scientific reports

Check for updates

OPEN

Automatic parking trajectory planning in narrow spaces based on Hybrid A* and NMPC

Pei Zhang[1,2], SiLong Zhou[1,2], Jie Hu[1,2]✉, WenLong Zhao[1,2], Jiachen Zheng[1,2], Zhiling Zhang[1,2] & Chongzhi Gao[3]

The rapid acceleration of urbanization and the surge in car ownership necessitate efficient automatic parking solutions in constricted spaces to address the escalating urban parking issue. To optimize space utilization, enhance traffic efficiency, and mitigate accident risks, a method is proposed for smooth, comfortable, and adaptable automatic parking trajectory planning. This study initially employs a hybrid A* algorithm to generate a preliminary path, then fits the velocity and acceleration based on a cubic polynomial. The kinematic constraints of the vehicle and obstacle avoidance constraints are then meticulously defined, and a coupled nonlinear model predictive control (NMPC) method is employed to optimize the trajectory. Compared to the hybrid A* algorithm, the optimized trajectory demonstrates superior space utilization and improved smoothness. The experimental results indicate that the proposed method performs effectively in automated parking tasks in confined spaces, suggesting promising applications and broad prospects for future.

Keywords Automatic parking, Hybrid A* algorithm, Cubic polynomial, NMPC, Optimal control, Collision constraint

Over the past decade, substantial advancements have been made in the field of autonomous driving vehicles. Given the current surge in vehicle numbers and congested parking spaces[1], there is a burgeoning interest in the field of automated parking[2]. Within the array of components integral to automated parking technology, trajectory planning is a fundamental technique. The intricate nature of vehicle motion, characterized by nonholonomic constraints, coupled with confined driving spaces, presents substantial challenges for the successful implementation of automated parking[3]. The challenge lies in navigating the narrow parking spaces, considering the nonholonomic constraints of the vehicle, and generating feasible and smooth trajectories while avoiding collisions with obstacles in the environment. At present, the most prevalent algorithms for automated parking trajectory planning include geometric methods, sampling-based methods, search-based methods, optimal control-based methods, and intelligent algorithms.

In terms of geometric methods, Kim et al.[4] proposed a local path planning algorithm based on the approximate clothoid and demonstrated its stability using the Lyapunov stability theorem. Ghajar et al.[5] proposed a geometric-based method for local parking path planning that considers both nonholonomic constraints and obstacle collisions and has a short computation time. Huang et al.[6] combined geometric curve planning techniques, such as sixth-degree polynomial, clothoid, and Bezier curves, to plan parking paths with continuous curvature. Song et al.[7] used B-spline curves for parking path planning while imposing constraints to avoid collisions but considered only collisions with parking spaces and road boundaries. Piao et al.[8] employed multiple circular arcs to generate parallel parking paths without the need for initial poses to be parallel to the parking space, but the curvature is discontinuous. To address the curvature discontinuity issue in combined straight line and circular arc paths, Zhang et al.[9] used fifth-degree polynomials for smoothing, Cai et al.[10] applied Bezier curves for curvature smoothing, and Chen et al.[11] utilized the ArcTanGent function for optimization.

Within the realm of sampling-based methods, Dong et al.[12] improved the traditional Rapidly-exploring Random Tree (RRT) by directly connecting branches using Reed-Shepp curves and adjusting the RRT sampling positions based on vehicle and parking spot information. Wang et al.[13] proposed an adaptive and bidirectional RRT* algorithm for narrow and multi obstacles environments, which extends the tree simultaneously from the

[1]Hubei Key Laboratory of Modern Auto Parts Technology (Wuhan University of Technology), Wuhan 430070, People's Republic of China. [2]Hubei Technology Research Center of New Energy and Intelligent Connected Vehicle Engineering, Wuhan 430070, People's Republic of China. [3]Commercial Product R&D Institute, Dongfeng Automobile Co., Ltd., Wuhan 430070, People's Republic of China. ✉email: auto_hj@163.com

start and end points and adjusts the sampling probability based on the number of collision detection failures. Gan et al.[14] presented the 1–0 Bg-RRT algorithm with less computation time.

Regarding search-based methods, Sedighi et al.[15] proposed an innovative and highly computationally efficient method that combines hybrid A* search engine visibility graph planning to find the shortest feasible nonholonomic path for valet parking in complex environments. On the basis of collision-free paths generated by RS curve, Li et al.[16] employed the hybrid A* algorithm to perform offline computation of navigation points in various working scenarios. Based on hybrid A* algorithm, Zhang et al.[17] came up with a layered parking path planning method and improved the search speed of hybrid A* algorithm in complex parking scenarios through dividing a complex path planning problem into several simpler problems. Meng et al.[18] proposed an improved hybrid A* algorithm that integrates Voronoi fields into the path search process to enhance the safety of trajectory. By introducing anchor points, He et al.[19] presented the FAST A* algorithm specifically designed for narrow parking spaces, which improves the success rate of trajectory planning while reducing computation time.

As the computed path acquired by search-based method may be far from the global optimum, the optimal control-based method is used widely for trajectory planning. According to the chosen objective function, necessary constraints and vehicle dynamics model, the computed trajectory by optimal control-based method is at least locally optimal. Zhang et al.[20] utilized the strong duality of convex optimization to transform non-differentiable obstacle constraints into smooth differentiable constraints, thereby enhancing the accuracy of obstacle constraints and the safety of automatic parking. Ma et al.[21] proposed a local-learning-enabled constrained Iterative Linear Quadratic Regulator (iLQR) based on hybrid dynamic optimization and machine learning, and improved trajectory planning efficiency in automatic parking compared with model-based iLQR. Duan et al.[22] introduced Discrete-time Control Barrier Functions (DCBF) and slack variables into Model Predictive Control (MPC), effectively addressing the challenges of traditional methods in balancing feasibility and safety. Zhou et al.[23] initially established a path set composed of multiple geometric curves, and fine-tuned the parameters of geometric curves during the parking process using Sequential Quadratic Programming (SQP) to avoid the impact of cumulative control errors on parking performance. Qiu et al.[24] presented a semi-trailer automatic parking trajectory planning method based on segmented gaussian pseudospectral method, which has faster computation speed and shorter parking time compared to traditional pseudospectral method.

In addition, several intelligent algorithms have also been applied in the field of automatic parking trajectory planning. Daniali et al.[25] employed the Multi-Objective Particle Swarm Optimization (MOPSO) to minimize parking time and parking space length. Yu et al.[26] transformed the parking path planning problem into an optimization problem with maximum curvature and curvature at both ends as objective functions, and used the simulated annealing algorithm for path optimization. Su et al.[27] proposed a genetic algorithm-based endpoint partitioning quadratic parallel automatic parking method to enhance the accuracy of automatic parking in narrow spaces. Han et al.[28] combined the grid method with an improved ant colony algorithm to find the destination by setting the ant's movement rules in the grid, and identified the shortest path for automatic parking. Tang et al.[29] introduced a segmented parking training framework based on Soft Actor-Critic (SAC) and incorporated policy entropy into the objective function to comprehensively consider environmental information during the learning process. Zhang et al.[30] proposed a reinforcement learning-based end-to-end automatic parking algorithm, enabling vehicle to learn the optimal steering angles for different parking spaces through continuous learning. Zhang et al.[31] achieved rapid and autonomous learning of automatic parking strategies without relying on expert data or prior knowledge through a model-driven reinforcement learning approach.

The comparison of various algorithms is summarized in Table 1.

In narrow spaces, both sampling-based and search-based planning methods have high success rates of trajectory planning. The sampling-based methods have strong randomness and unstable effect, so that hybrid A* algorithm is employed to deal with automatic parking in narrow space. The planning accuracy of hybrid A* algorithm is constrained by resolution of grid map. Although the hybrid A* algorithm considers vehicle kinematic constraints, path curvature changes calculated by this algorithm are discontinuous, which is unfavorable for control tracking. Furthermore, decoupling path and velocity obtained by the hybrid A* algorithm may lead to unnecessary shifting behaviors. To overcome these drawbacks of the hybrid A* algorithm, an optimal control algorithm called NMPC with longitudinal and lateral coupling is proposed. However, the pure NMPC algorithm

Method	Principle	Advantages	Disadvantages
Geometric method	Describe parking path curves through analytic expressions such as straight lines, arcs, spirals, typically derived from the geometric relationship between parking spots and vehicles	Exhibit fast computation speed, high stability, and generate paths with excellent smoothness	Poor self-adaptation to different parking spots and environments, with limited capability for handling complex scenarios
Sampling-based method	Begin by randomly sampling in space, connect various nodes, and design cost functions to evaluate each node, represented by RRT series	Enable efficient resolution of trajectory planning problems in high-dimensional spaces	The planned path may not satisfy the vehicle's kinematic constraints, lower stability caused by random sampling
Search-based method	Discretize space into grids and utilize dynamic programming to find the optimal solution connecting the start and end nodes, represented by the A* series	Good stability and a high planning success rate even in complex environments	The curvature of the path may be discontinuous and the solution accuracy is influenced by resolution
Optimal control-based method	Find the minimum or maximum value of the objective function while satisfying a series of constraints	Handle complex, nonlinear constraints, improve precision by increasing the number of finite elements	Possible to get trapped in a local optimal solution
Intelligent algorithm	Input the start and end positions along with environmental information, design a learning network, iterate model parameters, and then utilize this model to solve trajectory planning problems	Handle trajectory planning in complex scenarios	Require a large dataset for iterative training, poor interpretability and maintainability

Table 1. Comparison of commonly used algorithms.

is time-consuming to solve and prone to local optima. Based on above analysis, an improved trajectory planning method composed of hybrid A* and NMPC is presented in this study. The contributions of this paper are as follows:

1. The reverse hybrid A* search algorithm is utilized to improve the success rate of searching in narrow spaces, followed by the adoption of cubic polynomials and linear interpolation to obtain rough trajectories.
2. Based on precise obstacle contours described by OBB bounding box, the collision-free constraint is established to incorporated into optimal control of automatic parking trajectory planning, in order to make full use of the parking space.
3. The coupled NMPC algorithm is employed to optimize the path, smooth the rough trajectory and significantly reduce the number of gear shifts, making the parking maneuver more reasonable.

The remaining structure of this paper is as follows: "Vehicle model" introduces the vehicle kinematic model. "Automated parking planning strategy" provides a detailed explanation of the automated parking algorithm and designs scenarios for parallel parking and vertical parking to demonstrate the effectiveness of trajectory planning in narrow spaces. "Algorithm validation on actual vehicles" presents the experimental platform and communication mechanism for real-world vehicle experiments, validating the adaptability and feasibility of the algorithm. Finally, "Conclusions" concludes this paper.

Vehicle model

The vehicle model can be divided into two types: kinematic model and dynamic model. The kinematic model does not consider the vehicle's dynamics but focuses on the motion of the mass center. It is mainly used for vehicle planning and control in low-speed scenarios. In the parking scenario, where the vehicle speed is normally below 10 km/h, to simplify the system complexity, the kinematic model can be used to describe the vehicle's motion trajectory. The vehicle is considered as a rigid body moving in a two-dimensional plane. Assuming that the front wheels' steering angles and speeds are equal, furthermore, longitudinal and lateral load transfer is not considered, the vehicle kinematic model can be presented as Fig. 1. The state-space equations of the model are as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} cos\,(\theta)\,v \\ sin\,(\theta)\,v \\ tan\,(\delta)\,v/L \\ a \end{bmatrix} \tag{1}$$

where $\dot{x}$ and $\dot{y}$ are longitudinal velocity and lateral velocity of the mass center respectively; θ is the yaw angle; v is the speed; δ is the front wheel steering angle; L is the wheelbase; a is the acceleration.

Automated parking planning strategy

For the automatic parking scenario, a coarse path is planned using the hybrid A* algorithm. Cubic polynomial is then used to fit the vehicle's velocity and acceleration. The vehicle's pose, velocity, acceleration, and steering angle are considered as the initial values for the optimization variables in the optimal control problem. While the hybrid A* algorithm merges vehicle states occupying the same grid in discrete space, it cannot guarantee finding the solution with the minimum cost. However, the solution obtained by the hybrid A* algorithm is often in the vicinity of the global optimum. Therefore, a linear interpolation method is used to densify the trajectory, and then NMPC is applied to refine the trajectory, approaching the global optimum. The process is illustrated in the following Fig. 2:

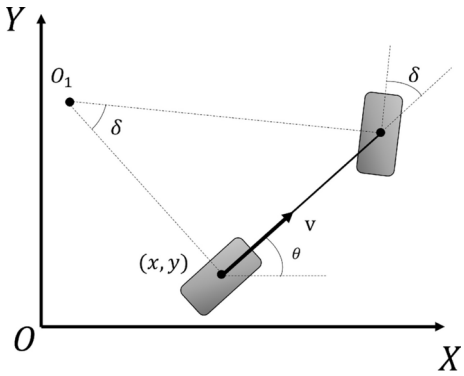


Fig. 1. Schematic diagram of the kinematic model.

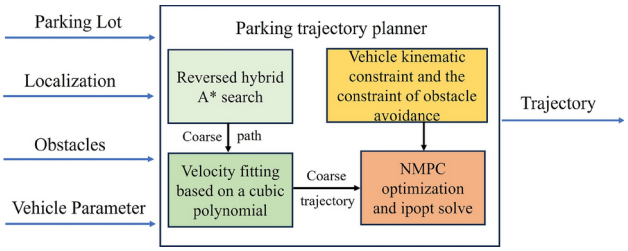


Fig. 2. Automatic parking trajectory planning strategy diagram.

Coarse trajectory planning based on hybrid A* algorithm

Compared to traditional A* algorithm, the hybrid A* algorithm takes into account the vehicle's kinematic constraints in node expansion and the planning result better adheres to the vehicle's motion characteristics. It has been widely used for path planning in open spaces.

Hybrid A path search*

The hybrid A* algorithm considers the vehicle's maximum steering angle constraint during node expansion. The vehicle's maximum steering angle is uniformly discretized into 7 steering angles, and the expansion directions are divided into forward and backward. Therefore, each parent node generates 14 child nodes. When the search step size is particularly small, it is possible to assume a constant heading angle during node expansion. To ensure that the child nodes are not in the same grid as the parent node, the search step size d_{size} is set to be $\sqrt{2}$ times the grid resolution.

In addition, each node in the search process contains some data for the A* algorithm, which can be defined by the following equation:

$$node = \left(X, \tilde{X}, np, g, h\right) \tag{2}$$

where $X = (x, y, \theta)$ represents the vehicle pose, x is the abscissa, y is the ordinate, θ is the heading angle; $\tilde{X}$ indicates the index number of the vehicle pose in the grid map; np is a pointer to the parent node; g is the sum of the costs between all nodes from the starting point to the current point, including the distance cost, direction conversion cost, front wheel angle change cost, and obstacle distance cost; h is the heuristic cost.

Assuming that the cost of the parent node is g_{pre}, the cost of the current node g is calculated by the following equation:

$$\begin{cases} g = g_{pre} + cost_1 + cost_2 + cost_3 \\ cost_1 = \omega_1 * d_{size} \\ cost_2 = \omega_2 * |\Delta steer| \\ cost_3 = \omega_3 * max\left(f\left(d_1\right), f\left(d_2\right) \ldots f\left(d_N\right)\right) \end{cases} \tag{3}$$

$$f\left(d_i\right) = \begin{cases} 0, & d_i \geq d_0 \\ \frac{\varepsilon}{\varepsilon + d_N}, & d_i < d_0 \end{cases} \tag{4}$$

where $\omega_1, \omega_2, \omega_3$ are the corresponding weight coefficients; $cost_1$ is the travel distance cost, d_{size} is the search step size; $cost_2$ denotes the cost of steering angle change, which is introduced to avoid frequent steering adjustments that can affect the smoothness and comfort of the trajectory, the $\Delta steer$ is the change in steer angle; $cost_3$ is the penalty for ego vehicle being close to obstacles. d_i is the distance between the node and the i-th obstacle. For the i-th obstacle, the penalty term can be described by Eq. (4), where d_0 represents the obstacle distance threshold and ε is the attenuation coefficient for the obstacle cost. If the current node's vehicle bounding box collides with an obstacle, the node is discarded.

The heuristic cost is calculated based on the grid map, as shown in Eq. (5).

$$h = max\left(h_1, h_2\right) \tag{5}$$

where h_1 represents the heuristic cost considering only the nonholonomic constraints of the vehicle, calculated by the length of the RS curve; h_2 denotes the heuristic cost considering only the path length, calculated using the Dijkstra algorithm.

The pseudocode for the hybrid A* algorithm is shown in the Table 2 below.

Input: cost gridmap, *node_start* , *node_end*

Init $open\_list = \varnothing, close\_list = \varnothing$
$open\_list \leftarrow close\_list$
while $open\_list \neq \varnothing$
  $node\_current$ = minimum $f\_\cos t$ in $open\_list$
  $open\_list \leftarrow close\_list$
  find a RS curve from $node\_current$ to $node\_end$
  if the RS curve is no collision
    break
  end if
  generate expend nodes $node\_list$
  for node: $node\_list$
    If is collision with obstacles
      continue
    end if
    if is beyond grid map boundaries
      continue
    end if
      if node $\in close\_list$
      continue
    else if node $\in open\_list$
      update the node cost if the cost is lower
    else
      $open\_list \leftarrow node$
    end if
  end for
end while

Output: path point set

Table 2. Pseudocode for the hybrid A* algorithm.

Speed planning based on cubic polynomial
A cubic polynomial is used to roughly fit the time-distance curve. The rough path is divided into several segments based on forward and backward movements. The variations of distance, velocity, and acceleration over time are described by the following equations:

$$\begin{cases} s(t) = bt^3 + ct^2 + dt + e \\ v(t) = 3bt^2 + 2ct + d \\ a(t) = 6bt + 2c \end{cases}, \forall t \in [0, t_f] \tag{6}$$

where *b*, *c*, *d*, and *e* are the coefficients of the distance polynomial; *t* is the current time; *s(t)*, *v(t)*, *a(t)* respectively represent the distance traveled, velocity, and acceleration of the vehicle at time *t*; t_f is the total time required for this path segment. The equality constraints satisfied by Eq. (6) are shown in the following expression:

$$\begin{cases} s(0) = 0 \\ v(0) = 0 \\ s(t_f) = S \\ v(t_f) = 0 \\ a(t_{vmax}) = 0 \\ v(t_{vmax}) = v_{max} \end{cases} \tag{7}$$

where S represents the total length of this path segment; $v_{\max}$ is the maximum speed; $t_{v\,\max}$ indicates the time corresponding to the maximum speed.

According to Eqs. (6) and (7), the velocity and acceleration can be assigned to the path points, resulting in an initial solution for numerical optimization.

Trajectory optimization based on optimal control

Although the trajectory obtained through hybrid A* direct search is drivable, the path formed by node expansion and the RS curve consists of arc and straight line, resulting in discontinuous curvature. Therefore, the optimal control method is used to optimize the initial trajectory.

State space equation of the optimal control problem

To ensure the continuity of the optimized trajectory and satisfy the kinematic constraints, the vehicle's motion state must adhere to the following state space equation:

$$\dot{Z}(t) = f(Z(t), u(t)) \tag{8}$$

where Z is the state variable, $Z = (x, y, \theta, v)$; u is the control variable, $u = (\delta, a)$.

The discretized state space equation can be described by the following equation:

$$\begin{bmatrix} x(t+1) \\ y(t+1) \\ \theta(t+1) \\ v(t+1) \end{bmatrix} = \begin{bmatrix} cos(\theta)\, v \\ sin(\theta)\, v \\ tan(\delta)\, v/L \\ a \end{bmatrix} dt + \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \\ v(t) \end{bmatrix} \tag{9}$$

Objective function of the optimal control problem

The trajectory is discretized into a series of trajectory points according to time, which can be expressed as $V = (x_{ti}, y_{ti}, \theta_{ti}, v_{ti}, \delta_{ti}, a_{ti})$, $ti \in [0, T]$. The optimization is performed on these trajectory points. The objective cost function is composed of parking space occupancy cost, control variables cost, control variables change cost, and parking trajectory length cost, as described by the Eq. (10):

$$\begin{cases} \min J(V, t) = J_1 + J_2 + J_3 + J_4 \\ J_1 = w_1 \sum_{i=0}^{T} (x_{ti}^2 + y_{ti}^2) \\ J_2 = \sum_{i=0}^{T-1} (w_{21} a_{ti}^2 + w_{22} \delta_{ti}^2) \\ J_3 = \sum_{i=0}^{T-1} (w_{31} (a_{ti+1} - a_{ti})^2 + w_{32} (\delta_{ti+1} - \delta_{ti})^2) \\ J_4 = w_4 \sum_{i=0}^{T} (|v_{ti}| * dt) \end{cases} \tag{10}$$

where the J is the total cost; J_1 is the space occupation cost;J_2 is control variables cost; J_3 represents the control variables change cost; J_4 represents the parking trajectory length cost;$w_1, w_{21}, w_{22}, w_{31}, w_{32}, w_4$ are the corresponding weights; $x_{ti}, y_{ti}, \theta_{ti}, v_{ti}, \delta_{ti}, a_{ti}$ respectively represents the abscissa, ordinate, heading angle, speed, steering angle and acceleration of the i-th discrete trajectory point; T is the number of the discrete trajectory points.

Constraints of the optimal control problem

The constraints of proposed optimal control method are composed of three parts.

The first part is limit constraints for v, δ, and a, considering speed limitation in parking scenario and vehicle kinematics, as shown in Eq. (11):

$$\begin{cases} |v(t)| \le v_{max} \\ |\delta(t)| \le \delta_{max} \\ |a(t)| \le a_{max} \end{cases} ,\forall t \in [0, t_{f\_all}] \tag{11}$$

where $\delta_{\max}$ is the max steering angle, $a_{\max}$ is the max acceleration, t_{f_all} is the total parking time.

The second part is the pose constraints and velocity constraints when the vehicle is at the start points and end points for parking, as depicted in Eq. (12):

$$\begin{cases} x_0 = x_{start} \\ y_0 = y_{start} \\ \theta_0 = \theta_{start} \\ v_0 = v_{start} \\ x_T = x_{end} \\ y_T = y_{end} \\ \theta_T = \theta_{end} \\ v_T = v_{end} \end{cases} \tag{12}$$

where $[x_{start}, y_{start}, \theta_{start}]$ is the starting pose; $[x_{end}, y_{end}, \theta_{end}]$ is the final pose; v_{start} is the starting velocity and v_{end} is the final velocity.

As collisions with obstacles must be avoided during vehicle movement, the collision-free constraints should be considered as the third part of constraints, which are obtained by the following method.

Obstacle collision detection typically utilizes bounding boxes to describe the contours of the vehicle and obstacles. Common types of bounding boxes include axis-aligned bounding boxes (AABB), oriented bounding boxes (OBB), and sphere bounding boxes. AABB does not change the direction of the bounding box with the heading, OBB employs directed bounding boxes to better describe the original shape of objects, and sphere bounding boxes envelop the vehicle using multiple circles. The specific forms of these bounding boxes are illustrated in Fig. 3[32]. For parking scenarios, the environment is usually narrow and complex. Therefore, OBB are adopted to describe the obstacle contours for automatic parking.

The ego vehicle and obstacle outlines are described as rectangles using OBB. A simple and effective method for detecting collisions between rectangles is employed by projecting them into each other.

When there is no collision between the ego vehicle and the obstacle, the four corners of the obstacle are outside the rectangle of the ego vehicle, and the four corners of the ego vehicle are also outside the rectangle of the obstacle. Therefore, the collision-free constraint with obstacles can be transformed into constraints where points are outside rectangles. For one obstacle in a state of the ego vehicle, such constraints amount to 8. The constraint for points outside a rectangle can be described by the following Eq. (13):

$$(|x| - L_{half}) + (|y| - W_{half}) + ||x| - L_{half}| + ||y| - W_{half}| > d_{safe} \tag{13}$$

where W_{half} is half of the width of the vehicle, and L_{half} is half of the length of the vehicle.

Figure 4 illustrates the above method by means of determining if a point is outside a rectangle. As shown in Fig. 4a, the obstacle is projected into the ego vehicle's coordinate system, this corner of the obstacle is outside the rectangle of the ego vehicle when Δx or Δy is greater than 0. When the ego vehicle is projected into the obstacle's coordinate system, as shown in Fig. 4b, this corner of the ego vehicle is outside the rectangle of the obstacle while Δx or Δy is greater than 0. If all corners meet the above condition, there is no collision between the ego vehicle and the obstacle. In order to improve the safety, the safe distance without collision is defined as a positive number which is greater than 0.

When the obstacle is projected into the ego vehicle's coordinate system, the rectangular coordinate system is established based on the ego vehicle's pose. The transformation matrix is as follows:

$$\begin{bmatrix} x_{obs\_in\_ego} \\ y_{obs\_in\_ego} \end{bmatrix} = \begin{bmatrix} cos\,(\theta_i) & -sin\,(\theta_i) \\ sin\,(\theta_i) & cos\,(\theta_i) \end{bmatrix} \begin{bmatrix} x_{par} - x_i \\ y_{par} - y_i \end{bmatrix} \tag{14}$$

$$\theta_{obs\_in\_ego} = \theta_{par} - \theta_i \tag{15}$$

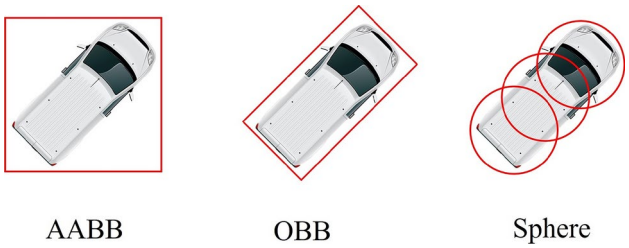
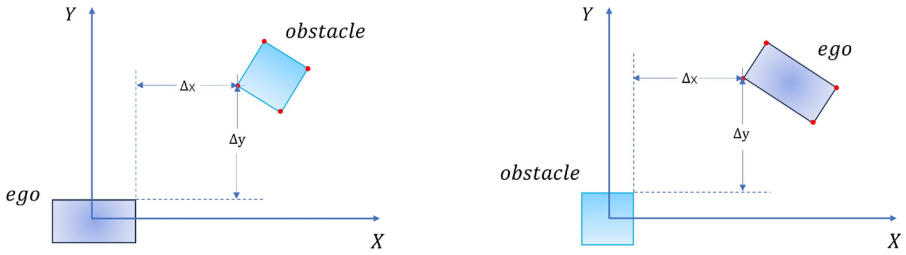


Fig. 3. Common bounding box types.



(a) Projection of obstacles into the ego vehicle

(b) Projection of the ego vehicle into obstacles

Fig. 4. Schematic diagram of collision avoidance.

Parameter	Value	Unit
Vehicle wheelbase L_f	3.309	m
Vehicle length L	5.995	m
Vehicle width W	2.18	m
Vehicle front overhang L_{fo}	1.031	m
Vehicle rear overhang L_{bo}	1.655	m
Max steering angle δ_{max}	0.44157	rad
Max acceleration/deceleration speed	2	m/s^2

Table 3. Main kinematic parameters of the vehicle.

where $x_{obs\_in\_ego}$, $y_{obs\_in\_ego}$ and $\theta_{obs\_in\_ego}$ are the abscissa, ordinate and heading of the obstacle in the ego vehicle coordinate system; x_{par}, y_{par} and θ_{par} are the abscissa, ordinate and heading of the obstacle in the parking coordinate system; x_i, y_i, and θ_i are current abscissa, ordinate and heading of the ego vehicle in the parking coordinate system.

Numerical simulation and discussion

The program is implemented in C++ and utilizes the IPOPT library to solve the nonlinear optimization problem. The vehicle parameters are set as shown in Table 3.

In practical scenarios, parking spaces often do not have sufficient length to allow a vehicle to park in one attempt. For parallel parking, in extremely narrow parking spaces, multiple maneuvers may be required to complete the parking. Similarly, for vertical parking, in situations with small road boundaries, multiple adjustments may be necessary at the parking entrance to successfully park the vehicle.

In the case of meeting the calculation time requirements, it is desirable to keep the grid size and search step size as small as possible in the hybrid A* search, as well as expanding more child nodes based on steering angle for avoiding search failure in narrow space. Furthermore, reverse search is utilized to accelerate the search process and help the vehicle swiftly navigate out of narrow parking slot.

Considering both the success rate and computational time in narrow parking scenarios, the proposed algorithm's main parameters are defined as Table 4.

Parallel parking simulation

The parallel parking space has a length of 8.0 m and a width of 3.5 m. The comparision of hybrid A* path and the NMPC optimized path are shown in Fig. 5a. Figure 5b shows the space occupied by the vehicle in the NMPC optimized path. Figure 6 illustrates the motion state of the vehicle in the NMPC optimized path and displays the variations of heading angle, velocity, acceleration, and front wheel angle over time. The planning process takes a total of 2.867 s.

Vertical parking simulation

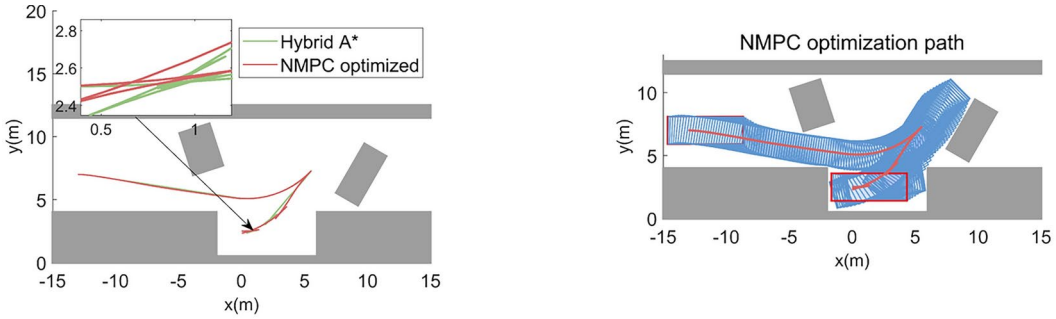
The vertical parking space is the same as the parallel parking space, which has a length of 8.0 m and a width of 3.5 m. Figure 7a displays the comparison of hybrid A* path and NMPC optimized path, while Fig. 7b illustrates the space occupied by the vehicle with NMPC optimized path. The heading angle, velocity, acceleration and front wheel angle of the vehicle for vertical parking are presented in Fig. 8. The planning process lasts 2.604 s.

Algorithm comparison and discussion

It can be observed from above simulation experiments that in parallel parking scenarios, the pure hybrid A* algorithm requires 13 gear shifts with a path length of 34.08 m, while the proposed hybrid A* with NMPC algorithm only requires 5 gear shifts with a path length of 29.06 m. The proposed algorithm not only significantly

Parameter	Value	Unit
Hybrid A* Grid XY resolution	0.2	m
Hybrid A* grid heading resolution	0.1	rad
Hybrid A* search step size	0.2	m
Maximum parking speed v_{max}	1	m/s
next-node expansion number	14	
Hybrid A* cost weights ω_1	20	
Hybrid A* cost weights ω_2	40	
Hybrid A* cost weights ω_3	10	
NMPC cost weight w_1	200	
NMPC cost weight w_{21}	100	
NMPC cost weight w_{22}	100	
NMPC cost weight w_{31}	100	
NMPC cost weight w_{32}	100	
NMPC cost weight w_4	400	
Obstacle envelope expansion	0.1	m

Table 4. Main parameters of the proposed algorithm.



(a) Comparison of planned paths for parallel parking **(b)** NMPC optimization path for parallel parking

Fig. 5. Parallel parking path.

reduces the number of gear shifts in complex scenarios but also shortens the trajectory length by 14.73%. In vertical parking scenarios, the hybrid A* algorithm requires 17 gear shifts with a path length of 32.51 m, whereas the hybrid A* with NMPC algorithm only requires 3 gear shifts with a path length of 25.89 m, similarly reducing the number of gear shifts in complex scenarios and shortening the trajectory length by 20.36%.

Therefore, the hybrid A* with NMPC algorithm exhibits better performance in complex narrow parking scenarios compared to the pure hybrid A* algorithm. It not only generates a shorter path but also optimally utilizes space. The unnecessary gear shifting behavior is avoided through adjusting the vehicle pose in advance. Additionally, the planning process is within 3 s, meeting the time requirements for full parking scenarios.

Parking scenes are generally stationary, so only static obstacles is considered in this study. When dynamic obstacles are incorporated into the proposed algorithm, a prediction module is required to provide accurate trajectories for these dynamic obstacles, and real-time planning is conducted at a fixed frame rate. However, in real driving scenario, precise trajectories of dynamic obstacles cannot be acquired, the planning process should be conducted at each frame, thus leading to huge calculation amount of proposed algorithm. Due to limited computing power of on-board controller, feasible trajectories cannot be obtained by proposed algorithm when considering dynamic obstacles. Based on above analysis, the dynamic obstacles are not included in the proposed algorithm.

The proposed algorithm describes the obstacles based on rectangular bounding boxes. The accuracy of bounding boxes is affected by accurate perception of the environment. In real driving scenarios, when there exists sensor noise or occlusions in the environmental perception, the bounding boxes acquired by proposed algorithm may differ from actual bounding boxes through actual obstacles. This will result in collision risks. Therefore, the reliability of proposed algorithm can be improved further by enhancing the environment perception accuracy.

Algorithm validation on actual vehicles

To validate the algorithm's practicability in real vehicle, real vehicle experiments are conducted using the EV18 platform. The experimental vehicle is equipped with a drive-by-wire chassis, a domain controller, a laser radar, millimeter-wave radar, surround-view cameras, and an integrated inertial navigation system, as shown in Fig. 9.

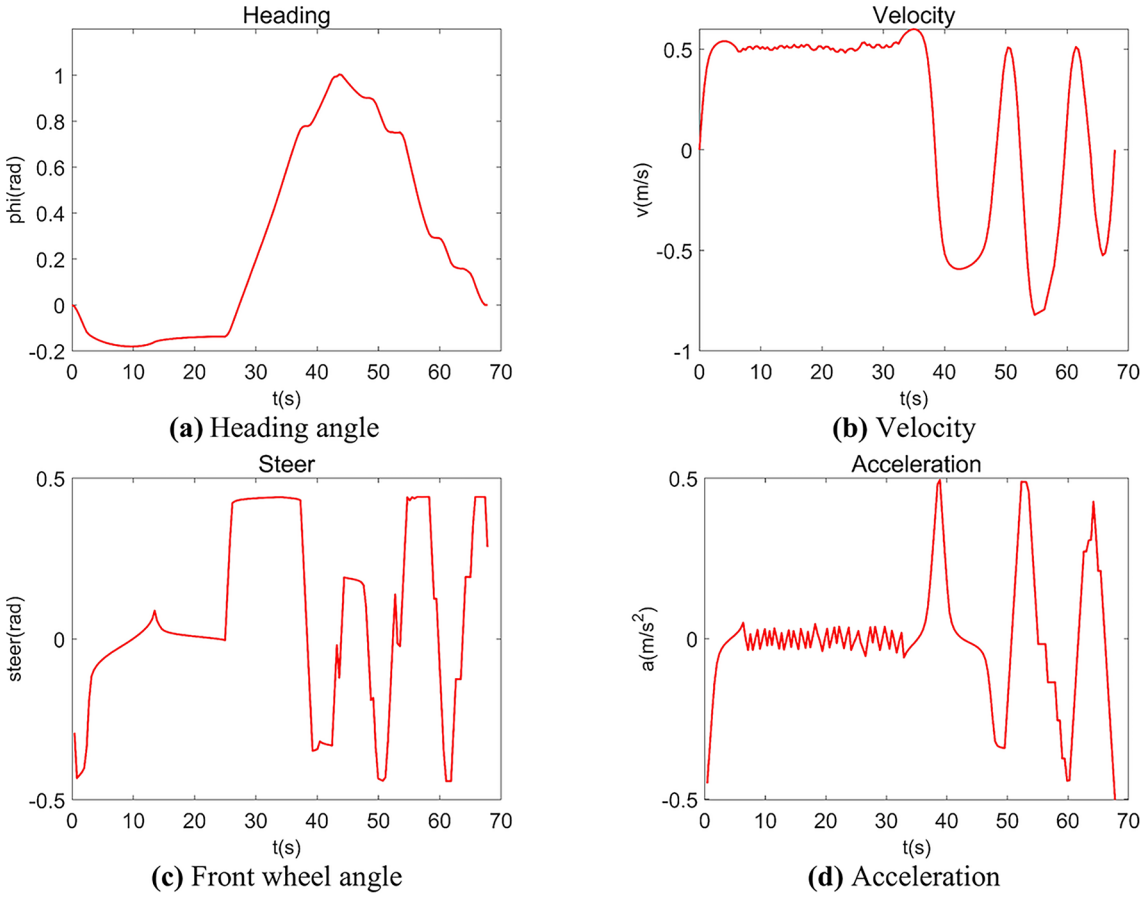


Fig. 6. Variation of heading angle, velocity, front wheel angle and acceleration over time for parallel parking.

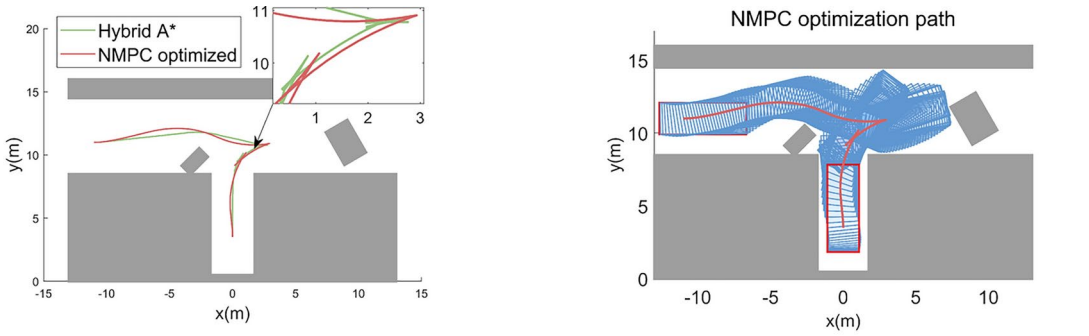


Fig. 7. Vertical parking path.

The software architecture of the vehicle's autonomous driving system can be represented as shown in Fig. 10. The bottom layer is the hardware layer, which includes various sensors, actuators, V2X devices, and other hardware components. The middle layer consists of the firmware, which utilizes the MCU, SOC, UDP, CAN, and various interfaces to communicate and process data from the hardware layer. Based on Ubuntu 20.04 and the Robot Operating System (ROS), the top layer which is called the application layer, include map, localization, perception, planning and control modules.

Due to the slow driving speed and low number of traffic participants, all obstacles are treated as static obstacles. In the application layer, the planning module receives upstream information including perception, localization, chassis messages, and parking space information. Then the planning module sends the trajectory point sequence and gear information to the control module at a frequency of 10 Hz. The control module utilizes MPC to track the trajectory, and control command are sent to the chassis at a frequency of 100 Hz.

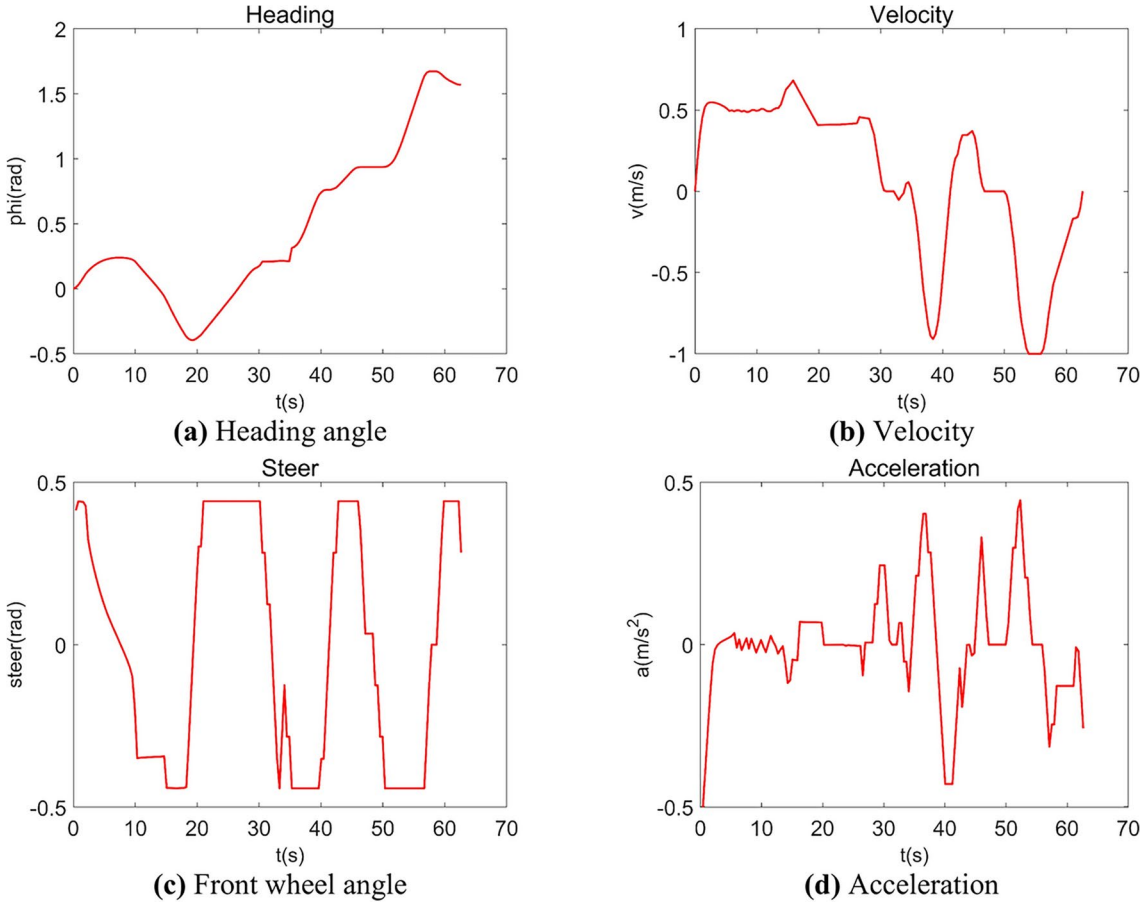


Fig. 8. Heading angle, velocity, front wheel angle and acceleration of the vehicle for vertical parking.

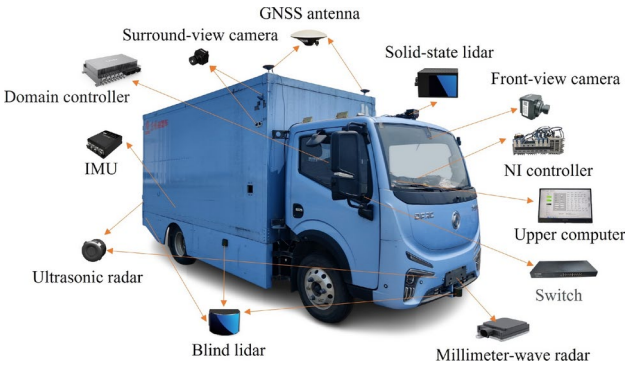


Fig. 9. EV18 experimental platform.

The kinematic parameters of the vehicle and the main algorithm parameters remain consistent with Tables 3 and 4. The size of the parallel parking spot and vertical parking spot remains 8.0 m in length and 3.5 m in width. All obstacles are described using OBB rectangular envelopes. The boundaries of parking spots and road boundaries can also be represented using rectangular boxes of zero length or width. The safety distance is determined by the inflation distance of the obstacle envelope as specified in Table 3. The vehicle experiment is conducted on autonomous driving test road within the Wuhan Economic Development Zone.

Parallel parking experiment

In the parallel parking scenario, the planned trajectory and the actual vehicle trajectory are shown in Fig. 11. The vehicle first moves forward to approach the parking space, then reverses into the parking space, and finally adjusts its position by changing gears twice within the parking spot to complete parallel parking.

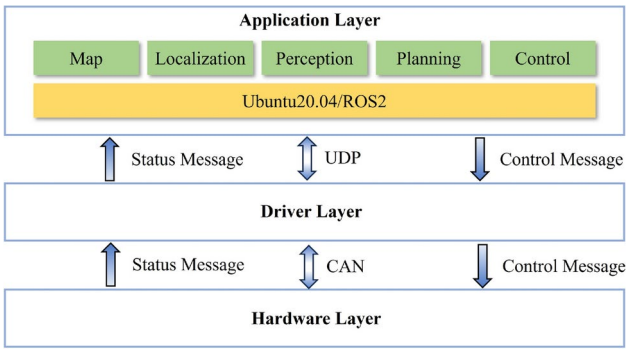


Fig. 10. Vehicle software architecture diagram.

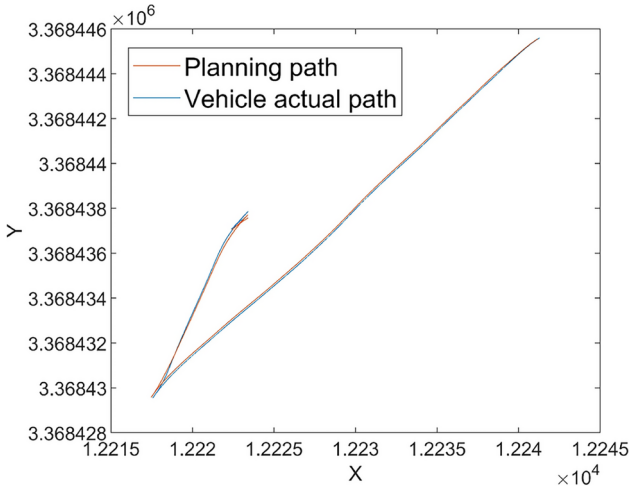


Fig. 11. Parallel parking planning trajectory and actual trajectory.

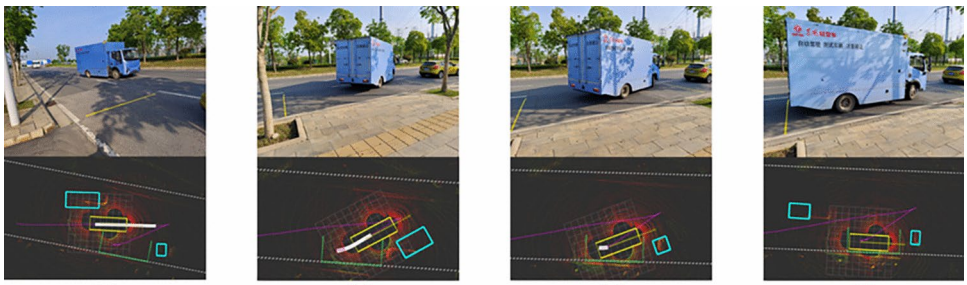


Fig. 12. Parallel parking monitoring diagram.

Figure 12 shows the real-time monitoring visualization of parallel parking using the ROS-based RVIZ tool. As depicted in Fig. 12, the ego vehicle's heading and outline are denoted by yellow arrows and a rectangular bounding box respectively, whereas obstacles are depicted by light blue rectangular bounding boxes. The trajectory of the rear axle center point throughout the parking process is depicted by the purple curve, while the white curve represents the real-time trajectory sent by the planning module. The white dashed lines are road boundaries.

The experimental results indicate that for narrow parallel parking spaces, the vehicle can still plan reasonable trajectories with minimal gear shifts, effectively executing the parallel parking maneuver. successfully accomplishing the parallel parking maneuver. The vehicle first moves to the front of the parking space and starts reversing, then adjusts its position through two gear shifts in the parking space. The planning duration amounts to 2.412 s, and the trajectory length is 42.52 m.

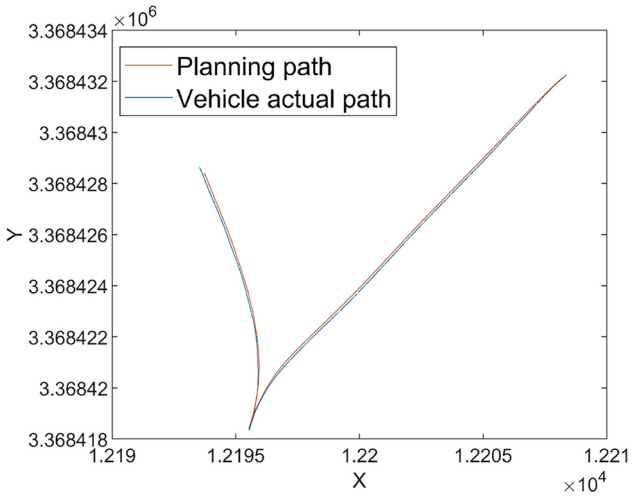


Fig. 13. Vertical parking planning trajectory and actual trajectory.

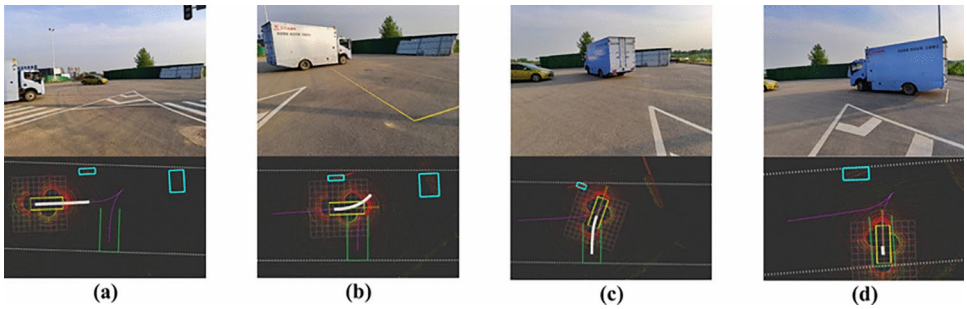


Fig. 14. Vertical parking monitoring diagram.

Vertical parking experiment

In the vertical parking scenario, the planned trajectory and the actual vehicle trajectory are shown in Fig. 13. The vehicle first moves forward and adjusts its heading angle and then returns to the vertical parking spot. Throughout the process, there are no collisions with obstacles or parking spot. Figure 14 displays the real-time monitoring visualization of the actual vehicle's operation process using RVIZ.

For vertical parking scenarios with obstacles and limited road boundaries, the proposed algorithm can plan excellent trajectories. The vehicle first drives to the obliquely above the parking space and adjusts its rear direction, then reverses to complete the vertical parking maneuver. The duration of the planning process is 1.967 s, and the trajectory spans a length of 30.48 m.

It can be seen from Figs. 11, 12, 13 and 14 that the proposed algorithm can effectively complete the parking task in both parallel and vertical parking.

Conclusions

To enable parallel and perpendicular parking in narrow spaces, this study proposes an improved trajectory planning method composed of hybrid A* and NMPC. The proposed algorithm initially utilizes the hybrid A* algorithm and backward search for initial path planning. Subsequently, speed planning is carried out based on a cubic polynomial. Vehicle kinematic constraints and precise collision avoidance constraints are then integrated into the environment. The NMPC with longitudinal and lateral coupling is employed to further optimize the trajectory. In parallel parking simulations, the path length is reduced by 14.73%, with gear shifts decreasing from 13 to 5. In vertical parking, the path length is reduced by 20.36%, with gear shifts decreasing from 17 to 3. The simulation and real-world experiments results demonstrate that proposed algorithm performs well in various scenarios, exhibiting strong adaptability, smooth trajectories and good computation efficiency within 3 s.

Based on precise obstacle contours described by OBB bounding box, the proposed algorithm can make full use of the parking space by establishing the collision-free constraint. Furthermore, proposed hybrid A* combining NMPC effectively reduces path length and gear shift frequency. Therefore, proposed algorithm can enable successful planning of reasonable and smooth trajectories during automatic parking planning in narrow spaces, as well as improving efficiency and comfort of automatic parking. However, on one hand, proposed algorithm does not consider dynamic obstacles, thus there is a risk of collision with dynamic obstacles under complex and variable driving environment; on the other hand, perception errors caused by sensor noise and occlusions is

not involved in proposed algorithm, hence obstacle envelope may fluctuate in real driving environment, there is also a risk of collision when the vehicle is very close to the obstacle. For future research, dynamic obstacles and perception errors caused by sensor noise should be incorporated into the trajectory planning, in order to improve the reliability of trajectory planning in real driving environment.

Data availability

The datasets generated and/or analysed during the current study are not publicly available due [Project confidentiality agreement] but are available from the corresponding author on reasonable request.

Received: 3 July 2024; Accepted: 3 January 2025
Published online: 09 January 2025

References

1. Jiang, B. & Fan, Z. P. Optimal allocation of shared parking slots considering parking unpunctuality under a platform-based management approach. *Transp. Res. E. Logist Transp. Rev.* **142**, 1–16 (2020).
2. Shafiei, S., Gu, Z., Grzybowska, H. & Cai, C. Impact of self-parking autonomous vehicles on urban traffic congestion. *Transportation* **50**, 183–203 (2023).
3. Li, S. & Wang, J. Parallel parking path planning in narrow space based on a Three-stage Curve interpolation method. *IEEE Access* **11**, 93841–93851 (2023).
4. Kim, D. J. & Chung, C. C. Automated perpendicular parking system with approximated clothoid-based local path planning. *IEEE Contr. Syst. Lett.* **5**, 1940–1945 (2020).
5. Ghajar, M., Alirezaei, M., Besselink, I. & Nijmeijer, H. A fast analytical local path planning method with applications in parking scenarios. *Proc. Inst. Mech. Eng. D. J. Automob. Eng.* **238**, 2012–2026 (2023).
6. Huang, J., Yang, Y., Ding, D., Li, Y. & He, Y. Automatic parking paths planning research based on scattering points six-degree polynomial and easement curve. *Proc. Inst. Mech. Eng. D. J. Automob. Eng.* **237**, 529–543 (2023).
7. Song, J. et al. Laser-based SLAM automatic parallel parking path planning and tracking for passenger vehicle. *IET Intell. Transp. Syst.* **13**, 1557–1568 (2019).
8. Piao, C., Zhang, J., Chang, K. H., Li, Y. & Liu, M. Multi-sensor information ensemble-based automatic parking system for vehicle parallel/nonparallel initial state. *Sensors* **21**, 1–19 (2021).
9. Zhang, B., Li, Z., Ni, Y. & Li, Y. Research on path planning and tracking control of automatic parking system. *World Electr. Veh. J.* **13**, 1–13 (2022).
10. Cai, L. et al. Parking planning under limited parking corridor space. *IEEE Trans. Intell. Transp. Syst.* **24**, 1962–1981 (2022).
11. Chen, Q. et al. Parallel parking path planning based on improved arctangent function optimization. *Int. J. Automot. Technol.* **24**, 23–33 (2023).
12. Dong, Y., Zhong, Y. & Hong, J. Knowledge-biased sampling-based path planning for automated vehicles parking. *IEEE Access* **8**, 156818–156827 (2020).
13. Wang, X., Wei, J., Zhou, X., Xia, Z. & Gu, X. AEB-RRT*: an adaptive extension bidirectional RRT* algorithm. *Auton. Robot* **46**, 685–704 (2022).
14. Gan, Y. et al. Research on robot motion planning based on RRT algorithm with nonholonomic constraints. *Neural. Process. Lett.* **53**, 3011–3029 (2021).
15. Sedighi, S., Nguyen, D. V. & Kuhnert, K. D. Guided hybrid A-star path planning algorithm for valet parking applications. *Int. Conf. Control, Autom. Robot.* 570–575(2019).
16. Li, C., Du, J., Liu, B. & Li, J. A path planning method based on hybrid a-star and RS algorithm. *SAE Tech. Pap.* 1–8 (2020).
17. Zhang, Y., Chen, G., Hu, H. & Gao, Z. Hierarchical parking path planning based on optimal parking positions. *Automot. Innov.* **6**, 220–230 (2023).
18. Meng, T. et al. Improved hybrid A-star algorithm for path planning in autonomous parking system based on multi-stage dynamic optimization. *Int. J. Automot. Technol.* **24**, 459–468 (2023).
19. He, J. & Li, H. Fast A* anchor point based path planning for narrow space parking. *IEEE Conf. Intell. Transport. Syst. Proc. ITSC* 1604–1609(2021).
20. Zhang, X., Liniger, A. & Borrelli, F. Optimization-based collision avoidance. *Trans. Contr. Syst. Technol.* **29**, 972–983 (2020).
21. Ma, J. et al. Local learning enabled iterative linear quadratic regulator for constrained trajectory planning. *IEEE Trans. Neural Netw. Learn. Syst.* **34**, 5354–5365 (2022).
22. Duan, S. *et al.* DSMPC-DCBF: A hierarchical parking trajectory optimization method based on MPC. *ACIRS.* 55–62 (2024).
23. Zhou, R., Liu, X. & Cai, G. A new geometry-based secondary path planning for automatic parking. *Int. J. Adv. Robot. Syst.* **17**, 1–17 (2020).
24. Qiu, D., Cheng, M., Yang, H., Dai, Y. & Wang, Y. Time-optimal trajectory planning for autonomous parking of tractor-semitrailer vehicle based on piecewise Gauss pseudospectral method. *Adv. Mech. Eng.* **15**, 1–17 (2023).
25. Daniali, S. M., Khosravi, A., Sarhadi, P. & Tavakkoli, F. An automatic parking algorithm design using multi-objective particle swarm optimization. *IEEE Access* **11**, 49611–49624 (2023).
26. Yu, L. et al. Parallel parking path planning and tracking control based on simulated annealing algorithm. *Int. J. Automot. Technol.* **22**, 949–965 (2024).
27. Su, B., Yang, J., Li, L. & Wang, Y. Secondary parallel automatic parking of endpoint regionalization based on genetic algorithm. *Cluster Comput.* **22**, 7515–7523 (2019).
28. Han, G. L. Automatic parking path planning based on ant colony optimization and the grid method. *J. Sens.* **2021**, 1–10 (2021).
29. Tang, X. et al. Path planning and tracking control for parking via soft actor-critic under non-ideal scenarios. *IEEE/CAA J. Autom. Sin.* **11**, 181–195 (2023).
30. Zhang, P. et al. Reinforcement learning-based end-to-end parking for automatic parking system. *Sensors* **19**, 1–24 (2019).
31. Zhang, J., Chen, H., Song, S. & Hu, F. Reinforcement learning-based motion planning for automatic parking system. *IEEE Access* **8**, 154485–154501 (2020).
32. Zhu, H. & Luo, Q. Indoor localization of mobile robots based on the fusion of an improved AMCL algorithm and a collision algorithm. *IEEE Access* **12**, 67199–67208 (2024).

Acknowledgements

This work was supported by Major Science and Technology Projects of Hubei Province under grant 2022AAA001 and in part by Major research projects of Hubei Province under grant 2023BAA017.

Author contributions

P.Z.: Review, editing, methodology; S.Z.: Software, analysis, writing; J.H.: Supervision, resources, data; W.Z. and J.Z.: Expertise, editing suggestions; Z.Z.: Experimentation, visualization; C.G.: Project management, experimental platform.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-85541-x.

Correspondence and requests for materials should be addressed to J.H.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.