HOPE: A Reinforcement Learning-Based Hybrid Policy Path Planner for Diverse Parking Scenarios

Mingyang Jiang, Yueyuan Li, Songan Zhang, Siyuan Chen, Chunxiang Wang, and Ming Yang, *Member, IEEE*

***Abstract*— Automated parking stands as a highly anticipated application of autonomous driving technology. However, existing path planning methodologies fall short of addressing this need due to their incapability to handle the diverse and complex parking scenarios in reality. While non-learning methods provide reliable planning results, they are vulnerable to intricate occasions, whereas learning-based ones are good at exploration but unstable in converging to feasible solutions. To leverage the strengths of both approaches, we introduce Hybrid pOlicy Path plannEr (HOPE). This novel solution integrates a reinforcement learning agent with Reeds-Shepp curves, enabling effective planning across diverse scenarios. HOPE guides the exploration of the reinforcement learning agent by applying an action mask mechanism and employs a transformer to integrate the perceived environmental information with the mask. To facilitate the training and evaluation of the proposed planner, we propose a criterion for categorizing the difficulty level of parking scenarios based on space and obstacle distribution. Experimental results demonstrate that our approach outperforms typical rule-based algorithms and traditional reinforcement learning methods, showing higher planning success rates and generalization across various scenarios. We also conduct real-world experiments to verify the practicability of HOPE. The code for our solution is openly available on https://github.com/jiamiya/HOPE.**

***Index Terms*— Automated parking, reinforcement learning, path planning.**

I. Introduction

AUTOMATED parking is a tempting technology to improve driving safety and efficiency [1]. An automated parking system comprises several vital components, including perception, planning, and control, in which path-planning algorithms play a crucial role [2]. In parking scenarios, the path-planning task involves generating a feasible path from the start position to the target parking spots under specific physics constraints. Compared to other scenarios, finding a feasible path in parking scenarios is usually more challenging because of the lower error tolerance of the target spot and the lack of navigational reference lines [3]. Additionally, the limited space and surrounding obstacles shrink the number of potential solutions. While existing path-planning approaches have proved practical in most simple scenarios, their inherent difficulty in understanding the surrounding environment may lead to planning failures, especially when scenarios become more intricate [4].

Learning-based planners have the potential to understand environments and intelligently plan routes through a data-driven approach, diverging from reliance on pre-defined planning methods rooted in human priors. While expert data is used as the ground truth for imitation learning in various tasks, the scarcity of large-scale datasets for parking scenarios necessitates researchers to collect data manually [5]. This supervised learning approach risks overfitting the model to specific parking strategies due to the insufficient diversity in the training scenarios. Meanwhile, Reinforcement Learning (RL) has gathered increasing attention in the field of autonomous driving [6]. Indeed, through interaction with the environment, RL methods enable the training of agents without labeling trajectory ground truth. Nonetheless, training RL agents in complex and diverse scenarios remains a challenging task [7]. It is straightforward for the agent to fit into fixed parking strategies but poses greater difficulty in obtaining generalization capability across various parking scenarios. Moreover, the agent faces challenges in exploring effectively, especially in complex scenarios with narrow parking spaces, significantly impacting the training efficiency.

This paper focuses on employing RL methods in the parking path planning task with static obstacles. To achieve efficient and effective learning under diverse parking scenarios, we propose reinforcement learning-based **H**ybrid p**O**licy **P**ath plann**E**r (HOPE). The hybrid policy planner is designed to leverage RL-based methods and a classical geometric-based path planning method, the Reeds-Shepp (RS) curve [8]. A transformer-based structure is used as the information fusion network in actor and critic networks [9]. Since the diversity of scenario difficulty has a significant impact on training and testing, we rank the difficulty of scenarios by referencing related standards for automated parking. Overall, the main contributions of this paper include:

- We develop a hybrid policy method for parking path planning which achieves over 97% success rate across diverse and challenging parking scenarios and verifies its generalization ability with real-world experiments.

Received 25 January 2024; revised 4 July 2024 and 2 November 2024; accepted 26 February 2025. Date of publication 26 March 2025; date of current version 5 May 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62173228, Grant 62373250, Grant U22A20100, and Grant 52402504. The Associate Editor for this article was B. Fidan. *(Mingyang Jiang and Yueyuan Li are co-first authors.) (Corresponding author: Ming Yang.)*

Mingyang Jiang, Yueyuan Li, Siyuan Chen, Chunxiang Wang, and Ming Yang are with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai 200240, China (e-mail: MingYANG@sjtu.edu.cn).

Songan Zhang is with the Global Institute of Future Technology, Shanghai Jiao Tong University, Shanghai 200240, China.

Digital Object Identifier 10.1109/TITS.2025.3550417

- We propose to implement an action mask in the path planning task. This mechanism excludes improper actions for reinforcement learning agents and significantly improves training efficiency and performance.
- We propose a criterion to rank the difficulty of static parking scenarios. Comprehensive experiments are conducted across simulations of different difficulty levels, demonstrating a notable improvement in success rates compared to rule-based and naive reinforcement learning-based methods.

II. Related Work

A. Non-Learning-Based Parking Path-Planning

Non-learning-based path planning methods under parking scenarios mainly include geometric-based methods and sampling-and-search-based methods [3]. Geometric planners construct paths connecting the start position and the target position using different kinds of geometric curves [10], [11]. Typical geometric methods include the Dubins curves and the RS curves [8], [12]. Building upon the RS curve, several improvements have been made to enhance its performance in parking scenarios [13], [14]. The sampling-and-search-based approach discretizes and searches either in the state space or in the control space to find a valid path. One widely employed approach is the Hybrid A* method [15], which devises a variant of the A* algorithm that incorporates the kinematic state space of the vehicle to derive a kinematically feasible trajectory. Hybrid A* was initially utilized in the Defense Advanced Research Projects Agency Challenge (DARPA) and has since undergone improvements and application to path planning tasks in parking scenarios [16], [17]. Both geometric and sampling-and-search-based methods, as rule-based methods, leverage human prior knowledge to design algorithms. In most common cases, these priors can serve as a strong fallback to obtain satisfactory solutions. However, these methods can hardly achieve human-level proficiency in complex and diverse parking scenarios [18].

Another category of methods involving path planning is the optimization-based trajectory planning approach. These methods solve the trajectory planning problem by formulating it as an optimal control problem. Related works have made improvements in collision avoidance constraint formulation [19], iterative solution efficiency [3], and robustness [20]. While this approach can produce smoother paths adhering to vehicle kinematics, the optimization process relies on another path-planning algorithm, often the Hybrid A* method [19], [20], [21], to obtain initial solutions. This reliance helps improve computational efficiency and allows these methods to focus on optimizing smoother trajectories, as well as determining the limit of planning success rate.

B. Learning-Based Parking Path-Planning

Learning-based methods represent a potential avenue to enhance the planner's performance in the task of parking. Existing methods mainly include imitating learning-based and reinforcement learning-based methods. Imitating learning requires the learnable planner to fit on the ground truth data. Liu et al. employed a neural network to align parking paths more closely with human behavior [22]. Rathour et al. derived a parking policy by imitating trajectories collected from expert drivers [23]. Other works include implementing the deep neural network or deep recurrent neural network as the behavior cloner on real vehicles [24], [25]. However, imitation learning-based methods are not good at handling the scenarios beyond their training sets, and the behavior cloners cannot outperform their imitation targets in terms of performance [26].

In RL-based approaches, the agents are developed without any labeled ground truth. The Monte Carlo tree search method was applied to search the available path in parallel parking scenarios [27]. Bernhard et al. learned the heuristics with Deep Q-learning (DQL) and improved the path search process in the Hybrid A* algorithm [28], [29]. Du et al. directly employed DQL to train an agent to produce single-step path planning results iteratively in a parallel and vertical case [30]. Yuan et al. proposed a hierarchical planning approach where high-level RL agents are trained to generate initial reference solutions [31]. The existing works typically map the elements of the task to the components of RL and directly employ them to train the agent in a limited number of scenarios. As more complex and diverse parking scenarios are introduced, it would be harder for RL-based methods to explore the proper parking policy, making it challenging to obtain planners with generalization capabilities.

III. Preliminaries

A. Reinforcement Learning for Path Planning

The RL problem can be addressed as policy learning in a Markov decision process (MDP) defined by a 4-element tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the action space. In the path-planning problem, $s_t \in \mathcal{S}$ includes the vehicle position and orientation $p_t = (x_t, y_t, \theta_t)$ as well as other observable information about obstacles and target parking space. We selected velocity v and steering angle δ as the action space $\mathcal{A} = \{a = (v, \delta)\}$. p denotes the state transition probability density of the next state s_{t+1} given the current state s_t and action a_t. In practice, the transition is modeled using the single-track bicycle model, as implemented in [32], and this uncertainty is not considered in a deterministic environment. A reward $r = r(s_t, a_t)$ is given by the environment E based on the state and action in each interaction step. The objective is to learn the policy $\pi(a_t|s_t)$ that maximizes the future reward $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$ with a discounting factor $\gamma \in [0, 1]$. Given the start point p_0 and the target p_T, the feasible parking path $P = \{p_0, p_1, p_2, \ldots, p_T\}$ can be obtained iteratively using the policy π.

To obtain the optimal policy, reinforcement learning algorithms primarily use the action-value function to describe the reward in terms of expectation after taking action a_t in state s_t with policy π:

$$Q_\pi(s_t, a_t) = E_{i>t, s_i \sim E, a_i \sim \pi}[R_t | s_t, a_t]$$
$$V_\pi(s_t) = E_{a_t \sim \pi(\cdot|s_t)}[Q_\pi(s_t, a_t)], \quad (1)$$

and both the Q function and value function can be updated by the Bellman equation:

$$Q_\pi(s_t, a_t) = E_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma Q_\pi(s_{t+1}, a_{t+1})]$$
$$V_\pi(s_t) = E_{a_t \sim \pi(\cdot|s_t)}[r(s_t, a_t) + \gamma V_\pi(s_{t+1})]. \quad (2)$$

To demonstrate the improvement of our proposed method across different reinforcement learning approaches, in this paper, we chose the commonly used on-policy algorithm, Proximal Policy Gradient (PPO) [33], and off-policy algorithm, Soft Actor-Critic (SAC) [34], to obtain the reinforcement learning policy π_θ in HOPE.

1) PPO: PPO is a policy-gradient method that performs the gradient ascent in the trust region. To model the change between the old policy parameters $\pi_{\theta_{\text{old}}}$ before the update and the new parameters π_θ, a probability ratio is denoted as $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. The loss function is:

$$L(\theta) = \hat{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \quad (3)$$

where $\hat{A}_t = Q(s_t, a_t) - V(s_t)$ is the estimated advantage value at time step t. ϵ is a hyperparameter indicating how much $r_t(\theta)$ can deviate from 1. By taking the minimum of the two terms, the policy is updated within a certain range in each iteration to achieve stable and efficient training.

2) SAC: SAC is a maximum entropy reinforcement learning method that maximizes the cumulated reward as well as policy entropy. The loss function of SAC can be expressed as:

$$L(\theta) = E_{s_t \sim D} \left[\alpha \mathcal{H}(\pi(\cdot|s_t)) - Q(s_t, a_t) \right]. \quad (4)$$

$\mathcal{H}(\pi(\cdot|s_t) = \log \pi_\theta(a_t|s_t)$ is the augmented entropy term and is scaled with a temperature parameter α. With the entropy regularization, SAC encourages the agent to keep a diverse policy and explore different strategies.

B. Reeds-Shepp Curve

RS curve is designed to generate the shortest path that links two positions with the minimal steering radius of the vehicle and straight lines. It is mathematically proved that the shortest path belongs to 48 types of curves and can be represented by one of the following 9 expressions:[1]

$$C|C|C,\ CC|C,\ CSC,\ CC_u|C_uC,\ C|C_uC_u|C,$$
$$C|C_{\pi/2}SC,\ C|C_{\pi/2}SC_{\pi/2}|C,\ C|C|C,\ CSC_{\pi/2}|C. \quad (5)$$

Here, C represents an arc segment with left or right steering, and S represents a straight-line segment. | means the path segment after it uses an opposite direction compared to the one before it. $C_{\pi/2}$ refers to a circular arc in which the central angle is fixed at $\pi/2$, and the arcs noted C_u in one formula share the same central angle u.

When calculating the RS curve, the path scale is initially normalized based on the vehicle's minimal turning radius r_{min}. Subsequently, the lengths of all segments for each type of path are determined using the given scaled starting point $p_0 = (x_0/r_{min}, y_0/r_{min}, \theta_0)$, ending point $p_T = (x_T/r_{min}, y_T/r_{min}, \theta_T)$, and geometric constraints specified by the expression 5. The total length of each path is obtained by summing the lengths of all segments, and the curve with the shortest total length is then selected as the optimal RS curve. The path waypoints $P = \{p_0, p_1, \ldots, p_T\}$ along the entire curve can be obtained through interpolation based on the lengths and types of each curve segment. Additionally, the vehicle's steering angle at each waypoint can be calculated using the vehicle model.

[1]Visualization examples of RS curves can be found in https://tactics2d.readthedocs.io/en/latest/tutorial/interpolator_visualization/

IV. Methodology

A. Overview of Architecture

The overall RL-based path planning framework and network structure are shown in Figure 1. To enhance training efficiency, we employ a hybrid policy reinforcement learning approach. Our proposed HOPE combines the learnable policy from original reinforcement learning with a rule-based policy derived from the RS curve. In each interaction step, the agent outputs the action, namely the single-step path planning result, based on the current state given by the environment. The action is then adjusted using the action mask mechanism before interacting with the environment. We employ four forms of network input as state representation, including:

- Obstacle distance (vector-based) l_t: the value of the nearest distance to obstacles at certain angles, which will be introduced later in section IV-C1.
- Target position (vector-based) P_{tgt}: P_{tgt} is defined as a 5-elements tuple $(d, \cos(\theta_t), \sin(\theta_t), \cos(\phi_t), \sin(\phi_t))$, includes the information about distance to target parking spot's position d, orientation θ_t, and heading ϕ_t in the ego vehicle's coordinate system.
- Action mask (vector-based) f_{am}: the representation of max valid step size at different steering angles, which will be introduced in section IV-C2.
- Bird-eye-view information (image-based) I_{BEV}: a low-resolution depiction of the drivable area, target parking spot, and the historical trajectory of the ego vehicle.

A transformer-based structure with learnable view encoding is designed to fuse the inputs and get the outputs in the actor and critic network [35]. We also utilize the auto-encoder structure to pre-train the image encoder. More details of our network and reward function are discussed in Appendix A and B. The pseudo-code of our method is shown in Algorithm 1.

B. Hybrid Policy

To improve the exploring and training efficiency, we combine the RL policy π_θ and the derived RS policy π_{RS} to facilitate the planning process. The hybrid process at timestamp t can be expressed as a function $h_t : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ that maps the actions $a_t \sim \pi_\theta(\cdot|s_t)$ from RL policy and $a'_t \sim \pi_{RS}(\cdot|s_t)$ from RS policy to the hybridized action $\tilde{a}_t = h_t(a_t, a'_t) \sim \pi_h(\cdot|s_t)$. Such an action hybrid process can be regarded as the procedure of making a suitable switch between two actions. Specifically, it activates the rule-based policy at certain timestamps and otherwise the reinforcement learning policy is used.

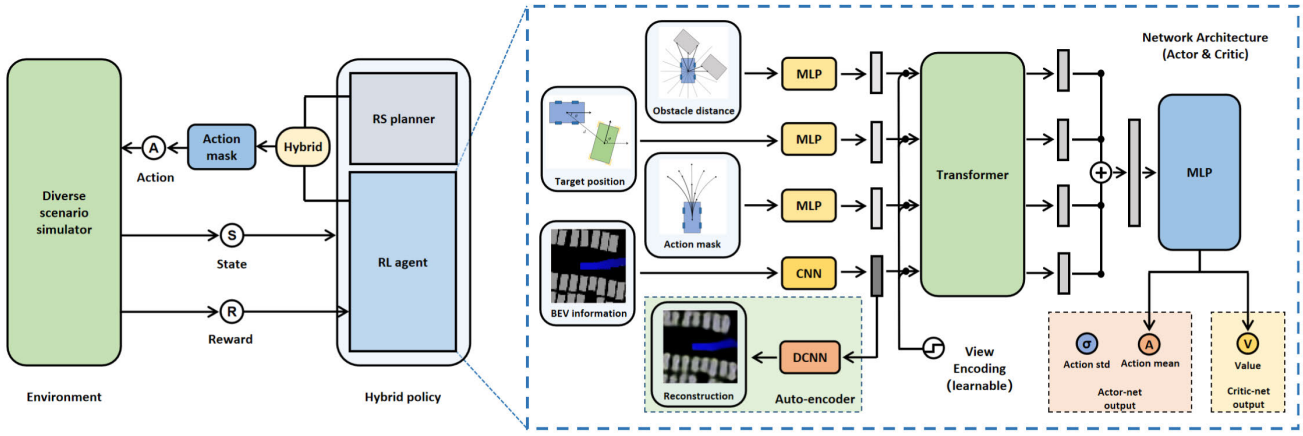


Fig. 1. The overall structure of the proposed method, including the interaction loop with the simulator (left) and the network architecture (right).

Algorithm 1 The Training Process of HOPE

Initialize the simulation environment E, the agent policy π_θ, the value network V_ψ and the Q network Q_ϕ, the hybrid policy π_h, the replay buffer D
for iteration $k = 0, 1, 2, \ldots$ **do**
  Reset the environment: $t \leftarrow 0$, $s_t, terminate \leftarrow$ E
  while $terminate$ is False:
    Choose action with the hybrid policy $a_t = \pi_h(s_t)$
    Get the masked action $\hat{a}_t$ using Equation 14, 15
    Interact with E: $s_{t+1}, r_t, terminate = E(\hat{a}_t)$
    Add to D: $D = D \cup (s_t, \hat{a}_t, r_t, s_{t+1}, terminate)$
    if update condition reached **then**
      (PPO) Update θ, ψ using Equation. 3, 6
      (SAC) Update θ, ϕ using Equation. 6, 8
    end if
    $t \leftarrow t + 1$
  end while
end for

1) Rule-Based Policy From Reeds-Shepp Curve: The nine expressions in Equation 5 represent 48 types of curves that give the shortest path in free space. However, in the presence of obstacles, the shortest RS curve may not be feasible. To implement the RS method in the parking scenario, we make two modifications in practice:

- We calculate the shortest K curves and assess their feasibility in ascending order of length, selecting the shortest feasible, collision-free path as the final route.
- We include paths of the $S(|)C(|)S$ form. These paths are excluded in the original RS method because they are strictly suboptimal in obstacle-free spaces. However, in environments with obstacles, these paths may still be feasible and beneficial, as indicated in Appendix D.

2) Exploring and Learning With RS Policy: During training, the agent explores and interacts with the environment using the hybridized action $\tilde{a}_t$ rather than raw action a_t. This choice also affects the updates on the policy π_θ and Q function or value function. For the Bellman equation in Equation 2, the update on the parameter V_ψ and Q_ϕ can be written as:

$$J_V(\psi) = E_{s_t \sim D_{\pi_h}} \left[\frac{1}{2} \left(V_\psi(s_t) - E_{\tilde{a}_t \sim \pi_h} \left[Q_\phi(s_t, \tilde{a}_t) \right] \right)^2 \right]$$

$$J_Q(\phi) = \mathbb{E}_{(s_t, \tilde{a}_t) \sim D_{\pi_h}} \left[\frac{1}{2} \left(Q_\phi(s_t, \tilde{a}_t) - r(s_t, \tilde{a}_t) - \gamma \mathbb{E}_{s_{t+1} \sim \rho_{\pi_h}} \left[V_\psi(s_{t+1}) \right] \right)^2 \right]. \quad (6)$$

Since the convergence of the Bellman equation for updating V_ψ and Q_ϕ is independent of the specific policy, when we replace the policy π with π_h, the iterative updates for V_ψ and Q_ϕ still converge. For policy updates in PPO, the probability ratio $r_t(\theta)$ in Equation 3 can be rewritten as:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(\tilde{a}_t|s_t)}, \quad \tilde{a}_t = h(a_t, a_t'). \quad (7)$$

The ratio clipping operation prevents excessive gradient updates caused by a significant KL divergence between π_θ and π_h. Meanwhile, Equation 4 in SAC can be rewritten as:

$$J_\pi(\theta) = E_{s_t \sim D_{\pi_h}} \left[\log \pi_\theta(a_t|s_t) - Q_\phi(s_t, a_t) \right]. \quad (8)$$

Generally, gradients on the Q function and value function are modified by using the data collected by π_h instead of π_θ, since they are now estimations on distribution ρ_{π_h}. We still update the original RL policy π_θ as shown in Equation 7 and 8, which indicates that the hybrid policy π_h, as well as the RL policy π_θ, are optimized. In practice, we apply a switch strategy that the RS policy is activated only when: 1) the distance from the vehicle's position to the target position is smaller than a threshold d_{rs}, and 2) there exists a collision-free RS curve from the vehicle's position to the target position. By the hybrid policy approach, during the initial phases of the learning process when the agent is not well-trained, the RS method serves as an alternative strategy to offer additional positive examples for updates. The agent can thus learn how to adjust the vehicle's pose to explore feasible parking routes.

C. Action Mask

Action masks have been employed in some discrete-space reinforcement learning tasks to filter out invalid actions,

which can enhance training efficiency and ensure alignment with deployment conditions [36]. By incorporating action masks, agents can focus on making complex decisions without spending extensive time learning calculable constraints. Additionally, this approach helps prevent invalid behaviors in applications. However, in path planning tasks, an action mask, which includes collision detection for all feasible actions of the agent vehicle across diverse state spaces, is computationally expensive. Here, we present a method for calculating and utilizing the action mask to enhance the training efficiency of reinforcement learning in path-planning tasks. Specifically, we use $Collide(s_t)$ to denote the event whether the vehicle at state s_t collides with obstacles. The action mask provides information about the largest safe step velocity v^* at any given steering angle δ:

$$v^* = \max_v\{Collide(s_{t+1}) = False\}, \tag{9}$$

where s_{t+1} is the new state after executing action $a = (v, \delta)$ at state s_t. By using v^* to constraint the raw speed v output by the actor net, we can find a collision-free new state using the masked action. Although calculating this maximum step size for a given action is always feasible, we need to compute the action mask before obtaining the final action and use it to influence the agent's planning. This implies the need to calculate the collision-free v for all given angles.

1) Efficient Estimation of Action Mask: We first introduce a vectorized obstacle distance representation l_t at timestamp t, where the i^{th} element $l_t[i]$ is the nearest obstacle distance at angle $\omega_i = i \cdot \Delta\omega$ in the ego coordinate and $\Delta\omega$ is the angular resolution. The movement distance during a certain period Δt with some speed v is denoted as step size $v\Delta t$. Consider the envelope area covered by the vehicle traveling with a steering angle δ_j and step size $v\Delta t$, denoted as $\mathrm{S}_{\mathcal{E}}(\delta_j, v\Delta t)$. Let $\mathcal{E}_{ij}(v)$ denote the distance from the envelope boundary to the origin in the ego vehicle's coordinate system at the i^{th} angle ω_i:

$$\begin{aligned}\mathcal{E}_{ij}(v) = max_e\{\|e\|_2 \mid e = (x, y) \in \mathrm{S}_{\mathcal{E}}(\delta_j, v\Delta t),\\ x\sin(\omega_i) = y\cos(\omega_i)\}.\end{aligned} \tag{10}$$

The collision constraint can then be expressed as $\mathcal{E}_{ij}(v) \leq l_t[i]$. Then, the action mask calculation is equivalent to the following problem by introducing a new 2-dim variable l:

$$\min\ l_{ij};\ \text{s.t.}\ l_{ij} = \mathcal{E}_{ij}(v),\ l_{ij} \leq l_t[i]. \tag{11}$$

While solving for the optimal l_{ij} considering only obstacle point at i^{th} angle ω_i and vehicle steering angle δ_j is equivalent to obtaining the maximum velocity, denoted as $v^*_{ij} = \mathcal{E}^{-1}_{ij}(l_t[i])$, calculating all envelope distances corresponding to each i and j in each interaction is computationally expensive, and the inverse of $\mathcal{E}_{ij}$ may not always exist. To deal with this issue, we propose the pre-calculation of the anchoring distance $\hat{l}_{ij}$ for K_v discretized velocities:

$$\hat{l}_{ij}[k] = \mathcal{E}_{ij}(\hat{v}[k]),\ \ \hat{v}[k] = v_{max}\frac{k}{K_v},\ \ k = 0, 1, \ldots K_v. \tag{12}$$

Then, we can obtain the upper and lower bounds of the masked velocity v^*_j considering all obstacle points at steering angle δ_j:

$$\begin{aligned}\hat{v}[k^*_j] \leq v^*_j < \hat{v}[k^*_j + 1],\\ k^*_j = \min_i \max\{k :\ l_t[i] - \hat{l}_{ij}[k] > 0\}.\end{aligned} \tag{13}$$

We take $v^*_j = \hat{v}[k^*_j]$ as a conservative estimation of max step velocity, and all actions at discretized angle δ_j are calculated in a vectorized manner at once. Since the anchoring distance $\hat{l}_{ij}$ is independent of information of obstacles at timestamp t, a matrix of all anchoring distance $\hat{\mathcal{L}}$ can be pre-calculated before the training process starts. This implies that, as shown in Equation 13, calculating the action mask at each interaction step is simplified to a comparison between two matrices.

2) Combine Action Mask With Agent's Policy: The action mask process can be expressed as a function f_{AM} from raw action to masked action:

$$\hat{a}_t = f_{AM}(\pi_h(s_t)) = h(f_{AM}(\pi_\theta(s_t)), \pi_{RS}(s_t)). \tag{14}$$

Here f_{AM} is applied only to π_θ because π_{RS} provides action only when a feasible RS curve exists. The action mask is also utilized to influence the probability distribution of actions. We here use $f_{am} : \mathcal{A} \rightarrow [0, 1]$ to denote the maximum step size calculated by the action mask, where $f_{am}(a) = p$ indicates the maximum safe step size is $p \cdot v_{max}\Delta t$. Noticed that $f_{am}(a)$ can serve as a prior probability of $Collide(s_{t+1}) = False$ when $a \sim [a_{min}, a_{max}]$, the action mask can be applied on the distribution of actions from raw network output:

$$log\,P(a_t|s_t) = \mathrm{SoftMax}(log(\pi_\theta(a_t|s_t)) + log f_{am}(a_t)). \tag{15}$$

The SoftMax here is the probability normalization operation, and $\pi_\theta(a_t|s_t)$ in practice we use a gaussian distribution:

$$log(\pi_\theta(a|s)) = -\frac{(a - a_{mean})^2}{2a^2_{std}} - log\left(\sqrt{2\pi}a_{std}\right), \tag{16}$$

where a_{mean} is obtained by actor network using input s and a_{std} is a learnable parameter, as shown in Figure 1. Equation 15 shows that the action mask can adjust the probability distribution of actions and avoid invalid actions by taking $f_{am}(a_t) = 0$. We also utilize the post-process in Equation 14 to clip the velocity into a collision-free range in practice.

V. Experiment

A. Scenario Difficulty Ranking

To better train and evaluate our approach, we rank the difficulty for parking scenarios into normal, complex, and extreme levels based on existing standards *ISO 20900* and *GB/T 41630-2022* [37], [38]. Generally, a parking space is defined by two boundary parking vehicles (or boundary obstacles) positioned on the two sides aligned along the curb-side edge of the road, as shown in Figure 2. The boundaries define the parking space's length L_{park} and width W_{park}. In our setup, other obstacles are located at least D_{obst} away from the parking spots to leave the drivable space. We denote the parking vehicle's width as W, length as L, and the distance from the start to the target position as D_{park}. Table I shows the categorization based on the aforementioned parameters, with narrower parking lots classified as more difficult. Since

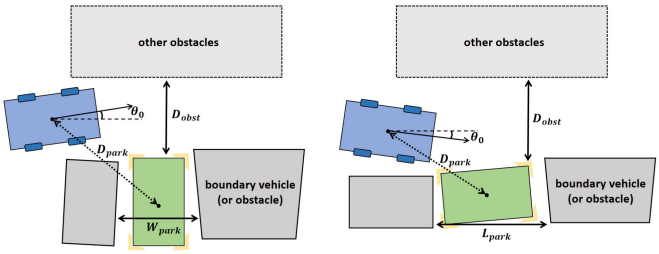


Fig. 2. The description for ranking parameters in vertical parking (left) and parallel parking (right).

TABLE I
PARKING SCENARIO DIFFICULTY

Difficulty	Parallel	Vertical
Normal	**P (N):** $D_{obst} > 4.5$ $L_{park} > max(L + 1.0, 1.25L)$	**V (N):** $D_{obst} > 7.0$ $W_{park} > W + 0.85$
Complex	**P (C):** $D_{obst} > 4.0$ $L_{park} > max(L + 0.9, 1.2L)$	**V (C):** $D_{obst} > 6.0$ $W_{park} > W + 0.4$
Extreme	**P (E):** $D_{obst} > 3.5$ $L_{park} > max(L + 0.6, 1.1L)$	/

V: vertical, P: parallel. N, C, and E in the bracket indicate the parking scenario difficulty, i.e., normal, complex, and extreme. unit: meter

parallel parking is more challenging than vertical parking in practice, we introduce an extreme difficulty level for it [38]. Besides, since larger parking distances require more maneuvering over a longer distance and increase the number of obstacles encountered along the way, we rank the scenarios that $D_{park} > 15.0$ the complex scenario. Note that we do not specify the initial orientation or position of the vehicle, meaning the starting conditions for the vehicle can be any collision-free configuration, which adds to the difficulty and diversity of scenarios.

Based on the difficulty ranking method, scenarios can be generated and categorized using the pre-defined parameters. In our work, the scenarios are derived from two components:

- Random generation using simulator: As shown in Figure 2, obstacles and the parking spot in a scenario can be represented by multiple randomized parameters. Obstacles may include other stationary vehicles or irregular polygonal obstacles. We randomly set the initial orientation of the start with a Gaussian distribution with $mean(\theta_0) = 0$ and $std(\theta_0) = \pi/6$, and the initial position can be anywhere without collision between parking spots and other obstacles.
- Real-world scenario dataset: We utilize the Dragon Lake Parking (DLP) dataset to construct the DLP scenarios in our simulator [39]. This dataset is constructed from 3.5 hours of video data collected by a drone in a large parking lot, covering an area with about 400 parking spots and 5188 vehicles. While the original dataset was designed for intention and motion prediction tasks, we filtered out non-parking trajectories and dynamic interfering vehicles to obtain 253 static parking scenarios. The starting positions in these parking scenarios are randomly initialized along the recorded paths of the vehicles. These scenarios can be categorized into vertical parking with normal and complex difficulty levels.

TABLE II
PLANNING SUCCESS RATE IN DIFFERENT SCENARIOS

Algorithms	V(N)	P(N)	V(C)	P(C)	P(E)	D(N)	D(C)
RS	36.9	10.4	30.4	1.5	0.3	4.0	0.1
Hybrid A*	99.4	90.2	99.2	60.2	16.8	98.7	85.6
EBHS	96.4	95.3	92.4	89.1	43.2	91.8	61.4
PPO	93.2	74.2	82.9	69.0	58.4	65.2	34.2
SAC	93.8	33.7	92.9	29.6	18.9	33.3	32.7
HOPE(PPO)	**100.0**	99.4	99.8	97.5	94.2	**99.5**	97.6
HOPE(SAC)	**100.0**	**99.7**	**100.0**	**99.4**	**97.5**	99.4	**98.0**

D: DLP scenarios. unit: percentage

B. Implement Details

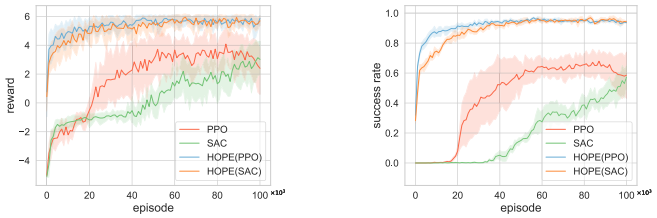
We conducted experiments in Tactics2D, an open-source simulator for driving decision-making [40]. This simulator provides sensor simulations, including lidar and bird's-eye view (BEV). In each episode, the simulator independently and randomly initializes the parallel or vertical parking scenarios of three difficulty levels generated by the simulator or from the DLP dataset. We conducted 100,000 episodes of training in total and tested 2,000 trials in each scenario category, with no overlap between training and testing scenarios. Each time the scenario and its parameters are randomly generated according to the ranking method specified in section V-A. Details about hyperparameters for algorithm and simulation are listed in Table VII in the appendix.

C. Results

1) Comaparion With Baselines: We compare the proposed method with the following baselines:

- RS method [8]: The RS method is a classical geometric-based approach that calculates possible path types to obtain feasible routes. In our experiments, we enhance the original RS method by utilizing all calculated paths instead of only the shortest one.
- Hybrid A* [15]: Hybrid A* is a search-based planning method incorporating heuristics considering obstacles and vehicle's non-holonomic constraints during node search. This method is widely applied in various planning tasks in the autonomous driving domain.
- Experience-based heuristic search method (EBHS) [29]: The EBHS leverages reinforcement learning to train a Q-network that serves as a heuristic function within the heuristic search algorithm. This learning-based approach allows the algorithm to derive more suitable heuristic functions from data and improve the exploring efficiency.
- Reinforcement learning baselines: When applying reinforcement learning methods to a specific task, a common practice is to establish a correspondence between the task elements and reinforcement learning components, using a deep network for function approximation. We explored this naive approach in our work, and experiments revealed that both PPO and SAC, in this manner, perform poorly in our parking path planning task.

Table II presents the planning success rates of these methods across different scene types and difficulty levels. We have separately listed the results for DLP scenarios since they originate from real-world parking environments. As shown in the table,



(a) (b)

Fig. 3. The episode reward curves (a) and success rate (b).

a direct implementation of the RS method fails to provide a feasible path in most cases. Hybrid A* achieves a success rate of over 90% in both normal scenarios and complex vertical parking cases. However, as the scene difficulty increases, its planning success rate significantly declines. In the extreme parallel parking scenarios, the most narrow cases, its success rate is only 16.8%. EBHS shows a significantly higher success rate in parallel parking scenarios than Hybrid A*. Specifically, in complex and extreme parallel parking scenarios, EBHS improves the success rate by nearly 30% over Hybrid A*. This improvement is attributed to the learning-based Q-function in EBHS, which can better estimate the state values in these scenarios compared to the rule-based heuristic function in Hybrid A*. However, the overall success rate of EBHS is still not high enough to be considered robust. The reinforcement learning baselines perform better than the rule-based RS method but fail to surpass the Hybrid A*. Their low success rates in several scenarios demonstrate that achieving good performance through overfitting in a few cases does not necessarily guarantee the trained agent's generalization capability. In contrast, our proposed HOPE, no matter based on on-policy PPO or off-policy SAC, outperforms all baselines and reaches a success rate of over 99.4% in all normal scenarios and over 94% in all scenarios. Figure 3 shows the reward and success rate curves in the training process. The proposed method significantly improves training efficiency and success rate over the naive RL method by combining the RL agent and the RS policy.

2) Further Comparison With Hybrid A:* As the Hybrid A* is a widely used method to this day, we further compare our approach with it through specific case studies. As shown in Figure 4, while both two methods can mask successful path planning in some cases, our method is capable of providing more concise and reasonable planning results, such as in Figure 4 (b) and (d). As the parking space in the scene becomes narrow, Hybrid A* fails to explore a feasible path, as shown in Figure 4 (c) and (e). In contrast, our method, through training, achieves the ability to maneuver within tight parking spaces and overcome local optima situations.

3) Computational Consumption: The average time consumption for a single-step prediction is 8.5 ms and can be broken down as follows: 2.7 ms for a single-step network forwarding, 2.8 ms for action mask calculation, and 3.0 ms for RS curve calculation. The simulator takes 8.3 ms each step for kinematics simulation and other information rendering. The total time required to generate a complete planning result, including algorithmic computation and simulator simulation overhead, is shown in Table III.

TABLE III
TIME COST OF COMPLETE PATH GENERATION ON AVERAGE

Alg.	V(N)	P(N)	V(C)	P(C)	P(E)	D(N)	D(C)
HOPE(PPO)	314.6	451.6	369.1	549.8	891.4	**433.0**	699.0
HOPE(SAC)	**304.4**	**372.3**	**328.0**	**476.6**	**638.4**	464.8	**633.2**

unit: microsecond(ms)

TABLE IV
EXPERIMENT ON THRESHOLD DISTANCE FOR RS CURVE

Alg.	d_{rs}	V(N)	P(N)	V(C)	P(C)	P(E)	D(N)	D(C)
HOPE (PPO)	1	99.2	97.4	95.4	95.0	92.6	97.1	95.0
	5	99.6	98.7	97.6	96.1	93.2	99.5	96.7
	10	**100.0**	**99.4**	**99.8**	**97.5**	**94.2**	99.5	97.6
	15	**100.0**	99.3	**99.8**	96.6	**94.2**	**99.9**	**98.4**
HOPE (SAC)	1	99.8	97.8	99.4	98.9	97.8	95.8	89.1
	5	**100.0**	98.0	99.8	99.2	97.7	99.4	97.3
	10	**100.0**	**99.7**	**100.0**	**99.4**	97.5	**99.4**	98.0
	15	**100.0**	99.4	**100.0**	99.0	**98.2**	99.0	**98.1**

D. Ablation Studies

1) Hybrid Policy With RS-Curve: We designed experiments to investigate how the RS method usage influences the performance of the hybrid policy. A hyperparameter in the hybrid strategy is the threshold distance d_{rs}. The RS policy is considered only when the vehicle is closer to the target position than d_{rs}. The results show that even reducing d_{rs} to 1 m has less than a 5% impact on the success rate while increasing d_{rs} does not lead to a significant improvement. This result indicates that while RS curves assist in training the hybrid policy, the reinforcement learning agent does not overly rely on the RS method, allowing it to outperform rule-based approaches.

Besides, while the original RS method only utilizes the shortest path, the shortest K paths are considered in the hybrid policy. In practice, we choose $K = 2$ to avoid redundant computations. At this point, the algorithm's performance has nearly saturated, as shown in Figure 5.

2) Other Proposed Modules: More ablation experiment results are presented in Table V. In the absence of the action mask mechanism, the success rate declines by more than 20% in complex parallel parking scenarios and 30% in extreme scenarios. This significant drop in performance indicates that utilizing an action mask to influence and constrain the agent's action is crucial for training effectiveness. Besides, replacing the transformer with a multi-layer perceptron leads to a decline of more than 10% in several scenarios, demonstrating the importance of multi-modal information fusion. In situations where BEV data is unavailable, our model can still work effectively by removing the BEV-related input branch from the transformer module. This adjustment does not lead to significant performance drops, and in some scenarios, it even achieves comparable performance to the original. Using the auto-encoder to pre-train the image encoder can enhance the training for the PPO-based agent but has a relatively minor impact on the SAC-based agent.

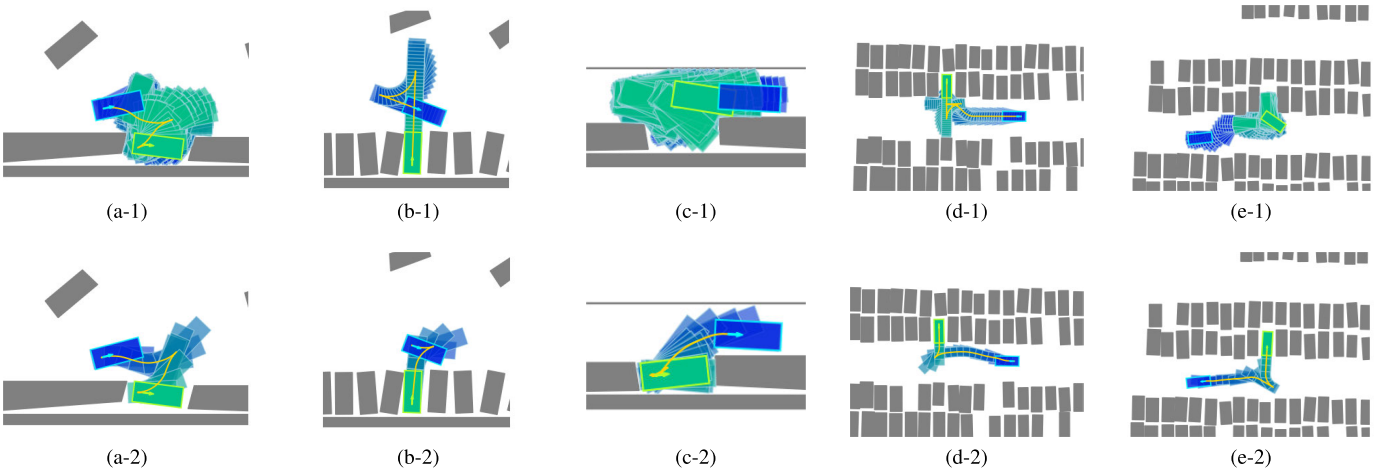


Fig. 4. The visualization of the planning process and results of the Hybrid A* (a-1)-(e-1) and the proposed HOPE (a-2)-(e-2). The blue-to-green gradient rectangles represent the states explored during the algorithm's search process, while the yellow curves indicate the path planning result. In the normal parallel parking case shown in (a), both methods provide concise path planning results. In the vertical parking scenario shown in (b) and the normal dlp scenario in (d), although both methods succeed in planning, our approach yields more reasonable results. In the narrow parallel parking scenario (c) and the scenario requiring parking with the front of the vehicle facing inward (e), the Hybrid A* fails to plan, while our approach succeeds.

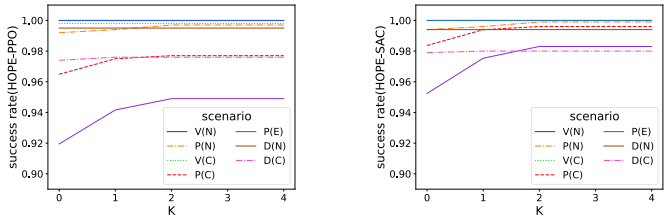


Fig. 5. Success rate using the shortest K RS curves.

TABLE V
ABLATION EXPERIMENT

Experiments	V(N)	P(N)	V(C)	P(C)	P(E)	D(N)	D(C)
HOPE(PPO)	**100.0**	**99.4**	99.8	**97.5**	**94.2**	**99.5**	**97.6**
W/O AMP	96.9	94.7	96.0	70.1	62.4	92.8	75.4
W/O TF	99.8	98.5	99.7	93.3	82.8	99.4	92.1
W/O AMI	99.6	99.0	98.1	97.3	91.0	98.8	94.4
W/O BEV	**100.0**	98.8	**99.9**	96.1	92.8	99.4	93.5
W/O AE	99.9	98.2	99.5	95.4	86.2	97.0	85.7
HOPE(SAC)	**100.0**	99.7	**100.0**	**99.4**	97.5	99.4	98.0
W/O AMP	96.5	49.1	96.3	36.5	23.4	78.2	57.2
W/O TF	99.8	90.6	98.8	82.3	72.1	97.8	87.9
W/O AMI	99.9	96.4	98.7	91.9	84.9	97.2	92.3
W/O BEV	99.8	97.8	98.7	97.9	**97.6**	98.1	96.5
W/O AE	**100.0**	**99.8**	**100.0**	99.0	96.7	**99.7**	**99.0**

W/O AMP: without the action mask in post-processing. W/O TF: use MLP instead of transformer as the backbone. W/O AMI: without the action mask in network input. W/O BEV: without the BEV image in network input. W/O AE: not employ auto-encoder to pre-train the image encoder.

3) Scenario Difficulty: We also examined the impact of restricting the training scenario difficulty on the agent's performance. As shown in Table VI, training solely on normal cases leads to a 20% to 30% decline in success rate in extreme scenarios, even with the help of the RS policy and action mask mechanism. Generalization capability under all difficulties can only be achieved when the training scenarios are sufficiently complex and diverse, which also underscores the importance of ranking difficulty in scenario taxonomy.

TABLE VI
EXPERIMENTS ON TRAINING SCENARIO DIFFICULTY

difficulty		V(N)	P(N)	V(C)	P(C)	P(E)	D(N)	D(C)
HOPE (PPO)	N	**100.0**	98.4	99.8	89.5	64.6	99.8	67.5
	+C	99.9	98.5	**99.9**	94.8	76.1	**99.9**	**98.1**
	+E	**100.0**	**99.4**	99.8	**97.5**	**94.2**	99.5	97.6
HOPE (SAC)	N	**100.0**	97.6	**100.0**	91.1	77.6	98.4	60.3
	+C	**100.0**	98.3	**100.0**	91.5	80.0	**99.7**	95.4
	+E	**100.0**	**99.7**	**100.0**	**99.4**	**97.5**	99.4	**98.0**

N: use only normal cases in training. +C: use normal and complex cases. +E: use cases of all difficulty levels.

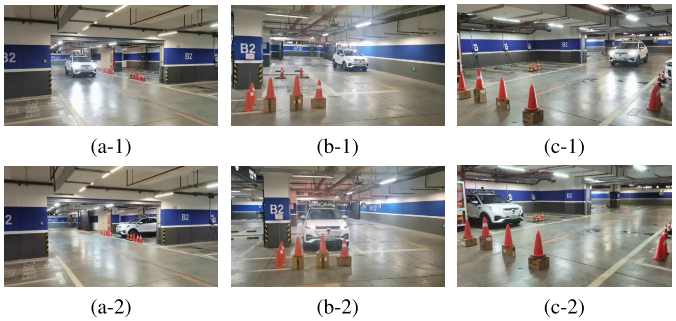


Fig. 6. The illustration of our real-world experiments in scenarios of vertical parking (a-1,2), parallel parking (b-1,2), and dead-end parking (c-1,2). The initial position is shown in the upper images and the lower images show the target parking lots where the vehicle is positioned.

E. Real-World Experiments

We also conducted real-world experiments in an indoor parking garage to show the applicability of our method in the real world. Our experimental platform was a Changan CS55E-Rock vehicle with a dimension of $4.6m \times 1.9m$ and we fine-tuned our model with the vehicle's parameter in the same simulation environment. More details about the platform and the overall system can be found in Appendix C.

We tested our method on vertical parking, parallel parking, and an additional challenging dead-end vertical parking scenario, as shown in Figure 6. To test the generalization ability of our algorithm, these scenarios are not included in the training set. In all three scenarios, our algorithm successfully planned

collision-free paths and guided the vehicle to complete the parking process. Detailed videos of the parking processes can be found at https://www.youtube.com/watch?v=62w9qhjIuRI. It is worth noting that the dead-end scenario in Figure 6 (c) represents a situation where the drivable area in front of the parking spot is narrow, with one side obstructed by obstacles. This requires the vehicle to perform extensive maneuvering and multiple gear shifts to adjust its orientation, making it a highly challenging scenario. These results demonstrate the practical applicability and generalization capability of our algorithm in real-world scenarios.

VI. Conclusion and Future Work

In this paper, we proposed a reinforcement learning-based path planning method in diverse parking scenarios with static obstacles. Instead of directly training the parking agent with existing RL methods, we introduced a hybrid policy that integrates the RS curve and PPO as well as SAC. To enhance the training efficiency and performance of reinforcement learning agents, we introduce a method to calculate and implement the action mask mechanism in the task of parking path planning. A transformer-based network structure is utilized to fuse information about obstacles and target parking spots from different inputs and output the planning actions.

To better train and evaluate the proposed method, we introduce a difficulty ranking approach for parking scenarios, which categorizes scenarios into normal, complex, and extreme based on obstacles and parking space conditions. We utilized a simulator and a real-world dataset to construct training and testing scenarios. Comprehensive experiments show the effectiveness of our method against other rule-based and RL-based methods. The results indicate that our approach outperforms the Hybrid A* in planning success rates across all types of scenarios, particularly in complex and extreme situations. Compared to the typical approach of directly using reinforcement learning, our proposed method combines the advantages of both rule-based and learning-based methods, resulting in an effective planner with generalization capabilities across different scenarios.

This paper shows that the learning-based approach can serve as a promising tool in dealing with complex and diverse parking scenarios. The proposed HOPE serves as a potential planner to deal with the intricate scenarios where the rule-based methods fail to plan a feasible path. Besides, since only static scenarios are considered in our work, our method can be integrated with a dynamic obstacle avoidance system to handle real-world participants to ensure compatibility and safety in dynamic environments. In the future, more efforts could be made to apply the learning-based method directly in parking scenarios with dynamic obstacles. To support this, enhancing simulation environments with interactive traffic participant models that realistically react to the autonomous vehicle would greatly improve the training and evaluation of such methods.

Appendix A
Implementation Detials

For the implementation of the information fusion transformer, we employ a multi-layer perceptron model for vector-based input to encode it into a feature token. For image-based input, we use a convolution neural network (CNN) with residual blocks to extract 2D features, which are later flattened to match the dimensions of other tokens. Tokens from different views are then fused in the transformer encoder, and their values are selected as outputs. Finally, an MLP-based network is used to map the concatenation of the values to the target output, namely the action for actor-net or the estimated value for value-net. Moreover, we use an auto-encoder structure to train the encoder in self-supervision. The CNN encoder extracts the features from the original image, and a deconvolutional neural network is used to reconstruct the image from the features. The pre-training data is collected with a naive agent trained initially without image input. The loss function of the auto-encoder using a mean square error can be expressed as:

$$L_{\text{AE}} = \|\text{Decoder}(\text{Encoder}(I_{BEV})) - I_{BEV}\|_2^2. \tag{17}$$

The key parameters for the algorithm and simulation are listed in Table VII. Besides, the training process was conducted on an NVIDIA GeForce RTX 3090 GPU and AMD EPYC 7542 CPU, and evaluation was performed on an NVIDIA GeForce RTX 3060 GPU and Intel i7-11800H CPU.

Appendix B
Reward Function

A basic reward design involves assigning a positive reward r_{succ} when the goal is completed and a negative penalty r_{fail} when the interaction terminates in failure. Based on this, we design several additional step rewards to encourage the agent to get close to the success state.

A. Intersection-of-union(IOU) Reward

The IOU area of the vehicle bounding box and the target parking spot is selected as guidance. We denote $\text{IOU}(s_t)$ as the IOU area of vehicle at time step t, and the IOU reward is:

$$r_{IOU}(t) = \max\left(\text{IOU}(s_t) - \max_{i\in(0,t-1)} \text{IOU}(s_i), 0\right). \tag{18}$$

This reward focuses on the increase of the IOU area compared to the largest value reached before and does not penalize it when the vehicle attempts to move out of the parking spot.

B. Distance Reward

We assign a positive reward to the agent when the vehicle is getting closer to parking spots:

$$r_{dist}(t) = -\frac{(D(t) - D(0))}{\max(D(0), D_{min})}. \tag{19}$$

$D(t)$ is the distance from the vehicle to the target position at time step t. D_{min} is a hyperparameter to avoid large rewards.

C. Time Consumption Penalty

For each interaction, the agent receives a small negative penalty as the cost of interaction, and this penalty increases over time. We bound the penalized value with $tanh$:

$$r_{time}(t) = -tanh(t/(10 \cdot T_{max})). \tag{20}$$

Here T_{max} is the maximum interaction times in an episode.

TABLE VII
PARAMETERS IN SIMULATION AND ALGORITHM

Parameter	Description	Value
lr_{actor}	Learning rate for the actor-network	5e-6
lr_{critic}	Learning rate for the critic-network	2.5e-5
$\|D\|$	Replay buffer size	8192
γ	Discount factor for rewards	0.98
ϵ	Clipping parameter in PPO	0.2
d_{rs}	Threshold distance for the RS policy	10.0
T_{max}	Maximum interaction times in a episode	200
n_{MLP}	Number of layers in the input MLP	2
n_{CNN}	Number of layers in the input CNN	2
d_{token}	The size of embedded tokens	128
d_{hidd}	The size of hidden layers	128
n_{attn}	Number of attention layers	1
n_{head}	Parameters of multi-head mechanism	8
n_{out}	Number of layers in the output MLP	2
Δt	Simulated time for an interaction step	0.5 s
$\Delta\omega$	Simulated lidar angular resolution	$\pi/60$ rad
$dim(l_t)$	The dimension of obstacle distance vector	120
$dim(f_{am})$	The dimension of action mask	42
R_{lidar}	Simulated maximum lidar range	10 m
$H_{img}(W_{img})$	BEV image height (width)	64 px
L	Vehicle length	4.69 m
W	Vehicle width	1.94 m
v_{max}	Maximum velocity	2.5 m/s
δ_{max}	Maximum steering angle	0.75 rad

D. Overall Reward Function

The above basic rewards and step rewards are linearly combined to form the final reward:

$$r = w_1 r_{succ} + w_2 r_{fail} + w_3 r_{IOU} + w_4 r_{dist} + w_5 r_{time}. \quad (21)$$

In practice, we set $r_{succ} = 5$, $r_{fail} = -5$, $w_1 = w_2 = w_3 = 1$, $w_4 = 0.5$ and $w_5 = 0.1$.

APPENDIX C
REAL-WORLD EXPERIMENT SETUPS

For sensors, we used Hesai Pandar40, RoboSense RS-LiDAR-32 and Bpearl LiDAR to support the vehicle's localization and perception modules and no GPS was included. Figure 7 illustrates the framework of the parking system and how our method operates in real-world scenarios. The input to the planning module, where our method is applied, consists of a BEV grid map, obstacle distance, and the position of the parking spot. The target parking spot is selected by the driver and its precise location is obtained from a high-definition map, then transformed into the ego coordinate system through the localization system. Obstacle distance information is derived from the LiDAR point cloud, while the BEV grid map is generated by augmenting the drivable area from the high-definition map with real-time detected obstacles from the LiDAR. To mitigate the impact of upstream perception errors, we also add virtual occupancy grids in other parking spaces in the BEV grid map based on high-definition map information, ensuring the vehicle does not drive into these areas during maneuvering. These inputs are then converted into the format required by the network, which outputs a one-step planned path. Finally, a Quadratic Programming (QP) and Model Predictive Control (MPC) approach is used for path following to obtain the control commands from the waypoints. Besides, we also utilize a collision avoidance module to give way to dynamic obstacles in practice. The vehicle only follows a path if it has been successfully planned as collision-free and is connected to the target parking position. All modules and our algorithm are deployed on an Intel i7-1165G7 2.80GHz CPU.

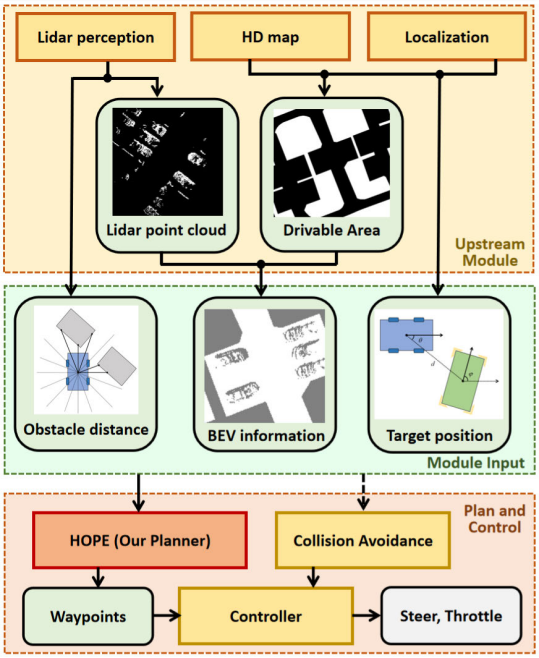


Fig. 7. The overall structure of the parking system, where our proposed planning module is in red.

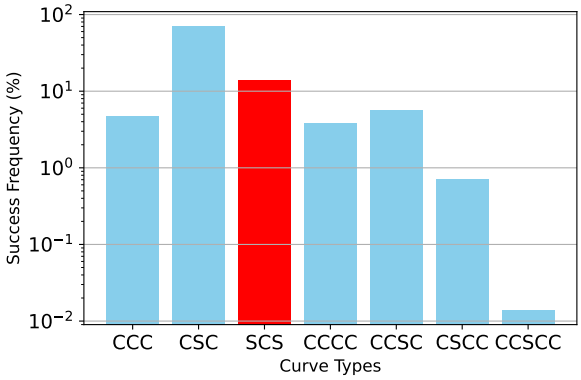


Fig. 8. The success frequency of different RS curve types.

APPENDIX D
EXPERIMENTS ON THE EXTENDED REEDS-SHEPP CURVES

We conducted experiments to evaluate the impact of different Reeds-Shepp (RS) curve types on the success rate of parking maneuvers. The results are summarized in Figure 8, where the horizontal axis represents different RS curve types, and the vertical axis represents the frequency of usage of each curve type in successful planning cases for all vertical parking scenarios. The results show that SCS curves have the second-highest frequency of 13.8%, demonstrating their effectiveness.

APPENDIX E
PROOF OF CONVERGENCE FOR THE HYBRID POLICY

The convergency of the update on the parameter V_ψ and Q_ϕ can be proven by showing the update process of the Bellman expectation operator is a contraction mapping, which is independent of specific policy π. Thus the convergency stands for any given π and so is the π_h. We here provide the proof of convergency of the value function V_ψ, and the Q function Q_ϕ can be obtained using a one-step iteration:

$$Q^\pi(s,a) = \mathbb{E}\left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s, A_t = a\right] \quad (22)$$

Firstly, the Bellman expectation equation for a policy π is defined as:

$$V_\pi(s_t) = \mathbb{E}_{a_t\sim\pi(\cdot|s_t)}[r(s_t,a_t) + \gamma V_\pi(s_{t+1})] \quad (23)$$

We define the Bellman expectation operator B_π as:

$$(B_\pi V)(s_t) = \mathbb{E}_{a\sim\pi(\cdot|s)}[r(s_t,a) + \gamma V(s_{t+1})] \quad (24)$$

To prove that the Bellman expectation operator B_π is a contraction mapping, we need to show:

$$\|B_\pi V_1 - B_\pi V_2\| \le \gamma \|V_1 - V_2\| \quad (25)$$

For any two value functions V_1 and V_2, we have:

$$\begin{aligned}
&|B_\pi V_1(s_t) - B_\pi V_2(s_t)| \\
&= \left|\mathbb{E}_{a\sim\pi(\cdot|s_t)}\left[r(s_t,a) + \gamma \mathbb{E}_{s_{t+1}\sim E}[V_1(s_{t+1})]\right]\right. \\
&\quad \left. - \mathbb{E}_{a\sim\pi(\cdot|s)}\left[r(s_t,a) + \gamma \mathbb{E}_{s_{t+1}\sim E}[V_2(s_{t+1})]\right]\right| \\
&= \gamma \left|\mathbb{E}_{a\sim\pi(\cdot|s_t)}\left[\mathbb{E}_{s_{t+1}\sim E}[V_1(s_{t+1}) - V_2(s_{t+1})]\right]\right| \\
&= \gamma \left|\mathbb{E}_{a\sim\pi(\cdot|s_t),s_{t+1}\sim E}[V_1(s_{t+1}) - V_2(s_{t+1})]\right| \\
&\le \gamma \mathbb{E}_{a\sim\pi(\cdot|s_t),s_{t+1}\sim P(\cdot|s_t,a)}\left[|V_1(s_{t+1}) - V_2(s_{t+1})|\right] \\
&\le \gamma \sup_{s_{t+1}} |V_1(s_{t+1}) - V_2(s_{t+1})| \\
&= \gamma \|V_1 - V_2\|_\infty
\end{aligned}$$

This shows that B_π is a contraction mapping because $\|B_\pi V_1 - B_\pi V_2\|_\infty \le \gamma \|V_1 - V_2\|_\infty$ for $0 \le \gamma < 1$.

According to the Banach fixed-point theorem, a contraction mapping B_π on a complete metric space $(\mathbb{R}^S, \|\cdot\|_\infty)$ has a unique fixed point V_π^*, i.e.,

$$B_\pi V_\pi^* = V_\pi^* \quad (26)$$

Therefore, through the iterative update process:

$$V_{k+1} = B_\pi V_k \quad (27)$$

the value function V_k will converge.

The convergence proof above shows that the Bellman expectation operator B_π is a contraction mapping and hence will converge to a fixed point for any given policy π. This means that the convergence of the value function updates is not dependent on the specific policy π. Therefore, even if we replace the policy π with another policy π_h, the convergence properties still hold. We can use the following loss function to update the value function:

$$J_V(\psi) = \mathbb{E}_{s_t\sim D_{\pi_h}}\left[\frac{1}{2}\left(V_\psi(s_t) - \mathbb{E}_{\tilde{a}_t\sim\pi_h}\left[Q_\phi(s_t,\tilde{a}_t)\right]\right)^2\right] \quad (28)$$

We minimize this loss function to update the parameter ψ:

$$\psi_{k+1} = \psi_k - \alpha \nabla_\psi J_V(\psi_k) \quad (29)$$

This loss function minimizes the mean squared error between $V_\psi(s_t)$ and $\mathbb{E}_{\tilde{a}_t\sim\pi_h}\left[Q_\phi(s_t,\tilde{a}_t)\right]$. According to the Bellman expectation equation:

$$V_\pi(s_t) = \mathbb{E}_{\tilde{a}_t\sim\pi_h}\left[Q_\phi(s_t,\tilde{a}_t)\right] \quad (30)$$

minimizing $J_V(\psi)$ is equivalent to approximating the Bellman expectation equation.

REFERENCES

[1] Y. Song and C. Liao, "Analysis and review of state-of-the-art automatic parking assist system," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Jul. 2016, pp. 1–6.

[2] W. Wang, Y. Song, J. Zhang, and H. Deng, "Automatic parking of vehicles: A review of literatures," *Int. J. Automot. Technol.*, vol. 15, no. 6, pp. 967–978, Oct. 2014.

[3] B. Li et al., "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11970–11981, Aug. 2021.

[4] A. Likmeta, A. M. Metelli, A. Tirinzoni, R. Giol, M. Restelli, and D. Romano, "Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving," *Robot. Auto. Syst.*, vol. 131, Sep. 2020, Art. no. 103568.

[5] S. Teng et al., "Motion planning for autonomous driving: The state of the art and future perspectives," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 6, pp. 3692–3711, Jun. 2023.

[6] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[7] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *J. Field Robot.*, vol. 37, no. 3, pp. 362–386, Apr. 2020.

[8] J. Reeds and L. Shepp, "Optimal paths for a car that Goes both forwards and backwards," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, Oct. 1990.

[9] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.

[10] L. Zhao, G. Zheng, and J. Li, "Automatic parking path optimization based on Bezier curve fitting," in *Proc. IEEE Int. Conf. Autom. Logistics*, Sep. 2012, pp. 583–587.

[11] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking with geometric continuous-curvature path planning," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 465–471.

[12] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Amer. J. Math.*, vol. 79, no. 3, pp. 497–516, 1957.

[13] X. Du and K. K. Tan, "Autonomous reverse parking system based on robust path generation and improved sliding mode control," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1225–1237, Jun. 2015.

[14] J. M. Kim, K. I. Lim, and J. H. Kim, "Auto parking path planning system using modified Reeds–Shepp curve algorithm," in *Proc. 11th Int. Conf. Ubiquitous Robots Ambient Intell. (URAI)*, Nov. 2014, pp. 311–315.

[15] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, Apr. 2010.

[16] S. Sedighi, D. Nguyen, and K. Kuhnert, "Guided hybrid A-star path planning algorithm for valet parking applications," in *Proc. 5th Int. Conf. Control, Autom. Robot. (ICCAR)*, Apr. 2019, pp. 570–575.

[17] W. Sheng, B. Li, and X. Zhong, "Autonomous parking trajectory planning with tiny passages: A combination of multistage hybrid A-star algorithm and numerical optimal control," *IEEE Access*, vol. 9, pp. 102801–102810, 2021.

[18] M. Czubenko, Z. Kowalczuk, and A. Ordys, "Autonomous driver based on an intelligent system of decision-making," *Cognit. Comput.*, vol. 7, no. 5, pp. 569–581, Oct. 2015.

[19] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, May 2021.

[20] R. He et al., "TDR-OBCA: A reliable planner for autonomous driving in free-space environment," in *Proc. Amer. Control Conf. (ACC)*, May 2021, pp. 2927–2934.

[21] Z. Han et al., "An efficient spatial–temporal trajectory planner for autonomous vehicles in unstructured environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 2, pp. 1797–1814, Feb. 2024.
[22] W. Liu, Z. Li, L. Li, and F.-Y. Wang, "Parking like a human: A direct trajectory planning solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3388–3397, Dec. 2017.
[23] S. Rathour, V. John, M. K. Nithilan, and S. Mita, "Vision and dead reckoning-based end-to-end parking for autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 2182–2187.
[24] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. P. Chen, "Design and implementation of deep neural network-based control for automatic parking maneuver process," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1400–1413, Apr. 2022.
[25] R. Chai, D. Liu, T. Liu, A. Tsourdos, Y. Xia, and S. Chai, "Deep learning-based trajectory planning and control for autonomous ground vehicle parking maneuver," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 3, pp. 1633–1647, Jul. 2023.
[26] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 4565–4573.
[27] S. Song, H. Chen, H. Sun, M. Liu, and T. Xia, "Time-optimized online planning for parallel parking with nonlinear optimization and improved Monte Carlo tree search," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2226–2233, Apr. 2022.
[28] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[29] J. Bernhard, R. Gieselmann, K. Esterle, and A. Knoll, "Experience-based heuristic search: Robust motion planning with deep Q-learning," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 3175–3182.
[30] Z. Du, Q. Miao, and C. Zong, "Trajectory planning for automated parking systems using deep reinforcement learning," *Int. J. Automot. Technol.*, vol. 21, no. 4, pp. 881–887, Aug. 2020.
[31] Z. Yuan, Z. Wang, X. Li, L. Li, and L. Zhang, "Hierarchical trajectory planning for narrow-space automated parking with deep reinforcement learning: A federated learning scheme," *Sensors*, vol. 23, no. 8, p. 4087, Apr. 2023.
[32] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. IEEE Intell. Veh. Symp.*, Oct. 2017, pp. 719–726.
[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[35] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, "Safety-enhanced autonomous driving using interpretable sensor fusion transformer," in *Proc. Conf. Robot Learn.*, 2023, pp. 726–737.
[36] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," 2020, *arXiv:2006.14171*.
[37] *Intelligent Transport Systems—Partially-Automated Parking Systems (PAPS)—Performance Requirements and Test Procedures*, Standard ISO 20900, 2023.
[38] *Performance Requirements and Test Methods for Intelligent Parking Assist System*, Standard GB/T 41630, 2022.
[39] X. Shen, M. Lacayo, N. Guggilla, and F. Borrelli, "ParkPredict+: Multimodal intent and motion prediction for vehicles in parking lots with CNN and transformer," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2022, pp. 3999–4004.
[40] Y. Li et al., "Tactics2D: A highly modular and extensible simulator for driving decision-making," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 5, pp. 4840–4844, 2024.

Mingyang Jiang received the bachelor's degree in engineering from Shanghai Jiao Tong University, Shanghai, China, in 2023, where he is currently pursuing the master's degree in control science and engineering. His main research interests include end-to-end planning, driving decision-making, and reinforcement learning for autonomous vehicles.

Yueyuan Li received the bachelor's degree in electrical and computer engineering from the University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai, China, in 2020. She is currently pursuing the Ph.D. degree in control science and engineering with Shanghai Jiao Tong University.

Her main fields of interest are the security of the autonomous driving system and driving decision-making. Her current research activities include driving decision-making models, driving simulation, and virtual-to-real model transferring.

Songan Zhang received the B.S. and M.S. degrees in automotive engineering from Tsinghua University in 2013 and 2016, respectively, and the Ph.D. degree in mechanical engineering in 2021. Then, she went to the University of Michigan, Ann Arbor. After graduation, she worked with the Robotics Research Team, Ford Motor Company, as a Research Scientist. Presently, she is an Assistant Professor with the Global Institute of Future Technology (GIFT), Shanghai Jiao Tong University. Her research interests include accelerated evaluation of autonomous vehicles, model-based reinforcement learning, and meta-reinforcement learning for autonomous vehicle decision-making.

Siyuan Chen received the bachelor's degree in engineering from Shanghai Jiao Tong University, Shanghai, China, in 2023, where he is currently pursuing the master's degree in control science and engineering. His main research interests include planning and control and V2X systems for autonomous vehicles.

Chunxiang Wang received the Ph.D. degree in mechanical engineering from Harbin Institute of Technology, China, in 1999.

She is currently an Associate Professor with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. She has been working in the field of intelligent vehicles for more than ten years and has participated in several related research projects, such as European CyberC3 project and the ITER transfer cask project. Her research interests include autonomous driving, assistant driving, and mobile robots.

Ming Yang (Member, IEEE) received the master's and Ph.D. degrees from Tsinghua University, Beijing, China, in 1999 and 2003, respectively. Presently, he holds the position of a Distinguished Professor with Shanghai Jiao Tong University and the Director of the Innovation Center of Intelligent Connected Vehicles. He has been engaged in the research of intelligent vehicles for more than 25 years.