is then randomly selected from the sample pool for training the neural network.

The state value of the vehicle serves as the baseline, and the cumulative benefit of current vehicle action execution is estimated through both current return and critic network estimation results.

$$G_t = r_t + \gamma V(s_{t+1};w),\ b(s_t) = V(s_t;w) \tag{8}$$

Here, w represents the parameters of the critic network. Subsequently, adjustments are made to both the critic network and actor network in accordance with specific equations.

$$w_{t+1} \leftarrow w_t + \alpha_w \left(G_t - b\left(s_t\right)\right) \nabla V\left(s_t; w\right) \tag{9}$$

$$\theta_{t+1} \leftarrow \theta_t + \alpha_\theta \gamma^t \left(G_t - b\left(s_t\right)\right) \nabla \ln\left(\pi\left(a_t | s_t; \theta\right)\right)) \tag{10}$$

Upon entering the next state s_{t+1}, the vehicle adjusts in accordance with the algorithm. If a termination state is reached during this period, such as a collision or successful completion of the automatic parking task, this necessitates reinitializing the system environment and repeating training until the action strategy output by the network meets the requirements of the automatic parking experiment. The flowchart is illustrated in Fig. 8 and the algorithmic execution is stated briefly below.

(1) Initially, two deep neural networks are employed to generate vehicle action execution strategies. The actor network is responsible for action generation, while the critic network estimates the value of the vehicle state. Subsequently, the actor and critic networks are initialized for the automatic parking task, with the output of the actor network denoted as $\pi(a|s;\theta)$ and that of the critic network denoted as $V(s;w)$. The learning steps α_θ and α_w for both networks are then specified. Following this, in accordance with the environmental design, the vehicle is positioned at its initial location and its corresponding state is recorded as s_t.

(2) Input data from the current front camera of the vehicle is obtained to describe its current state and is then input into the actor network. The actor network returns the vehicle action decision policy in the current state and determines the action a_t is executed in accordance with this policy. Concurrently, it observes the revenue r_t obtained by the vehicle in this round and the next state s_{t+1}.

The critic network's loss function is formulated as

$$Loss\left(w\right) = \sum[r_t + \gamma V\left(s_{t+1}\right) - V\left(s_t\right)]^2, \tag{11}$$

where r_t denotes the immediate reward obtained by taking an action, γ represents the discount factor for future rewards, and V(*) signifies the value estimation of the specified state by the critic network. The revenue gradient of the actor network is expressed as

$$\nabla E_{\pi(\theta)}\left[G_0\right] \approx E\left[\gamma^t \left(r_t + \gamma V\left(s_{t+1}; w\right) - V\left(s_t; w\right)\right) \nabla \ln\left(\pi\left(a_t | s_t; \theta\right)\right)\right], \tag{12}$$

where $\pi(\theta)$ denotes the action execution strategy generated by the actor network, G_0 stands for the cumulative reward acquired through this strategy, γ serves as a discount factor for future rewards, r_t indicates an immediate reward following action execution, and V(*) refers to the estimated value of a given state output by the critic network. The policy gradient generation follows both the policy gradient theorem and state-based baseline criterion.

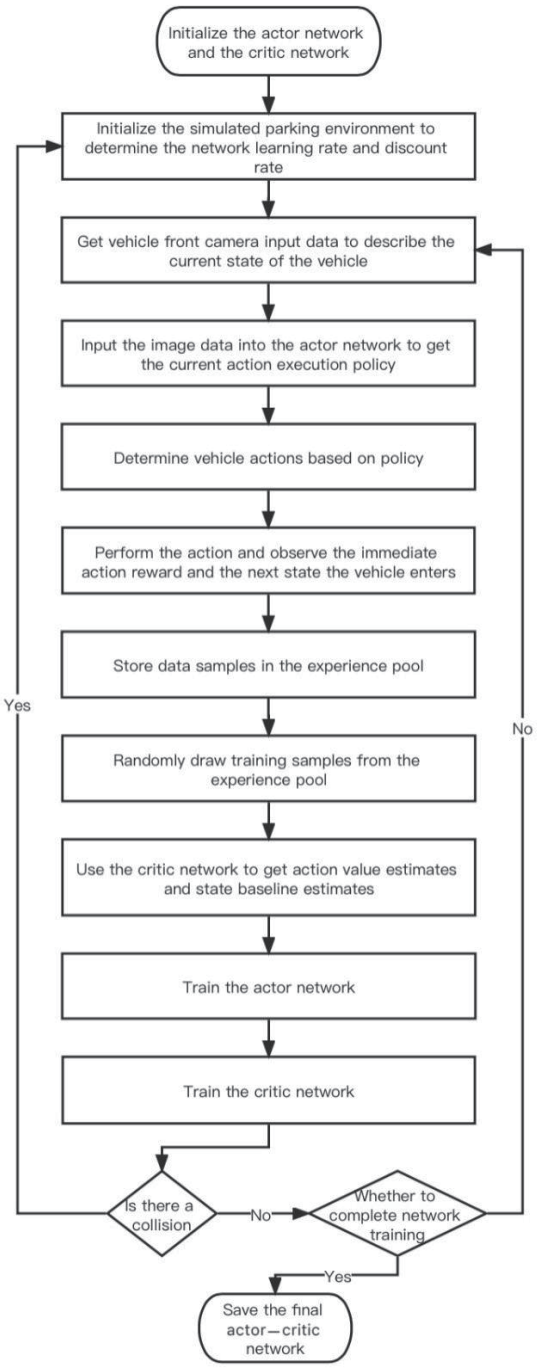


Fig. 8. (Color online) Training flowchart of the actor–critic network.

3. Experiments and Validation of Trained-network Performance

The validation experiments for the trained actor–critic network of automatic vehicle parking are carried out in the CARLA virtual environment. First, the configuration of the experimental environment is addressed.

3.1 Setting experimental environment

In this study, a variety of extraneous vehicles are introduced into the CARLA environment's parking lot,[17] each set to distinct models and colors, serving two primary objectives.

(1) The first aim is to augment environmental visual feature information while mitigating superfluous trial and error during retraining. The original sparsely populated parking lot in CARLA software solely exhibits demarcated parking spaces. Consequently, during automated vehicle parking training, the predominant content of camera input comprises a predominantly monotonous gray asphalt road and a distant sky. This dearth of discernible data features results in protracted network training stages and impedes the attainment of the final target position owing to continued cyclical network training limited to local optimal solutions.
(2) Furthermore, the inclusion of stationary auxiliary vehicles concurrently constrains the operational scope of trained vehicles, curbing unnecessary environmental exploration and expediting vehicle training processes.

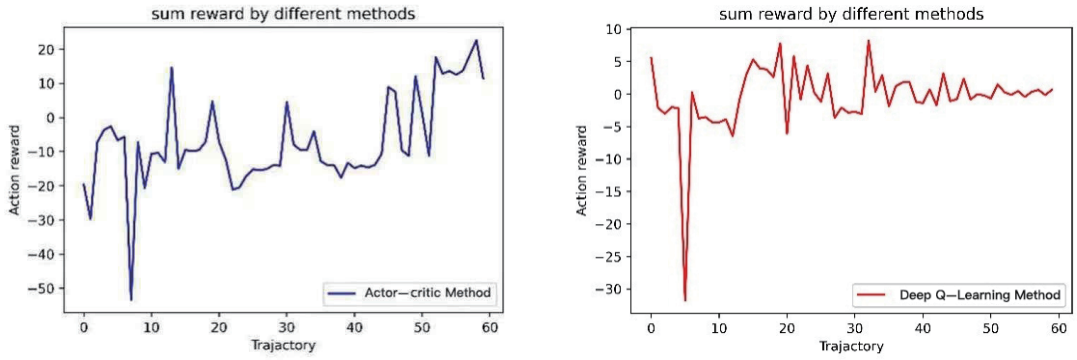
3.2 Training actor-critic network and validation of performance

The training process of the actor–critic network refers to the sequence of actions executed by a vehicle from its initial state until it reaches the termination state. The latter may denote the successful completion of a specific driving task or an early cessation due to vehicular collision.

To show the training process of the actor–critic network, the sum reward $\sum R_{\Phi}^{t}$ of each training trajectory is selected. The sum reward of each training trajectory is defined by the cumulation of the reward function R_{Φ}^{t} in Eq. (4). During the same training, a higher value of the sum reward serves as empirical evidence that the vehicle is in closer proximity to the designated free parking space upon termination. Consequently, the sum reward can serve as an indirect indicator of the training progress for the automatic parking task.

The actor–critic network method is utilized to accomplish the automatic parking task, and in this study, we also employed the deep Q-learning method[20,21] for performance comparison. Both methods train the same untrained random initialization network synchronously, and their training progress is evaluated by comparing the sum rewards obtained in each training sequence. The sum rewards for each training sequence using the actor–critic network method and the deep Q-learning method are illustrated in Figs. 9(a) and 9(b), respectively. It is evident that the actor–critic method exhibits a faster training rate, achieving a larger sequence sum reward with fewer rounds. Conversely, the sum reward growth rate of each training sequence obtained by the deep Q-learning method is significantly smaller than that of the actor–critic method, as depicted in Fig. 9.

Following extensive training, the actor–critic network successfully attained the desired automatic parking goal using the state baseline. The trained algorithm leverages the actor network for vehicle action selection and proficiently executes automatic parking within the CARLA simulation environment. Screenshots from the comprehensive automatic parking



(a) (b)

Fig. 9. (Color online) Sum rewards of the actor–critic and deep Q-learning methods.

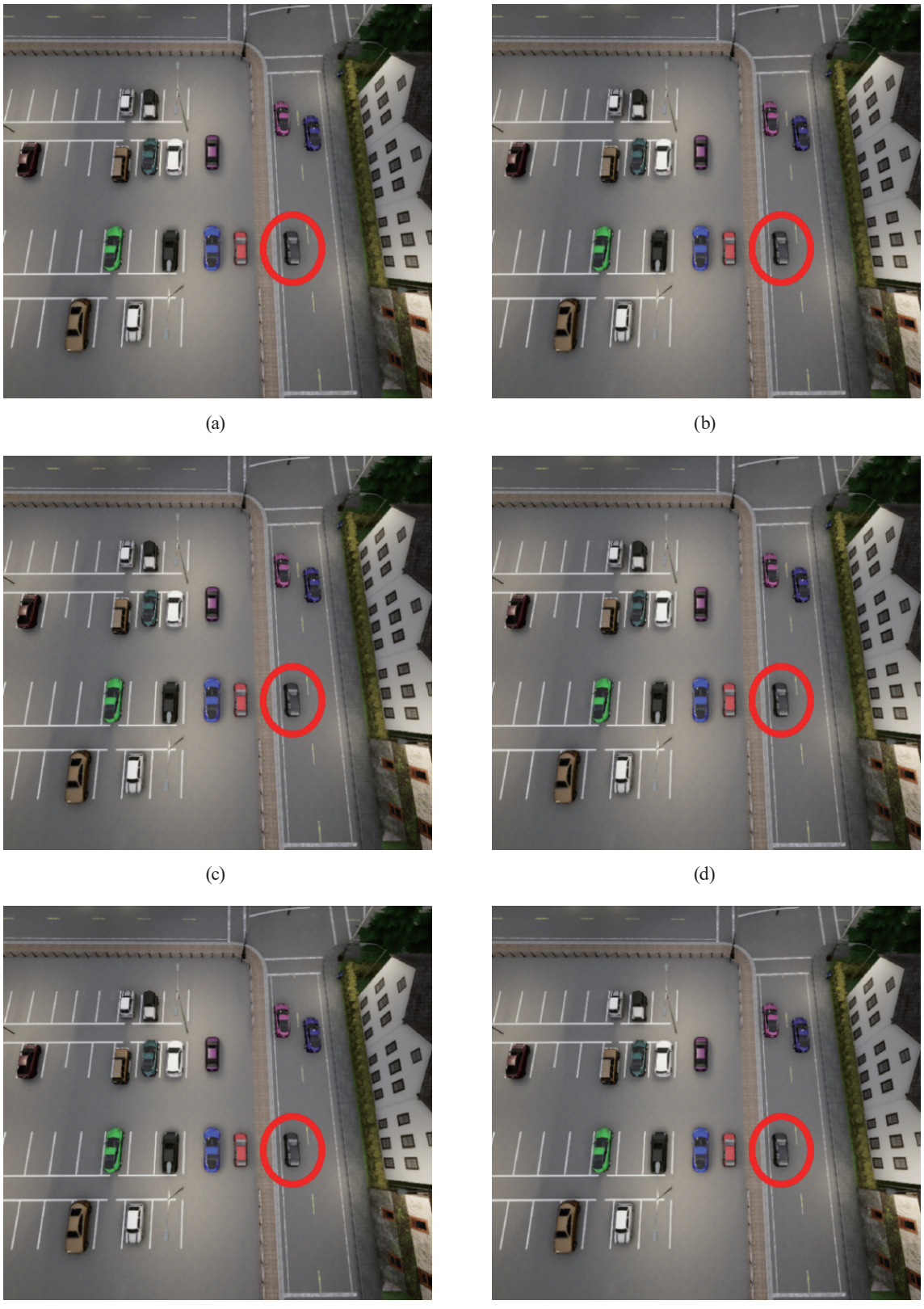
demonstration video are depicted in Fig. 10. The vehicle initiates the parking task on the main road [Fig. 10(a)], proceeds to an adjacent parking lot [Fig. 10(b)], navigates around existing vehicles within the lot [Figs. 10(c) to 10(e)], and ultimately accomplishes automatic parking in a designated space through its own steering control [Fig. 10(f)].

The detailed procedural guidelines for the execution of the automatic parking process are delineated as follows. The vehicle is to be propelled forward throughout the entire process, accompanied by a gradual shift in the steering angle. Specifically, the initial stage requires a steering angle of 45 degrees to the left, while the concluding stage necessitates a further increase to 90 degrees to the left. This detailed sequence is schematically represented in Fig. 11.

3.3 Discussion and Limitation of Proposed Method

Although demonstrating potential, the proposed network method still has room for considerable improvement. First, the design of the action space in its current form adopts a discrete scheme, utilizing only ten distinct steering angles with a fixed throttle opening. This approach, while functionally adequate, does not accurately represent the diverse range of driving habits and situations that drivers typically encounter. To address this limitation, it is recommended that the action space be expanded and fine-tuned to better reflect the continuous nature of real-world driving. The deep deterministic gradient descent reinforcement learning algorithm, which can directly output vehicle motions like steering angle and accelerator in a more realistic and continuous manner, could be used to do this. Unfortunately, owing to time constraints, this solution was not implemented in this study.

Second, the implementation of the automatic parking function, while demonstrating functionality, primarily focuses on proven free parking spaces. This approach neglects the importance of detecting available parking spaces in a more thorough and comprehensive manner. To enhance the applicability of the automatic parking function, future work on the detection of available parking spaces and a variety of criteria, such as driver preferences, vehicle size, and parking space dimensions, should be considered to improve the application of the automatic parking feature.



(a) (b) (c) (d) (e) (f)

Fig. 10. (Color online) Demonstration of automatic vehicle parking process.

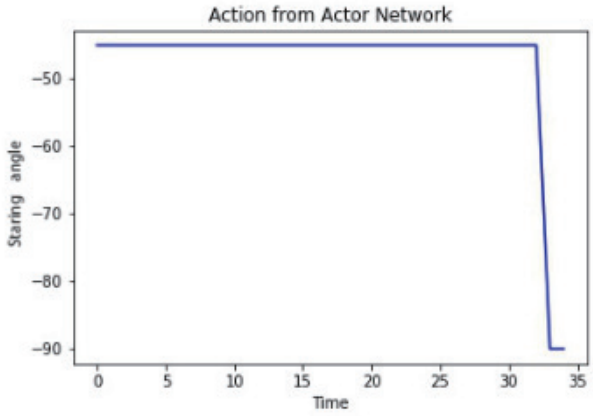


Fig. 11. (Color online) Action execution record diagram of automatic parking demonstration.

Third and last, while the proposed network method shows promise, it is essential to further refine and enhance its design to better accommodate the complex and diverse nature of real-world driving. This will not only improve the overall performance of the system but also increase its applicability and practicality in everyday scenarios.

4. Conclusions

In this study, we presented an automatic vehicle parking algorithm based on the actor–critic framework. The actor–critic network utilized a single front-facing camera as an environmental sensor for the vehicle and processed image data through a deep-reinforcement-learning-based network. It directly formulated the action selection strategy for the current vehicle state and achieved automatic parking in identified spaces by enabling autonomous training of the network. The adjustment process of the actor–critic network's parameters was theoretically analyzed to improve the network convergence rate and further enhance the design of the reinforcement learning environment for automated parking.

Acknowledgments

This work was carried out as part of the Joint Innovation Project of Industry University Research of Fujian Province (Grant No. 2023H6036), Major Science and Technology Projects of Fujian Province (Grant Nos. 2023T5001 and 2022HZ026025), the Program for Innovative Research Team in Science and Technology in Fujian Province University, the Production and Research Collaboration with Innovative in Key Scientific and Technological Project of Sanming City (Grant No. 2022-G-17), and the Operational Funding of the Advanced Talents for Scientific Research (Grant no. 19YG04) of Sanming University. The authors also acknowledge the support from the School of Mechanical and Electric Engineering, Sanming University.

References

1 D. González, J. Pérez, V. Milanés, and F. Nashashibi: IEEE Trans. Intell. Transp. Syst. **17** (2016) 1135. https://doi.org/10.1109/TITS.2015.2498841
2 B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli: IEEE Trans. Intell. Veh. **1** (2016) 33. https://doi.org/10.1109/TIV.2016.2578706
3 A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne: ACM Comput. Surv. **50** (2017) 1. https://doi.org/10.1145/3054912
4 L. L. Mero, D. Yi, M. Dianati, and A. Mouzakitis: IEEE Trans. Intell. Transp. Syst. **23** (2022) 14128. https://doi.org/10.1109/TITS.2022.3144867
5 Z. Guo, Q. Zhang, F. Ding, X. Zhu, and K. Yu: IEEE Trans. Comput. Social Syst. (2023) 5079. https://doi.org/10.1109/TCSS.2023.3298480
6 N. Wang, Y. Gao, H. Zhao, and C.K. Ahn: IEEE Trans. Neural Networks Learn. Syst. **32** (2021) 3034. https://doi.org/10.1109/TNNLS.2020.3009214
7 N. Wang, Y. Gao, and X. Zhang: IEEE Trans. Neural Networks Learn. Syst. **32** (2021) 5456. https://doi.org/10.1109/TNNLS.2021.3056444
8 G. Liu, W. Deng, X. Xie, L. Huang, and H. Tang: IEEE Trans. Cybern. **53** (2023) 7187. https://doi.org/10.1109/TCYB.2022.3198259
9 N. Wang, Y. Gao, C. Yang, and X. Zhang: Neurocomputing **484** (2022) 26. https://doi.org/10.1016/j.neucom.2021.04.133
10 T. Li, Z. Wang, G. Yang, Y. Cui, Y. Chen, and X. Yu: Int. J. Intell. Syst. **37** (2022) 6605. https://doi.org/10.1002/int.22428
11 S. Zhang, Y. Li, and Q. Dong: Appl. Soft Comput. **105** (2022) 108194. https://doi.org/10.1016/j.asoc.2021.108194
12 F. Guo, H. Xu, P. Xu, and Z. Guo: Math. Biosci. Eng **21** (2024) 1058. https://doi.org/10.3934/mbe.2024044
13 Z. Yuan, Z. Wang, X. Li, L. Li, and L. Zhang: Sensors **23** (2023) 4087. https://doi.org/10.3390/s23084087
14 E. Pagot, M. Piccinini, E. Bertolazzi, and F. Biral: IEEE Access **11** (2023) 124163. https://doi.org/10.1109/ACCESS.2023.3330431
15 S. Bhatnagar, R. Sutton, and M. Ghavamzadeh: Autom. **45** (2009) 2471. https://doi.org/10.1016/j.automatica.2009.07.008
16 S. Qiu, Z. Yang, J. Ye, and Z. Wang: IEEE J. Sel. Areas Inf. Theory **2** (2021) 652. https://doi.org/10.1109/JSAIT.2021.3078754
17 CARLA simulator Institution: https://carla.readthedocs.io/en/latest/ (accessed June 2024).
18 R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour: Adv. Neural Inf. Proc. Syst. **12** (2000) 1057.
19 K. Zhang, A. Koppel, H. Zhu, and T. Başar: SIAM J. Control Optim. **58** (2020) 3586. https://doi.org/10.1137/19M1288012
20 Z. Cheng, Q. Zhao, F. Wang, Y. Jiang, L. Xia, and J. Ding: Energy Build. **127** (2016) 43. https://doi.org/10.1016/j.enbuild.2016.05.067
21 Q. Han, S. Feng, X. Wu, J. Qi, and S. Yu: Appl. Sci. **13** (2023) 6030. https://doi.org/10.3390/app13106030