Intelligent Parking Lot System Based on Automatic Parking Hybrid Controller

Junyi Li*, Yixiao Wang†, Songxin Lei*, Xinyao Guo*, Lu Zheng*, Yu Zhang*, and Huaming Wu†
*School of Mathematics, Tianjin University, China
†Center for Applied Mathematics, Tianjin University, China
E-mail: {junyi_li, wang_yixiao, whming}@tju.edu.cn

***Abstract*—Recently, the automobile industry has experienced rapid growth, driven by increasing population and living standards, resulting in a surge of vehicles in cities. Many large parking lots lack intelligent parking space management systems, leading to low efficiency and confusion for users. Existing intelligent parking systems primarily rely on automatic timing and charging, closed-circuit television, and video recorder systems, which are insufficient to handle complex terrain and high traffic flow. This often leads to traffic congestion and unreasonable parking space occupation. Additionally, parking lots in China face issues such as difficulty and high costs associated with parking, primarily due to the imperfect utilization and allocation of resources. To address these issues, we propose a smart parking system that maximizes the efficiency of existing parking spaces instead of solely focusing on increasing the size and number of parking lots.**

***Index Terms*—AC-PPO algorithm, MPC algorithm, digital twins, unmanned driving, IoT, automobile industry**

I. Introduction

With the improvement of people's living standards and the rapid increase in population, the number of automobiles has skyrocketed in cities, leading to significant growth in the automobile industry. While this is a positive reflection of our country's rapid economic development, it also presents considerable challenges to the city's parking system. It is crucial to address the parking situation and improve parking facilities urgently.

Currently, many large parking lots in China still lack comprehensive intelligent parking space management systems. Some ordinary parking lots only provide parking spaces without proper management and charging systems, while others are paid parking lots equipped with access gates, parking managers, and cashiers. However, for users, quickly finding an available parking space in a large parking lot remains a relatively challenging task. This results in low overall operational efficiency and management confusion due to parking inconveniences.

Existing intelligent parking systems primarily rely on automatic timing and charging systems, closed-circuit television, and video recorder systems to monitor the parking lots. While these systems have significantly improved the parking situation compared to ordinary parking lots, parking lots with complex terrain and high traffic flow still experience issues such as traffic congestion, long waiting times for parking, difficulty in finding suitable parking spaces, and unreasonable occupation of parking spaces. Additionally, due to the imperfect utilization and allocation of resources, parking in China often faces problems of being difficult and expensive.

The evidence suggests that while increasing the supply of parking spaces is important, it is more feasible to maximize the utilization efficiency of existing parking spaces. Therefore, it is essential to consider not only expanding the size and number of parking lots but also optimizing the use of current parking space resources.

In the field of automatic parking, there have been many research results on unmanned driving and path planning, and various technical means and analysis methods have been adopted. Wang *et al*. [3] utilized the Levenberg-Marquardt algorithm for image mosaicking to generate omnidirectional bird's eye view images, extracted parking space features using Radon transform, and achieved autonomous parking through double circular trajectory planning and preview control strategy. However, this method performs poorly in dimly lit indoor parking lots. Ávalos *et al*. [14] complemented this objective by proposing a novel approach that combines deep learning and sensor fusion techniques to accurately detect and track parking spaces in real-time, enhancing the capabilities of the computer architecture for the distributed system and further improving the efficiency of guiding drivers to available parking spots. However, this study has been criticized for overestimating the complexity of the parking lot environment, leading to more time costs. The algorithm mentioned above partially represents the limitations of traditional path control strategies and neural network methods in parking lot environments. In an attempt to address this, we aim to combine these two types of algorithms to construct a novel control model.

This paper presents a hybrid controller designed to automate the search for parking vacancies and perform parking operations. The hybrid controller utilizes model predictive control (MPC) to navigate along the reference path within the parking lot when only one car is present. Additionally, a deep reinforcement learning (DRL) agent, trained specifically for parking operations, is employed to handle the task of parking once a vacancy is identified. In scenarios where multiple cars exist within the parking lot, the hybrid controller integrates the AC-PPO algorithm along with a global environment model to achieve an optimal solution at a global level. The key contributions of this research can be summarized as follows:

- A method of fully automated parking lot scheduling based

2023 China Automation Congress (CAC) | 979-8-3503-0375-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CAC59555.2023.10451102

on DRL and combined with an MPC controller is proposed to deal with complex scenarios in fully automated parking lots and achieve intelligent automation.
- By adopting centralized training and distributed execution policies, the algorithm's computational complexity is reduced, making it suitable for problems where multiple agents cooperate in large-scale fully automated parking lots.
- Techniques such as discretization processing and importance sampling are used to improve the efficiency and accuracy of the algorithm, which has a wide range of application prospects and practical significance in achieving intelligent automation, improving parking efficiency, and optimizing vehicle flow in fully automated parking lots.

II. Establishment of Parking Environment

As shown in Fig. 1, we first establish the environment model of the fully automatic parking lot. The functional areas of parking lots include exits, entrances, parking spaces, lanes and specific functional areas. Rasterize the area, where each exit and entrance occupies a grid and each parking space occupies a grid. Lane is divided into different grids according to the size of parking spaces. The white grid indicates the feasible area and the black grid indicates the obstacles. For the simplicity of path planning calculation, the grids are numbered from the upper left corner of the map.

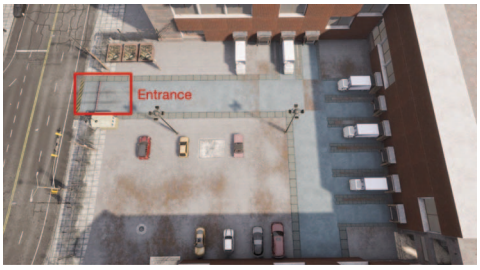


Fig. 1. Typical environment

To ensure the stability of the system and avoid collision between agents, we put forward the following assumptions:

- Each grid can only pass through or accommodate one agent at the same time
- Only one-way driving is allowed in the lane to avoid collision
- The agent moves at a fixed uniform speed V to avoid rear-end collision
- Set the priority of agents, and the agents with higher numbers have higher priority to avoid cross-collision

III. Hybrid Controller Based on MPC and AC-PPO

A. MPC Control Theory

The symbols that may appear in this section are shown in Table I.

1) Parking environment and vehicle kinematics model creation: The parking process is a low-speed running scene, and in the low-speed running scene, the bicycle kinematics model can meet the requirements. Therefore, we mainly realize the

TABLE I
Definitions of Symbols

Symbol	Definition
ψ	The heading information of the vehicle
u	Control deviation of the vehicle
δ_t	The heading angle of the vehicle at time t
$\delta_f(\delta_f ref)$	Front wheel angle input(expected input)
$\delta_r(\delta_r ref)$	Rear wheel angle input(expected input)
L	The wheelbase of the vehicle
v	The speed of the vehicle
$e\psi$	The angle between the vehicle and the road centerline
cte	Deviation between the vehicle and the reference path
cte_{ref}	The reference values of lateral offset
$e\psi_{ref}$	The reference values of angle
$eL(eL_{ref})$	Lateral offset error and its reference value
$eSp(eSp_{ref})$	Speed error and its reference value
$eAng(eAng_{ref})$	Angle offset and its reference value
J	The loss function
A	State transition matrix
$B(W)$	Control input matrices

MPC algorithm based on the bicycle kinematics model. The model is as follows:

$$\begin{cases} x' = v * \cos\psi, \\ y' = v * \sin\psi, \\ \psi' = v * \tan(\delta_f)/L, \end{cases} \tag{1}$$

where (x', y') respectively represents the lateral and longitudinal position information of the vehicle; ψ represents the heading information of the vehicle; δ_f is the corner of the vehicle; L represents the wheelbase of the vehicle; v represents the speed. According to the control model of the MPC algorithm under the framework of Autoware:

$$x' = f(x) = \begin{cases} y' = v * \sin\psi, \\ \psi' = v * \tan(\delta_f)/L, \end{cases} \tag{2}$$

For the linearization process of the above system, refer to the article -Linearization of Nonlinear Systems with Extended Kalman Filter (EKF), and get the above state transition matrix A, control input matrices B and W. And the final linearization model can be obtained by taking the derivative near the equilibrium point $(0, 0)$:

$$x' = A * x + B * u + W * \delta_f ref. \tag{3}$$

There are many forms of discretization of the control system, and we use Bilinear transformation and the Euler method to discretize the system.

By Bilinear transformation and Euler change we can finally get the state space model needed by MPC:

$$x(k+1) = A * x(k) + B * u(k) + W * \delta_f ref, \tag{4}$$

where the expected front wheel angle input $\delta_f ref$ can be inferred from Ackerman's rotation angle theorem, and the control model required by the MPC algorithm is completed.

The following describes how the MPC control algorithm is applied to the specific environment: when the discrete state equation of the system is obtained, the state equation can be

used to predict the state of multi-gait and the discretized state equation can be rewritten as:

$$x(k+1) = A_k * x(k) + B_k * u(k) + W_k * \delta_f ref. \quad (5)$$

2) Automatic parking model creation: Here, it is assumed that the deviation angle of the vehicle's heart side remains unchanged during steering, that is, the instantaneous steering radius of the vehicle is the same as the radius of curvature of the road.

At the rear wheel axis (X_r, Y_r), the speed is:

$$v_r = X_r \cos\psi + Y_r \sin\psi. \quad (6)$$

The kinematic constraints of the front and rear wheels are:

$$\begin{cases} X_f \sin(\psi + \delta_f) - Y_r \cos(\psi + \delta_r) = 0, \\ X_r \sin\psi - Y_r \cos\psi = 0. \end{cases} \quad (7)$$

The following results can be obtained by simultaneous expression:

$$\begin{cases} X_r = v_r \cos\psi, \\ Y_r = v_r \sin\psi. \end{cases} \quad (8)$$

According to the geometric relationship between the front and rear wheels:

$$\begin{cases} X_f = X_r + \cos\psi, \\ Y_f = Y_r + \sin\psi. \end{cases} \quad (9)$$

Then the yaw rate can be obtained as: $\omega = v_r \tan\delta_f$. Based on yaw rate and vehicle speed, the steering radius and front wheel deflection angle can be obtained:

$$\begin{cases} R = v_r/\omega, \\ \delta_f = \arctan(1/R). \end{cases} \quad (10)$$

The obtained vehicle kinematics model is as follows:

$$\begin{bmatrix} X_r \\ Y_r \\ \psi \end{bmatrix} = \begin{bmatrix} \cos\psi \\ \sin\psi \\ (\tan\delta_f)/L \end{bmatrix} * v_r. \quad (11)$$

The vehicle state is denoted as (x_t, y_t, ψ_t, v_t), where automobile world coordinates x_t, automobile world coordinates y_t and driving speed v_t. The control input of the vehicle is (δ_t, a). Two state variables, $e\psi_t$ and cte_t, are added to the complete model to describe the included angle between the vehicle and the road centerline and the lateral deviation between the vehicle and the reference trajectory. The reference trajectory is based on the polynomial fitting the curve expression $f(x_t)$, and the lateral offset cte_t is obtained in real time based on $f(x_t)$.The complete model is as follows:

$$x_{t+1} = x_t + v_t * \cos\psi_t * dt, \quad (12)$$

$$y_{t+1} = y_t + v_t * \sin\psi_t * dt, \quad (13)$$

$$\psi_{t+1} = \psi_t + v_t/L_f * \delta_t * dt, \quad (14)$$

$$v_{t+1} = v_t + a * dt, \quad (15)$$

$$cte_{t+1} = f(x_t) - y_t + v_t * \sin(e\psi_t) * dt, \quad (16)$$

$$e\psi_{t+1} = \psi_{t+1} + v_t/L_f * \delta_t * dt. \quad (17)$$

Rolling optimization is to obtain the optimal control solution, and based on constraints, make one or some performance indicators achieve the optimal control function. Then designing an appropriate optimization objective function is the key to the superiority of the result. The objective function is expressed as a quadratic function of state and control input: s

$$J = \sum_{t=1}^{N} \Big[(cte_t - cte_{ref})^2 + (e\psi_t - e\psi_{ref})^2 + (eL_t - eL_{ref})^2 + (eSp_t - eSp_{ref})^2 + (eAng_t - eAng_{ref})^2 \Big], \quad (18)$$

where J is the loss function, N is the prediction time domain, cte_{ref} and $e\psi_{ref}$ are the reference values of lateral offset and angle, respectively. The square is to unify the symbols, and the weight difference should not be too wide. There are dimensional differences between the data itself. Since there is no normalization, special attention should be paid to the trade-off between weights. The parameters of the loss function are arranged according to priority in Table II.

TABLE II
ORDER OF THE PARAMETERS

Order of the parameters
A Lateral offset error eL
B Speed error eSp
C Angle offset $eAng$

After building the loss function, you also need to set constraints: In the project, the control output is (δ_t, a), that is, the steering and acceleration of the vehicle. We have restrictions here

$$\begin{cases} -1 \le a \le 1, \\ -15 \le \delta_t \le 15. \end{cases} \quad (19)$$

B. AC-PPO Algorithm

The symbols that may appear in this section are shown in Table III.

TABLE III
DEFINITIONS OF SYMBOLS

Symbol	Definition
X_t	The state of the system at time t
x_i^t	The state of the agent i at time t
a_i	The action of the agent i
d_i	The target point position of the agent i
v_i^t	The linear velocity of agent i at time t
w_i^t	The angular velocity of agent i at time t
N	The prediction time domain
D^t	Distance between the agent and target point at time t
A_k	The advantage estimates
r_i	The reward function
θ_i^Q	Parameters of Critic Network for each agent i
θ_i^μ	Parameters of Old-Actor network for each agent i
$\theta_i^{\mu'}$	Parameters of New-Actor network for each agent i

1) Establishment of agent:

- **State**: The state of the system at time t is defined as:

$$X_t = [x_1^t, x_2^t, \ldots, x_N^t], \tag{20}$$

where x_i^t is the state of agent i, and N is the number of agents allowed to be active at the same time, that is, the sum of the total number of parking and picking up cars. For agent i, the state of time t is defined as:

$$x_t = [s_i^t, d_i, v_i^t, w_i^t], \tag{21}$$

where s_i^t denotes the position of agent i at time t, representing the numerical value on the grid map. d_i is the target point position of the agent i, specifically referring to the number assigned to the designated parking space on the grid map. To ensure consistency, the linear velocity of agent i at time t is denoted as v_i^t, normalized within the interval $[0, 1]$. Additionally, we employ w_i^t to represent the angular velocity of agent i at time t, which is normalized to the range of $[-1, 1]$.
- **Action**: ITo facilitate the agent's movement towards the target point, the allowed actions are moving forward, turning left, turning right and standing still. Therefore, the agent's action is defined as a control instruction encompassing both linear velocity and angular velocity, denoted as $a_i = (v_i^t, w_i^t)$.
- **Reward function (reward)**: Agent i takes the reward function of behavior a_t at time t and state X_t.

$$r_i(s_t, v_t) = \begin{cases} r_a, \text{ if } D_i^t < D_{arrive}, \\ r_b, \text{ if } C_i^t < D_{collsion}, \\ \sum_{i=1}^{N} d(D_i^{t-1} - D_i^t), \text{ otherwise}, \end{cases} \tag{22}$$

where D_i^t signifies the distance between agent i and its target point at time t. If this distance falls below the threshold value D_{arrive}, it is considered as reaching the target point, leading to the reward represented by the first item. If the distance between the agent and an obstacle is less than the safety threshold $D_{collision}$, it is deemed as an impending collision, resulting in a punishment represented by the second item. The third item is to guide the agent towards its target point. Each agent i measures the distance D_i^{t-1} from his target point at time $t-1$ and the distance D_i^t from his target point at time t. If it is farther away from its target point at time t, then it will be punished, otherwise, it will be rewarded. d is the reward parameter. The distance function D_i^t adopts Manhattan distance, which can be formulated as:

$$D_i^t = dist(s_i^t, d_i) = |x_i - x_j| + |y_i - y_j|, \tag{23}$$

where s_i is the number of the agent i on the grid map at time t, and d_i is the number of the target point on the grid map. Specifically, (x_i, y_i) denotes the row and column coordinates corresponding to the position of agent i represented by s_i^t, while (x_j, y_j) denotes the row and column coordinates corresponding to the target point represented by d_i.

Algorithm 1: PPO Algorithm

Initialization: $\theta^Q, \theta^{\mu'} = \theta^\mu, r_t$
for $k = 0, 1, 2, \ldots$ **do**
  while *Experience pool is Full* **do**
    Input environment information into μ';
    Construct a normal distribution;
    Sample a from the normal distribution;
    Input a into the environment and obtain r;
    Store $S[X, a, r]$;
  According to [16] compute the advantage estimates A_k;
  $Loss = mean(square(A_k))$;
  use backpropagation to update Q;
  for $Step = 1$ **to** $Step = Update\_Steps$ **do**
    Obtain the normal distributions N_1, N_2;
    get $prob_1$ and $prob_2$ from N_1 and N_2;
    $ratio = prob_2/prob_1$;
    According to [16], for $\epsilon \in [0, 1]$
    $a_{loss} = mean(min((ration * A_k, clip(ratio, 1-\epsilon, 1+\epsilon)^*$;
    use backpropagation to update μ';
  use backpropagation to update μ;

2) Establishment of network model: To implement the proposed approach, three neural networks are constructed for each agent: Critic Network (Q), New Actor Network (μ'), and Old Actor Network (μ). The network parameters of each agent i are denoted as θ_i^Q, $\theta_i^{\mu'}$, and θ_i^μ, respectively. Initially, the parameters of the New Actor Network are set to be the same as the Old Actor Network, i.e., $\theta_i^{\mu'} = \theta_i^\mu$. The state space X^0 and an experience replay buffer are initialized. The experience pool serves as a repository for storing training samples in the format of $[x_t, A_t, r_t, X_t']$. Here, $x_t = [x_1^t, x_2^t, \ldots, x_n^t]$ represents the current state, encompassing the observation values of N agents. Similarly, $A_t = [a_1^t, a_2^t, \ldots, a_n^t]$ denotes the behavior executed by the N agents. Furthermore, $r_t = [r_1^t, r_2^t, \ldots, r_N^t]$ corresponds to the respective return values associated with the state and action pairs.

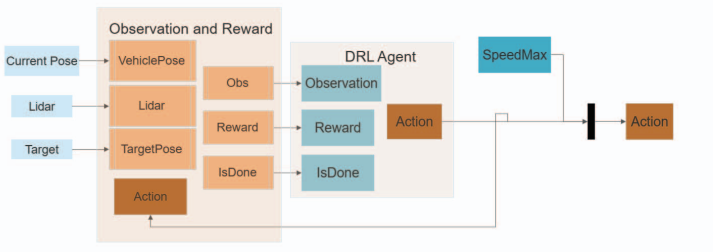


Fig. 2. The structure of DRL Agent

The updating process is presented in **Algorithm 1**, which outlines the sequence of steps involved in the training procedure. Moreover, to provide a visual representation of the

AC-PPO agent in relation to the specified settings, we depict its structure in Fig. 2.

C. Training Process

Algorithm 2 outlines the training process, which consists of two fully connected layers, each with 128 neurons. Both layers utilize the Tah activation function to stabilize the network model during reward function calculation. The second fully connected layer employs the ReLU activation function to promote sparse activation of the neurons.

Algorithm 2: Training process

Initialize state $S[X, a, r]$;
Reward in each round $ep_r = 0$;
while *need to complete an iteration* **do**
    determine the action that the car should take;
    increase motion noise, and restrain a, r;
    get the next state $S'[X', a', r']$ and reward R;
    record the data and store S';
if *the experience replay pool is full* **then**
    gradually reduce noise;
    use PPO algorithm to learn;
else
    $S = S', ep_r += R$, then stop;

IV. Experiments

A. Model Training

In the training process, exceptional parking actions and states from the experience replay pool are sampled with a step size of 20 rounds. The reward value converges over 10,000 rounds throughout the entire training process of the PPO algorithm. Fig. 3 illustrates the progressive convergence of the car parking reward value from -30 to near -20, indicating successful algorithm convergence.

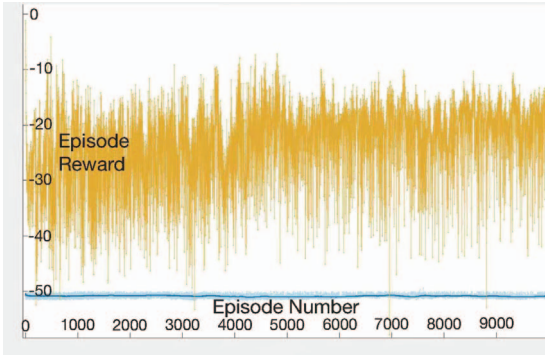


Fig. 3. DRL model training

B. Experiment and Analysis

Combining the MPC Control algorithm and AC-PPO agent we built, we achieve the final version of the agent, which is shown in Fig. 4. Based on the model, we try to show that our hybrid controller is effective by doing experiments.

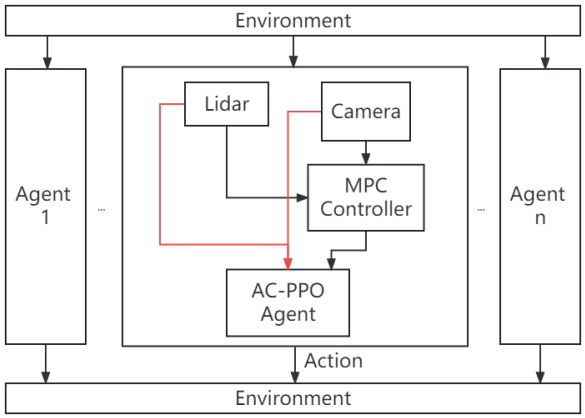


Fig. 4. Structure of the Hybrid controller

1) General Experiment: Modify the starting position of the car and conduct a set of generalization experiments. Implement the automatic parking model established in this paper. Among the many experimental results, we selected two categories of the most representative experiments for demonstration, one is to test the controller's suitability for **vertical parking spaces**, and the other one is to test the controller's suitability for **horizontal parking spaces**. The target pose of the cars in each environment is shown in Fig. 5.

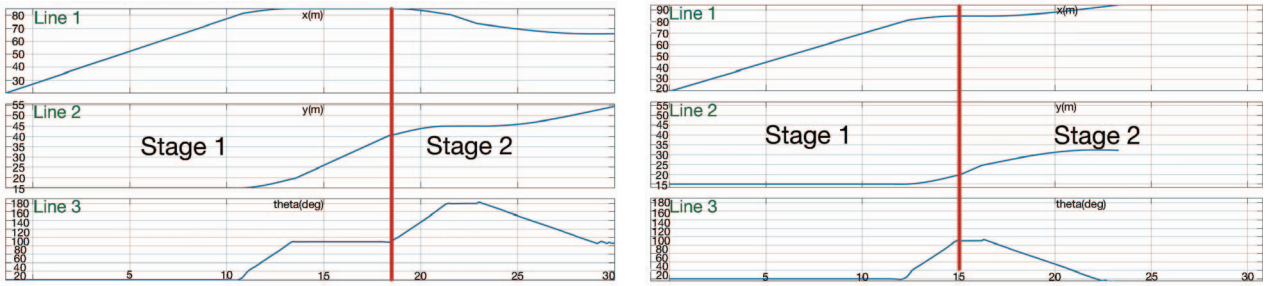
(a) Environment 1 (b) Environment 2

Fig. 5. General experimental environments

The result of the experiment is shown in Fig. 6. The change of car's location (x, y) [Line 1 and Line 2] and ψ [Line 3] during the parking process is divided into two stages. **Stage 1** echos to the MPC controller and **Stage 2** echos to AC-PPO Agent. A car equipped with autonomous driving functions drove into a parking lot, The network of the parking lot took control of the autonomous car and operated the onboard camera and radar. The autonomous car first cruised along the purple reference path guided by the MPC controller, as shown in **Stage 1** of the experiment. When the onboard camera detected an empty parking space, the vehicle began to call the AC-PPO module for operation and the car starts parking in the green area as shown in the figure, controlled by the AC-PPO controller, corresponding to the results of **Stage 2** in the experiment, until the car is parked in the blue target position.

2) Comparative Experiment: For the two mentioned environments, we conducted comparative experiments using multiple algorithms, and the results are presented in Table IV.

From the experimental results, we can observe that our **Hybrid controller** retains the advantages of the MPC algorithm



(a) Experiment 1 (b) Experiment 2

Fig. 6. Results of general experiments

in the first stage and demonstrates fast convergence similar to the DRL algorithm in the second stage.

The total time of the parking process guided by our **Hybrid controller** ranked among the top in both environments. However, we note that in Experiment 1, the Rule-based control strategy showed better performance, mainly due to the longer cruising phase in that specific environment, highlighting the strengths of the Rule-based control strategy. Nonetheless, our model's performance is superior to traditional machine learning algorithms.

TABLE IV
COMPARATIVE EXPERIMENT

Experiment 1	Stage 1(s)	Stage 2(s)	Total(s)
Rule-based control strategy	25.7	7.8	33.5
Optimization-based method(A^*)	26.0	8.8	34.8
Pure Model Predictive Control	26.4	7.9	34.3
DDPG Algorithm	27.1	7.5	34.6
Hybrid controller	**26.3**	**7.6**	**33.9**
Experiment 2	Stage 1(s)	Stage 2(s)	Total(s)
Rule-based control strategy	16.8	8.3	25.1
Optimization-based method(A^*)	17.3	8.4	25.7
Pure Model Predictive Control	17.5	8.0	25.5
DDPG Algorithm	19.0	8.2	27.2
Hybrid controller	**17.1**	**7.8**	**24.9**

V. CONCLUSION

We found that combining the MPC algorithm with DRL retains the simplicity and responsiveness of MPC in fixed-path scenarios, as well as the fast and accurate guidance of DRL in the parking process, reducing the parking time of autonomous vehicles and improving their efficiency. Experimental results in a simple parking lot environment demonstrate the feasibility of our algorithm. Comparative experiments with other algorithms have demonstrated that our algorithm can improve operational efficiency, which means our architecture has the potential for practical application in parking environments.

However, it should be noted that our assumptions did not fully consider how the onboard cameras and radar detect the environment and convert environmental features into data, as well as the braking response in unexpected situations, such as the sudden appearance of pedestrians. These aspects can be further improved in future work.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No. 62071327), Tianjin Science and Technology Planning Project (No. 22ZYYYJC00020). Huaming Wu is the corresponding author.

REFERENCES

[1] Sun Yinjian, "Research on trajectory tracking control algorithm of unmanned vehicle based on model predictive control," 2015.
[2] Peng Xiaoyan, "Research on local path planning algorithm for unmanned vehicles," Automotive Engineering, 2020.
[3] Wang C, Zhang H, Yang M, et al. Automatic parking based on a bird's eye view vision system[J]. Advances in Mechanical Engineering, 2014, 6: 847406.
[4] AdamShan, "Introduction to self-driving car systems (Part 10) - Model predictive control based on kinematic model," https://blog.csdn.net/AdamShan/article/details/112868531, 2021.
[5] Liu Qing, Liu Bin, Wang Guan, Zhang Chen, Liang Zhixing, and Zhang Peng, "Research on digital twin model, problems and progress," Journal of Hebei University of Science & Technology, vol. 40, no. 1, 2019.
[6] Chen J, Xue Z, and Fan D, "Deep reinforcement learning based left-turn connected and automated vehicle control at signalized intersection in vehicle-to-infrastructure environment," Information, 2020.
[7] Engstrom L, Ilyas A, Santurkar S, Tsipras D, Janoos F, Rudolph L, and Madry A, "Implementation matters in deep policy gradients: A case study on PPO and TRPO," arXiv preprint arXiv:2005.12729, 2020.
[8] Han Shi-Yuan and Liang Tong, "Reinforcement-learning-based vibration control for a vehicle semi-active suspension system via the PPO approach," Applied Sciences, vol. 12, no. 6, p. 3078, 2022.
[9] Li Dianzhao and Okhrin Ostap, "Modified DDPG car-following model with a real-world human driving experience with CARLA simulator," Transportation Research Part C: Emerging Technologies, vol. 147, p. 103987, 2023.
[10] Mahmud SA, Khan GM, Rahman M, Zafar H, et al., "A survey of intelligent car parking system," Journal of Applied Research and Technology, vol. 11, no. 5, pp. 714–726, 2013.
[11] Thomas Diya and Kovoor Binsu C, "A genetic algorithm approach to autonomous smart vehicle parking system," Procedia Computer Science, vol. 125, pp. 68–76, 2018.
[12] Lv Liangheng, Zhang Sunjie, Ding Derui, and Wang Yongxiong, "Path planning via an improved DQN-based learning policy," IEEE Access, vol. 7, pp. 67319–67330, 2019.
[13] Kober Jens, Bagnell J Andrew, and Peters Jan, "Reinforcement learning in robotics: A survey," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1238–1274, 2013.
[14] Ávalos H, Gómez E, Guzmán D, et al. ¿ Where to park? Architecture and implementation of an empty parking lot, automatic recognition system[J]. Enfoque UTE, 2019, 10(1): 54-64.
[15] Li Junzuo, Long Qiang. "An Automatic Parking Model Based on Deep Reinforcement Learning", Journal of Physics: Conference Series, 2021.
[16] Schulman et al, "Proximal Policy Optimization Algorithms", 2017.