A guaranteed collision-free trajectory planning method for autonomous parking

Zhiming Zhang\(^1\) | Shan Lu\(^2\) | Lei Xie\(^1\) | Hongye Su\(^1\) | Dongliu Li\(^3\) | Qibing Wang\(^3\)

Weihua Xu\(^1\)

\(^1\) State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou, China
\(^2\) Institute of Intelligence Science and Engineering, Shenzhen Polytechnic, Shenzhen, China
\(^3\) Sicher Elevator Co, Ltd, Huzhou, China

Abstract
Planning a feasible, and safe trajectory is a crucial procedure for autonomous parking which remains to be fully solved. Generally, a trajectory is depicted by a sequence of configurations which include positions, and orientations. However, existing approaches usually design parking trajectories without considering the evolution from one configuration to the next. Hence, these trajectories are practically infeasible in some scenarios, and safety risks may exist. In this paper, a guaranteed collision-free trajectory planning method is proposed for autonomous parking. A dynamic optimisation problem is built, which includes vehicle kinematics, collision avoidance constraints, and physical restraints. Then the dynamic optimisation problem is discretised into a finite-dimensional nonlinear programming problem. To describe collision avoidance constraints, the concepts of the virtual protection frame, and the magnification parameter are introduced. It is highlighted that a novel collision detection criterion is raised from the view of geometry. Tests in three common scenarios, and a complex scenario illustrate the effectiveness of the proposed approach. Moreover, the stability of the algorithm is discussed in this paper. Via the proposed approach, an integrated design of planning, and control is possible.

1 | INTRODUCTION
Nowadays, fully autonomous driving is still a long way off but many manufacturers have promoted cars equipped with parking systems, such as BMW, Ford, Toyota etc. Imagine that a driver gets off at the garage entrance and goes to his office directly, then the vehicle drives itself from the drop-off point to the parking spot. Autonomous parking can emancipate drivers from the last driving stage.

Besides these commercial applications, the number of literature on autonomous parking has been increasing steadily. Reference [1] divides the autonomous parking strategies into two categories, the path planning approach and the skill-based approach. The latter tries to park by imitating an experienced human driver, for example, neural network control and fuzzy control [2–4]. Since the recent two decades, the path planning approach has received a lot of attention because skill-based algorithms are computationally expensive and difficult to implement [5]. The trajectory (or path) planning problem for autonomous parking can be shortly defined as: given the initial and final configurations, find a feasible curve that an autonomous vehicle can move along. In theory, there may exist a lot of feasible trajectories from the initial configuration to the final configuration. An optimal trajectory is usually defined with the shortest length or shortest time.

As stated in [6, 7], trajectory planning approaches are classified into four types. In graph search-based methods such as Dijkstra's algorithm, A* algorithm, State Lattice algorithm, the states of a vehicle are stored in the grids, then a solution is searched in the whole grids [8–10]. Sampling-based planners like RRT [11] can quickly explore the space since they sample the configuration space or the state space randomly at the price of suboptimal solution. Interpolating curve planners are widely used for the ability to offer a smoothing path solution. To ensure the high order continuity, polynomial curves [12], Bezier curves [13], spline curves [14] are often adopted.
Compared with the above three types of planning algorithms, numerical optimisation aims to obtain a trajectory by solving a constrained dynamic optimisation problem (CDOP) [15–18]. The CDOP defines the objective function according to the driving mission. Vehicle kinematics, obstacle avoidance and physical limitations are included as constraints. Then the infinite-dimensional function space of the dynamic optimisation problem is projected into a finite-dimensional vector space to leverage existing nonlinear programming (NLP) methods, that is, the problem is discretised in the whole time horizon and then a sequence of configurations are obtained. However, discretisation can lead to improper simplification. Most existing CDOP approaches do not consider the possible constraints violation between two adjacent configurations. In other words, the evolution between two adjacent configurations is omitted when discretisation. In [17], at every configuration point, two auxiliary decision variables are induced to compute the minimum distance between the vehicle and the obstacle by dual-function. In [18], the collision-free is determined if at every configuration point the four vertices of a vehicle are not inside the obstacle. In consequence, the potential collision is still a problem in the CDOP approach for autonomous parking. To address this challenge, a guaranteed collision-free trajectory planning method for autonomous parking is proposed.

The contributions of this paper can be summarised as follows: (i) The concepts of the virtual protection frame and the magnification parameter are introduced to depict the region the vehicle covers along the trajectory. (ii) A collision detection criterion is proposed from the view of geometry. Specifically, the trajectory is collision-free only if there is no intersection between the virtual protection frame and the obstacle. The intersection is determined by the cross product. (iii) An iterative planning algorithm is stated which guarantees the collision-free solution in the whole time horizon.

The remainder of this paper is structured as follows: In Section 2, the formulation of CDOP when parking a four-wheel vehicle is introduced. In Section 3, the CDOP is solved via the nonlinear programming. The method to discretise the CDOP is detailed. Then a guaranteed collision-free trajectory planning algorithm is proposed. Numerical simulation results are shown in Section 4, which is followed by some analysis and comparisons with the conventional method in Section 5. At last, conclusions and future work are stated in Section 6.

## 2 FORMULATION OF THE CONSTRAINED DYNAMIC OPTIMISATION PROBLEM WHEN PARKING A FOUR-WHEEL VEHICLE

In this section, assuming a reliable map is available, the trajectory planning of a vehicle from the initial configuration to the final configuration is formulated as a CDOP, in which the kinematics model of a vehicle, the constraints to avoid collisions, and the physical restrictions are all taken into account.

### 2.1 Kinematics description of a four-wheel vehicle

A four-wheel vehicle can be regarded as a rigid body when parking at low speed [19]. Hence the dynamics of vehicle suspensions and the side-slip of tires are ignored. The kinematic equations [18] of a front-steering vehicle is denoted by Equation (1)

\[
\begin{align*}
    \dot{X} &= v \cdot \cos \phi \\
    \dot{Y} &= v \cdot \sin \phi \\
    \dot{\phi} &= \frac{r \cdot \tan \delta}{L} \\
    \dot{t} &= a
\end{align*}
\]

where \((X, Y)\) is the center position of the rear axle in the global coordinates as shown in Figure 1, and \(\phi\) is the orientation of the vehicle. \(\delta\) is the steering angle of the front wheel. \(v\) and \(a\) stand for the velocity and acceleration at the center of the rear axle. \(L\) represents the wheelbase. Figure 1 also indicates the vehicle width \(w\), four vertices in clockwise order starting from left-front \(E_1, E_2, E_3, E_4\), and an obstacle with five vertices \(V_1, V_2, V_3, V_4, V_5\) in clockwise order.

Summarily, \([a, \delta]\)^T are input variables and \(\xi = [X, Y, \phi, a]^T\) are state variables. Clearly, they are all functions of time. Once the input functions are decided, the states of the vehicle during the whole time horizon can be calculated by ordinary differential equation (ODE) \(\dot{\xi} = f(\xi, a)\) under the initial values.

### 2.2 Collision avoidance from the view of geometry

In this section, a geometrical condition of avoiding collision is discussed. Here, the vehicle and obstacles are modeled as
polygons. For the vehicle in Figure 1, its contour is defined by four vertices:

\[ E_1, E_2, E_3, E_4 = \{E_1E_2, E_2E_3, E_3E_4, E_4E_1\} \triangleq E. \]  

(2)

The contour of the vehicle is formed by connecting four vertices clockwise and thus is a set of four line segments.

Similarly, for the \( j \)th (assuming total \( J \) obstacles, \( j = 1, 2, 3, \ldots, J \)) obstacle, the contour can be described by \( m_j \) vertices:

\[ V_{j_1}, V_{j_2}, V_{j_3}, \ldots, V_{j_{m_j}} = \{V_{j_1}V_{j_2}, V_{j_2}V_{j_3}, \ldots, V_{j_{m_j}}V_{j_1}\} \triangleq O_j. \]  

(3)

From the point of geometry, there is no collision between the vehicle and the obstacle as long as the boundaries of vehicle and obstacles do not intersect at any moment, which is intuitive and can be presented by:

\[ \neg (E \cap O_j), \forall E \in E, \forall O_j \in O_j, j = 1, 2, 3, \ldots, J, \]  

(4)

where \( \cap \) means line segments \( E \) and \( O_j \) get intersected.

Remark 1. The vehicle should be outside the obstacles at the beginning. Then there is no collision as long as Equation (4) holds.

### 2.3 Physical restraints

Limited by the mechanical principle, the variable bounds should be considered. The physical constraints used in this study include:

\[
\begin{align*}
    v_{\text{min}} &\leq v \leq v_{\text{max}} \\
    a_{\text{min}} &\leq a \leq a_{\text{max}} \\
    \dot{a}_{\text{min}} &\leq \dot{a} \leq \dot{a}_{\text{max}} \\
    \delta_{f_{\text{min}}} &\leq \delta_f \leq \delta_{f_{\text{max}}} \\
    \dot{\delta}_{f_{\text{min}}} &\leq \dot{\delta}_f \leq \dot{\delta}_{f_{\text{max}}} 
\end{align*}
\]  

(5)

The front-wheel steering angle and its derivative are bounded to meet the torque requirement in the vertical direction of the wheel. Constraints of speed, acceleration and jerk (the derivative of acceleration) ensure the smoothness and stability for parking. Moreover, the rigid body assumption only holds when the vehicle moves at low speed. In practice, one can define the physical constraints to meet some other requirements.

### 2.4 The whole formulation

Now the overall formulation of CDOP when parking a vehicle should be given. Time consumption \( t_f \) is chosen as an objective function since people usually desire to park their cars as fast as possible. Consequently, for total \( J \) obstacles in the scenario, the optimisation problem can be formulated as Eq(6).

Unfortunately, computers cannot deal with infinite-dimensional problems such as the optimisation problem (6). It is necessary to discretise the problem first, which will be detailed in the next section.

### 3 DISCRETISATION OF THE CONSTRAINED DYNAMIC OPTIMISATION PROBLEM

In this section, the CDOP (6) in Section 2 is transformed into a finite-dimensional nonlinear programming (NLP) problem by discretisation. To discretise the collision avoidance constraints in Equation (4), a novel collision checking method is presented. Finally, a guaranteed collision-free trajectory planning algorithm is proposed with theoretical analysis.

#### 3.1 Multiple-shooting to discretise the vehicle kinematics model

Provided that the whole time horizon is divided into \( N \) intervals equally, on each which the control input is kept constant (see Figure 2). Then the control function \( u() \) is replaced
by $u_1, u_2, \ldots, u_k, \ldots, u_N$. Similarly, $\xi_1, \xi_2, \ldots, \xi_k, \ldots, \xi_N$ are states during each time interval and during each time interval. Given a start state $\xi_k$ and a fixed control input $u_k$, the state for the next interval is calculated by performing a time integration of Equation (1).

Provided the length of each interval is $T$:

$$T = \frac{t_f}{N}.$$  

The integration resorts to the fourth-order Runge–Kutta which has less truncation error:

$$r_1 = f(\xi_k, u_k),$$  

$$r_2 = f\left(\xi_k + \frac{T}{2} \cdot r_1, u_k\right),$$  

$$r_3 = f\left(\xi_k + \frac{T}{2} \cdot r_2, u_k\right),$$  

$$r_4 = f(\xi_k + T \cdot r_3, u_k),$$  

$$\xi_{k+1} = \xi_k + \frac{T}{6} \cdot (r_1 + 2r_2 + 2r_3 + r_4).$$

Note that there is a mismatch between integration outcome $\xi_{k+1}$ and the next state variable $\xi_{k+1}$. Multiple-shooting is adopted to add constraints that handle the mismatch.

$$\xi_{k+1} = \xi_{k+1}.$$  

### 3.2 Discretisation of collision avoidance constraints

Since $t_f$ is divided to $N$ intervals, it is a direct way to discretise collision avoidance constraints by satisfying the constraints just on the $N$ time points. However, a constraint violation may occur at any instant other than the $N$ configurations where the constraints are satisfied, which is unacceptable for vehicle trajectory planning (see Figure 3). When constraint violations happen in some time intervals, new time points are added into these intervals to enhance the resolution, which is called mesh refinement algorithm [20]. For trajectory planning, it is hard to calculate the minimum distance between the vehicle and obstacles [16], which makes it difficult to evaluate the violation level of collision avoidance constraints, and thus a proper refinement strategy is hard to define. Even more time points cannot ensure the non-existent of constraint violation. In addition, as a non-convex problem, the CDOP may convergence to a different local optimum after the mesh refinement, making the result hard to explain.

In this section, a novel method to discretise the collision avoidance constraints is presented, via which collision-free will be satisfied in the whole time horizon. To illustrate this method, the definitions of virtual protection frame (VPF) and magnification parameter are given first.

**Definition 1.** As shown in Figure 4(a), the virtual protection frame is a frame (dashed black) surrounding the vehicle. The length of the VPF equals to the length of the vehicle and the width $w^f$ is proportional to the width of the vehicle, in other words, $\frac{w^f}{w} = \alpha$, where $\alpha \geq 1$ is referred to as magnification parameter.
To consider the evolution between two adjacent configurations, the VPF only extends along the width of the vehicle, instead of the length. In Figure 4(a), four vehicle vertices on the VPF are marked correspondingly. $E_{4|k}$ denotes the position of point $E_{4|k}$ at time point $k$ and $E_{i|k}$ is the position of point $E_i (i = 1, 2, 3, 4)$. $l_i$ and $l_f$ are the distances from the rear axle to the front end and the rear end. In one time interval $[k, k + 1]$, the vehicle trajectory from the configuration at $k$ to the configuration at $k + 1$ can be described as Figure 4(b).

Three important elements should be noticed: (i) The red part is the region where the vehicle passes through, that is, at any time between $[k, k + 1]$, any point on the vehicle is in this region. It is a corollary of vehicle kinematics. (ii) Similar to the red region, the green part is the region where the VPF passes through. (iii) Connect $E_{1|k}^fE_{2|k}^fE_{3|k}^fE_{4|k}^fE_{1|k+1}^f$.

$E_{i|k}^fE_{i+1|k+1}^fE_{i+2|k+1}^fE_{i+3|k+1}^fE_{i+4|k+1}^fE_{4|k}^fE_{1|k}^f$. Omitting the inside line segments will obtain a polygon which is colored in orange. Then the following lemma is proposed:

**Lemma 1.** If the difference of vehicle’s orientation at $k$ and $k + 1$ is less than $\pi$, the orange polygon can cover the red region with a proper $\alpha$.

**Proof.** Intuitively, the greater $\alpha$ is, the larger the polygon is. Taking $\xi = [0, 0, 0, 0]^T$ as the origin configuration, where the coordinates of $E_{i}^f, (i = 1, 2, 3, 4)$ are $[v_f, -\frac{w_f}{2} T, v_f, -\frac{w_f}{2} T]$.

$[-l_f, -\frac{w_f}{2} T, -l_f, \frac{w_f}{2} T]$, the trajectory of point $E_{i}^f (i = 1, 2, 3, 4)$ on the VPF can be obtained by Equation (9).

\[
\begin{align*}
[X_{E_i}^f] &= \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} l_f \\ \frac{w_f}{2} \end{bmatrix}, \\
[Y_{E_i}^f] &= \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} l_f \\ -\frac{w_f}{2} \end{bmatrix},
\end{align*}
\]

Replacing $\frac{w_f}{2}$ with $w_f$, the trajectory of point $E_i (i = 1, 2, 3, 4)$ is given by Equation (10):

\[
\begin{align*}
[X_{E_i}^f] &= \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} l_f \\ \frac{w_f}{2} \end{bmatrix}, \\
[Y_{E_i}^f] &= \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} l_f \\ -\frac{w_f}{2} \end{bmatrix},
\end{align*}
\]

By curvature equation, radius of curvature at point $E_i^f$ is

\[
R_{E_i}^f = \frac{1}{\kappa_{E_i}^f},
\]

\[
= \left[ \left( 1 - \frac{w_f}{2} \cdot \kappa_i \right)^2 + l_f^2 \cdot \kappa_i^2 \right]^{\frac{1}{2}},
\]

where $\kappa = \frac{\tan \delta_f}{l}$. Similarly, radius of curvature at other points are

\[
R_{E_2}^f = \left( 1 + \frac{w_f}{2} \cdot \kappa_i \right)^2 + l_f^2 \cdot \kappa_i^2, \quad R_{E_3}^f = \left( 1 - \frac{w_f}{2} \cdot \kappa_i \right)^2 + l_f^2 \cdot \kappa_i^2, \quad R_{E_4}^f = \left( 1 - \frac{w_f}{2} \cdot \kappa_i \right)^2 + l_f^2 \cdot \kappa_i^2.
\]

\[
Error_{E_i}^f = -\left( R_{E_{i+1}^f} - R_{E_{i+1}^f} \cdot \cos \left( \frac{\varphi_i + \frac{1}{2} \cdot \varphi_f}{2} \right) \right),
\]

\[
Error_{E_{i+1}^f}^f = R_{E_{i+1}^f} - R_{E_{i+1}^f} \cdot \cos \left( \frac{\varphi_i + \frac{1}{2} \cdot \varphi_f}{2} \right).
\]
The guaranteed collision-free trajectory

FIGURE 5 Taking turning right as example, \(i = 1, 4\) in (a) and \(i = 2, 3\) in (b), fit the arc \(E_{i|k}^f\rightarrow E_{i|k+1}^f\) by the line segment \(E_{i|k}^f\rightarrow E_{i|k+1}^f\). The arc \(E_{i|k}^f\rightarrow E_{i|k+1}^f\) should be inside the line segment to ensure the coverage

\[
\text{Error}_{E_{i|k}^f} = R_{E_{i|k}^f} - R_{E_{i|k}^f} \cdot \cos \left( \frac{\left[ x_k T + \frac{1}{2} a_k T^2 \right] \cdot x_k}{2} \right),
\]

(12c)

\[
\text{Error}_{E_{i|k}^f} = -\left( R_{E_{i|k}^f} - R_{E_{i|k}^f} \cdot \cos \left( \frac{\left[ x_k T + \frac{1}{2} a_k T^2 \right] \cdot x_k}{2} \right) \right).
\]

(12d)

Equation (11) are all continuous functions. Note that for a circular arc in time interval \([k, k+1]\) (see Figure 5), given the difference of vehicle’s orientation of between \(k\) and \(k+1\) is less than \(\pi\), the place with the maximum error is at the midpoint of arc when fitting the arc by a line segment. The errors are defined with Equation (12).

With the above definition, \(\text{Error}_{E_{i|k}^f}\) is greater than zero if the line segment is inside the region which VPF passes through and \(\text{Error}_{E_{i|k}^f} < 0\) means line segment is outside the region.

Next, the cover property is certified in an indirect way. Define the difference between the radius of \(E_{i|k}^f\) and \(E_{i|k}^f\) as \(\delta R_{i|k}(i = 1, 2, 3, 4)\) according to:

\[
\delta R_{i|k} = R_{E_{i|k}^f} - R_{E_{i|k}^f},
\]

(13a)

\[
\delta R_{2|k} = -\left( R_{E_{2|k}^f} - R_{E_{2|k}^f} \right),
\]

(13b)

\[
\delta R_{3|k} = -\left( R_{E_{3|k}^f} - R_{E_{3|k}^f} \right),
\]

(13c)

\[
\delta R_{4|k} = R_{E_{4|k}^f} - R_{E_{4|k}^f}.
\]

(13d)

\(\delta R_{i|k}\) should always be greater than zero because the VPF is larger than the border of vehicle. Then the distance between the line segment and the inner circular arc is defined as:

\[
distance_{i|k} = \text{Error}_{i|k} - \delta R_{i|k}, i = 1, 2, 3, 4.
\]

(14)

The cover property is satisfied when the following relationship holds:

\[
\exists \alpha : \text{distance}_{i|k} \leq 0,
\]

\[
\text{for } \forall i = 1, 2, 3, 4, \forall k = 0, 1, 2, \ldots, N - 1.
\]

Therefore, the orange polygon can totally cover the region where the vehicle passes through.

Lemma 1 illustrates that the orange polygon must have no intersection with obstacles to ensure a collision-free trajectory in every time interval. It is too complex to calculate the orange polygon analytically. Here define \(\{E_{1|k}^f, E_{2|k}^f, E_{3|k}^f, E_{4|k}^f\}\) as \(E_{k\rightarrow k+1}\). In the light of Lemma 1, during a time interval in Figure 4(b), the collision avoidance constraints are presented as:

\[
\left( \overrightarrow{E_{k\rightarrow k+1}} + \overrightarrow{D} \right),
\]

(16)

where \(\overrightarrow{E_{k\rightarrow k+1}} \in E_{k\rightarrow k+1}\) and \(\overrightarrow{D} \in \mathcal{O}\).

Remark 2. Cormen et al. [21] have proposed a method to determine whether a line segment intersects with another. If the endpoints of the two line segments satisfy the relationship in the Appendix, the two segments do not intersect.

3.3 The guaranteed collision-free trajectory planning algorithm

After projecting the infinite-dimensional function space to the finite-dimensional vector space, the CDOP is reformulated as an NLP problem:

\[
\min_{\{\psi(k)^N\}_{k=1}} T \cdot N,
\]

(17a)
Theorem 1. Under Assumption 1, Algorithm 1 converges to \( \alpha_{\text{res}} \) and generates an optimal solution which is collision-free in the whole time horizon.

Proof. When \( \alpha = 1 \), the solution \( \{ Traj(k) \}_{k=0}^{N} \) and \( \{ u(k) \}_{k=1}^{N} \) obtained from step 1 may not be collision-free in the whole time horizon according to the analysis in Section 3.2. A larger \( \alpha \) can guarantee a collision-free but conservative solution. Suppose \( \alpha^* \) is the minimum \( \alpha \) results in a trajectory that happens to not collide, that is,

\[
\alpha^* = \max \{ \inf_{\alpha_{[i,k]}} \text{distance}_{[i,k]} \leq 0 \}, \quad \forall i = 1, 2, 3, 4, \forall k = 0, 1, 2, \ldots, N - 1, \tag{18}
\]

\[
f'(\alpha_{[i,k]}) = \frac{\left( 1 - \frac{\alpha_{[i,k]} \cdot \kappa_k}{2} \right)^2 + l_f^2 \cdot \kappa_k^2}{2} \cdot \cos \left( \frac{1}{2} \cdot \frac{\nu_k \cdot T + \frac{1}{2} \cdot a_k \cdot (T)^2 \cdot \kappa_k}{2} \right), \tag{19}
\]

\[
f'(\alpha_{[i,k]}) = \frac{-w + \frac{\alpha_{[i,k]} \cdot w^2 \cdot \kappa_k}{2} \left( 1 - \frac{\alpha_{[i,k]} \cdot \nu_k \cdot \kappa_k}{2} \right)^2 + l_f^2 \cdot \kappa_k^2}{2} \cdot \cos \left( \frac{1}{2} \cdot \frac{\nu_k \cdot T + \frac{1}{2} \cdot a_k \cdot (T)^2 \cdot \kappa_k}{2} \right), \tag{20}
\]

For any \( i \) and \( K \), according to the knowledge of monotonicity of \( \text{distance}_{[i,k]} \), it is apparent that the larger \( \alpha \) is, the less \( \text{distance}_{[i,k]} \) is. To ensure that the infimum exists, it needs to be determined that an \( \alpha(\alpha \geq 1) \) exists making \( \text{distance}_{[i,k]} = 0 \).

Taking \( i = 1 \) as an example, the \( \text{distance}_{[i,k]} \) is a function of \( \alpha_{[i,k]} \) as Equation (19) and the derivative of \( f'(\alpha_{[i,k]}) \) is given by Equation (20). When turning right, the derivative is always negative and \( f'(\alpha_{[i,k]}) \) is monotonic decreasing. Since \( f'(\alpha_{[i,k]}) > 0 \) when \( \alpha_{[i,k]} = 1 \), there must be a root near one for \( \alpha_{[i,k]} \) (see Figure 6). It is the same when turning left. As a result, the infimum of \( \alpha_{[i,k]} \) exists for any \( i \) and \( k \). Among all the possible infimum values, the maximum one is selected to ensure \( \text{distance}_{[i,k]} \leq 0 \) for all \( i \) and \( k \).

4 | SIMULATION RESULTS

In this section, simulation results in three common parking scenarios are presented, including reverse parking, parallel parking and diagonal parking. Then the proposed method is verified in a more complex parking scenario. All simulations are performed on a Windows 10 OS with an Intel i5 processor clocked at 1.8 GHz and the programming language is MATLAB with version 2019b. An open-source Matlab toolbox CasADi [22, 23] is chosen to solve the NLP problem.
4.1 Parking in common parking scenarios

As defined in Figure 1, the vehicle model is regarded as a rectangle of size 4.5 m × 1.9 m. The wheelbase is $L = 2.8$ m. Limitations and start/final configurations are listed in Table 1.

In the reverse parking scenario, the parking spot is 2.5 m wide and 5.3 m long. The parking spot in the parallel parking scenario is 2.5 m wide and 6 m long and in the diagonal scenario, it is also 2.5 m wide and 6 m long, facing at 45 degrees. In all three cases, a 6 m wide road is prepared for maneuvering. Results are shown in Figures 7–9. The gray regions are obstacles such as other parking spots and road boundaries. The red regions are the place which the vehicle passes through. In Figures 7–9, the planned trajectory of the center of the rear axle is shown in the black solid curve. The planned control inputs are shown in the second subgraph, respectively.

In Figure 7, the vehicle goes forwards and turns left slowly, then the vehicle should switch to reverse gear, at the same time turn right to move into the spot. The control inputs also satisfy the constraints listed in Table 1 and the planned trajectory is reasonable and feasible which will be illustrated in the next section.

It should be noted that in parallel parking (see Figure 8), at the final stage, the vehicle goes forward and turns right to align with the spot, which is similar to the action of a human driver. This action is necessary especially when the parking spot is not long enough.

Diagonal parking is not very common in daily life. Its difficulty is between that of reverse parking and parallel parking. In general, diagonal parking should start at a specific point, that is, the vehicle should start at left if the parking spot faces 45 degrees as Figure 9. An improper starting point may result in a failing trajectory planning.

4.2 Parking in irregular scenarios

In this subsection, the vehicle needs to park into an irregular spot which is slightly narrow in someplace. The
parameters of vehicle model are the same with Section 4.1 and the start/ final configurations are $\xi_s = [2.0, 6.0, \pi/24, 0, 0]$ and $\xi_f = [7.0, 2.0, \pi/2, 0, 0]$, respectively. The narrowest part of the path is about 2.0 m and the vehicle needs to get through it. As shown in Figure 10, the vehicle still has no collision during parking especially at some crucial points such as (5,7.5) and (5,5.3).

5 | DISCUSSION

5.1 | Performance comparison with the ordinary trajectory planning algorithm

As emphasised before, the most important feature of the algorithm proposed in this paper is to guarantee the trajectory is collision-free during the whole time horizon. It may be hard to explain the collision-free property from Figures 7 to 9 directly. In Figure 11, by zooming in point (5,5) when reverse parking and points (6,2), (12,2.5) when parallel parking, it illustrates that there is indeed no collision. To demonstrate the effect of constraint (17e), another simulation is carried on with considering collision avoidance constraints only at the $N$ time points. This method is regarded as a conventional CDOP-based planning algorithm. Note that for most CDOP-based trajectory planning approaches, collision avoidance constraints are implemented in different ways but with the same effect, so this comparison is representative.

As shown in Figure 12, in the parallel parking scenario, the same initial conditions are given and especially $\alpha = 1.0316$. It is apparent that the red region overlaps the gray one, which is clearly an undesired outcome. The vehicle may get a small scratch or even fail to parking. In general terms, one could argue that inflating the obstacle region is also an effective approach. Then even the vehicle has collisions with the larger region, it will not collide actually. It is truly a good idea but cannot solve the problem systematically.

5.2 | Computational efforts

In the reverse parking scenario, the computational time for the optimisation problem (17) is 2.19 s. By contrast, the computational time for parallel parking and diagonal parking are 2.31 s and 2.04 s, respectively. In Table 2, when considering collision constraints only at the total $N$ time points (without constraints (17e)), the optimisation problem is solved in a shorter time (1.47 s in reverse parking; 1.78 s in parallel parking, 2.75 s in

FIGURE 8  Case 2: Parallel parking. The region that the vehicle occupies has no intersection with obstacles. (6,2) and (12,2.5) are two critical points. $t_f = 32.569$s and $\alpha_{g/f} = 1.0316$

FIGURE 9  Case 3: Diagonal parking. The region that the vehicle occupies has no intersection with obstacles. $t_f = 22.655$s and $\alpha_{g/f} = 1.0139$
FIGURE 10 Case 4: Parking in an irregular scenario. The vehicle needs to get through the narrow path. \( t_f = 16.684 \text{ s} \) and \( \alpha = 1.0199 \)

TABLE 2 Computational time to solve the optimisation problem in three parking cases

|                          | Reverse | Parallel | Diagonal |
|--------------------------|---------|----------|----------|
| Consider collision constraints during the whole horizon | 2.19 s  | 2.31 s   | 2.04 s   |
| Consider collision constraints only at the \( N \) time points | 1.47 s  | 1.78 s   | 2.75 s   |
| Computational time increase | 48.97% | 29.77%   | -25.82%  |
| Solve \( \alpha \)       | 7.91 s  | 11.39 s  | 12.22 s  |

diagonal parking). In general, the absolute computational time is acceptable in real applications. It seems that the computational time has no significant relationship with the number of time intervals.

Computational time to solve \( \alpha \) are also shown in Table 2. It is a fair time-consuming part of the proposed algorithm. However, the parallel computing method might be helpful here to speed up the computation, which is beyond the scope of this paper.

5.3 Application of the designed trajectory

Different from path planning approaches like Hybrid A*, trajectory planning methods not only offer a reference path, but also consider the control execution time. Specifically, \( \{\text{Traj}(k)\}_{k=0}^{N} \) and \( \{u(k)\}_{k=1}^{N} \) obtained from Algorithm 1 are very suitable to design a trajectory tracking controller.

Model predictive control (MPC) has a wide application ranging from process industry to robot systems [17, 18, 24–28]. As a nonlinear model, Equation (1) should be linearised first. Specifically, linearising the kinematics model along the reference trajectory obtained by the proposed trajectory planning method, that is, \( \{\text{Traj}(k)\}_{k=0}^{N} \) and \( \{u(k)\}_{k=1}^{N} \), a control law is accessible from MPC by solving a quadratic program problem [29].

In the application, CarSim is employed to simulate the real vehicle and the MPC controller is an S-Function in Matlab. The whole planning and control scheme can be described as Figure 13.

The target speed and steering angle commands are used as the inputs of CarSim. Then the real states of the vehicle will be sampled for MPC to calculate the target control commands at the next sample time.

Figure 14 shows the target commands calculated by the MPC controller and the real states of the vehicle in CarSim. Due to the reliability of low-level controllers (e.g. PID controllers) in CarSim. The actual speed and steering angle are very close to the
target commands, hence the actual trajectories (see Figures 7–9) are almost the same with the planned ones. Table 3 shows the maximum tracking error in three scenarios when using the MPC controller. In addition, the control inputs are slightly different from the inputs/states planned by the proposed method since the kinematics model cannot always accurately represent the real vehicle, which also validates the stability of MPC [30].

| Scenario        | Max tracking error |
|-----------------|--------------------|
| Reverse         | 0.012 m            |
| Parallel        | 0.020 m            |
| Diagonal        | 0.009 m            |

6 CONCLUSION

In this paper, a guaranteed collision-free trajectory planning algorithm for autonomous parking is presented. The parking problem is described as a dynamic optimisation problem which
ACKNOWLEDGEMENTS
This work is supported by National Key Research and Development Program of China (NO. 2016YFB0303404).

ORCID
Zhining Zhang https://orcid.org/0000-0002-2814-4255

REFERENCES
1. Wang, W., et al.: Automatic parking of vehicles: A review of literatures. Int. J. Automatic Technol. 15(6), 967–978 (2014)
2. Gómez-Bravo, F., et al.: Parallel and diagonal parking in nonholonomic autonomous vehicles. Eng. Appl. Artificial Intell. 14(4), 419–434 (2001)
3. Lee, C.K., et al.: Autonomous vehicle parking using hybrid artificial intelligence approach. J. Intell. Rob. Sys. 56(3), 319–343 (2009)
4. Liu, W., et al.: Parking like a human: A direct trajectory planning solution. IEEE Trans. Intell. Transportation Sys. 18(12), 3388–3397 (2017)
5. Panomruttranarug, B.: Application of iterative learning control in tracking a Dubin's path in parallel parking. Int. J. Automatic Technol. 18(6), 1099–1107 (2017)
6. González, D., et al.: A review of motion planning techniques for automated vehicles. IEEE Trans. Intell. Transp. Syst. 17(6), I135–I145 (2015)
7. Paden, B., et al.: A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans. Intell. Veh. 1(1), 33–55 (2016)
8. Marchese, F.M.: Multiple mobile robots path-planning with MCA. In: International Conference on Autonomous and Autonomous Systems (ICAS96), pp. 56–56. IEEE, Piscataway (2006)
9. Ferguson, D., et al.: Motion planning in urban environments. J. Field Robot. 25(11–12), 939–960 (2008)
10. Dolgov, D., et al.: Path planning for autonomous vehicles in unknown semi-structured environments. Int. J. Robot. Res. 29(5), 485–501 (2010)
11. LaValle, S.M., Kuffner, Jr., J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. 20(5), 376–400 (2001)
12. Petrov, P., Nashashibi, F.: Modeling and nonlinear adaptive control for autonomous vehicle overtaking. IEEE Trans. Intell. Transp. Syst. 15(4), 1643–1656 (2014)
13. Rastelli, JP., et al.: Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings, pp. 510–515. IEEE, Piscataway (2014)
14. Thrän, S., et al.: Stanley: The robot that won the DARPA grand challenge. J. Field Robot. 23(9), 661–692 (2006)
15. Gu, T., Dolan, J.M.: On-road motion planning for autonomous vehicles. In: International Conference on Intelligent Robotics and Applications, pp. 588–597. Springer, Cham (2012)
16. Ziegler, J., et al.: Trajectory planning for Bertha—A local, continuous method. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings, pp. 450–457. IEEE, Piscataway (2014)
17. Zhang, X., et al.: Autonomous parking using optimization-based collision avoidance. In: 2018 IEEE Conference on Decision and Control (CDC), pp. 4327–4332. IEEE, Piscataway (2018)
18. Li, B., Shao, Z.: A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. Knowl. Based Syst. 86, 11–20 (2015)
19. Rajamani, R.: Vehicle Dynamics and Control. Springer Science & Business Media, New York (2011)
20. Betts, J.T., Huffman, W.P.: Mesh refinement in direct transcription methods for optimal control. Optimal Control Appl. Meth. 19(1), 1–21 (1998)
21. Cimen, T.H., et al.: Introduction to Algorithms. MIT Press, Cambridge (2009)
22. Andersson, J.A.E., et al.: CasADi — A software framework for nonlinear optimization and optimal control. Math. Program. Comput. 11, 1–36 (2018)
23. Biegler, L.T.: Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes, vol. 10. SIAM, Philadelphia (2010)
24. Borrelli, F., et al.: Collision-free UAV formation flight using decentralized optimization and invariant sets. In: 2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601), vol. 1, pp. 1099–1104. IEEE, Piscataway (2004)
25. Hsieh, M.F., Ozguner, U.: A parking algorithm for an autonomous vehicle. In: 2008 IEEE Intelligent Vehicles Symposium, pp. 1155–1160. IEEE, Piscataway (2008)
26. Liu, Z., et al.: Fast linear parameter varying model predictive control of buck dc–dc converters based on FPGA. IEEE Access 6, 52434–52446 (2018)
27. Zhang, Z., et al.: Rail pressure controller design of GD basing on predictive functional control. Control Theory Technol. 17(2), 176–182 (2019)
28. Zhang, X., et al.: Predictive functional control for integrator systems. J. Franklin Inst. 357(7), 4171–4186 (2020)
29. Lu, P.C., et al.: Iterative learning control (ILC)-based economic optimization for batch processes using helpful disturbance information. Ind. Eng. Chem. Res. 57(10), 3717–3731 (2018)
30. Rawlings, J.B., Muske, K.R.: The stability of constrained receding horizon control. IEEE Trans. Autom. Control 38(10), 1512–1516 (1993)

APPENDIX A
A convex combination of two distinct points \( p_1 = (x_1, y_1) \) and \( p_2 = (x_2, y_2) \) is any point \( p_3 = (x_3, y_3) \) such that for some \( \beta \) in the range \( 0 \leq \beta \leq 1 \), we have \( p_3 = \beta p_1 + (1 - \beta) p_2 \). In other words, \( p_3 \) is any point that is on or between \( p_1 \) and \( p_2 \) on the line. Given two distinct points \( p_1 \) and \( p_2 \), the line segment \( p_1 p_2 \) is the set of convex combinations of \( p_1 \) and \( p_2 \), which are called endpoints of segments \( p_1 p_2 \).

Considering two vectors \( v_1 = (x_1, y_1) \) and \( v_2 = (x_2, y_2) \) starting at the origin, the determinant of a matrix is adopted as the
Algorithm 2 Determining whether two line segments intersect

Input: End point \( p_1, p_2, p_3, p_4 \)

\[ d_1 = \text{Direction}(p_3, p_4, p_1) \]
\[ d_2 = \text{Direction}(p_3, p_4, p_2) \]
\[ d_3 = \text{Direction}(p_1, p_2, p_3) \]
\[ d_4 = \text{Direction}(p_1, p_2, p_4) \]

if \( d_1 \cdot d_2 \leq 0 \) and \( d_3 \cdot d_4 \leq 0 \)

return INTERSECTANT

else

return DISJOINT
endif

Direction\((p_i, p_j, p_k)\):

return \((p_k - p_i) \times (p_j - p_i)\)

definition of cross product:

\[ v_1 \times v_2 = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} = x_1y_2 - x_2y_1 = -v_2 \times v_1. \]

If \( v_1 \times v_2 \) is positive, then \( v_1 \) is clockwise from \( v_2 \) with respect to the origin \((0,0)\); if it is negative, then \( v_1 \) is counterclockwise from \( v_2 \). A boundary condition arises if the cross product is 0; in this case, the vectors are colinear.

To determine whether two line segments intersect, just check whether each segment straddles the line containing the other. A segment \( p_1p_2 \) straddles a line if point \( p_1 \) lies on one side of the line and point \( p_2 \) lies on the other side. Two line segments intersect if and only if either (or both) of the following conditions holds:

1. Each segment straddles the line containing the other.
2. An endpoint of one segment lies on the other segment.

(A natural condition derived from the boundary case)

Given segments \( p_1p_2 \) and \( p_3p_4 \), the following procedures in Algorithm 2 implement this idea. Consequently, two line segments are disjoint if and only if the following relationship holds:

\[ \max(d_1 \cdot d_2, d_3 \cdot d_4) > 0, \]

\( d_i \) has been defined in Algorithm 2.