Robotica (2026), **44**, pp. 1804–1829
doi:10.1017/S0263574726103737

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Vertical parking path planning based on Reeds–Shepp curve and an improved particle swarm optimization algorithm

Guangqi Hu, Likun Cui, Hao Wang and Xinle Li

School of Mechanical Engineering, Shaanxi University of Technology, China
Corresponding author: Likun Cui; E-mail: cuilikun@snut.edu.cn

Received: 13 May 2025 UTC; **Revised:** 27 April 2026 UTC; **Accepted:** 7 June 2026 UTC;
First published online: 17 July 2026 UTC

Keywords: vertical parking; path planning; Reeds–Shepp curve; improved particle swarm optimization algorithm

Abstract

Current geometric-based path-planning methods for vertical parking in automated parking systems face issues such as stringent initial position requirements, insufficient clearance from garage boundaries, and discontinuous curvature. To address these problems, this paper proposes a parking path-planning method that integrates the Reeds–Shepp curve with an improved particle swarm optimization (PSO) algorithm. The parking process is decomposed into forward and backward segments, each generated using the Reeds–Shepp curve to enhance flexibility. The shift point, defined as the transition between the two segments, serves as the decision variable in the PSO framework. The fitness function jointly considers path length and deviation between the target pose and garage entry. A quadratic Bézier curve and a nonlinear adaptation mechanism are incorporated into the PSO to determine the optimal shift point, mitigating the risk of proximity to garage boundaries. The initial path is then generated using the Reeds–Shepp curve based on the initial pose, shift point, and target pose. After discretization, the path points are optimized via a nonlinear optimization formulation to eliminate curvature discontinuities at segment junctions. Experimental results demonstrate that the proposed method generates feasible, continuous-curvature parking paths across a variety of initial poses, outperforming conventional approaches in terms of flexibility and clearance.

1. Introduction

The growing number of vehicles has made parking shortages a critical urban issue. Since drivers often rely on empirical adjustments and struggle to perceive their surroundings accurately, especially in rear blind zones, parking operations inevitably involve increased complexity and collision risks [1, 2]. Advanced driver-assistance systems, capable of autonomously identifying parking spaces and executing parking maneuvers [3], rely on automated parking systems (APS) for precise trajectory tracking and optimal path planning. In APS path planning, onboard sensors gather environmental data to generate kinematically feasible and collision-free trajectories [4]. Common strategies include curve interpolation, graph search, random sampling, numerical optimization, and artificial intelligence (AI) methods [5–7].

Dubins [8] introduced a geometry-based approach to derive minimal-length paths between start and end positions, though its forward-motion-only assumption limits applicability to elementary parking scenarios. Reeds and Shepp [9] advanced this framework by incorporating both forward and reverse motions, generating optimal path combinations through circular arcs and straight lines, significantly improving the flexibility of parking path planning.

With recent advancements in intelligent algorithms and neural networks, many scholars have integrated the Reeds–Shepp curve with other optimization algorithms, yielding a series of noteworthy results. Jiang and Li [10] combined reinforcement learning with the Reeds–Shepp curve to achieve path planning in complex static obstacle parking scenarios, enhancing the success rate of automated parking trajectory planning. Jhang et al. [11] proposed a revised Bidirectional Rapidly-Exploring Random Tree*

© The Author(s), 2026. Published by Cambridge University Press.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

algorithm, which integrates the Bidirectional Rapidly-Exploring Random Tree* algorithm (Bi-RRT*) with Reeds–Shepp curves, hereafter denoted as the Bi RRT* Reeds–Shepp algorithm. This design inherently ensures that the generated paths comply with vehicle kinematic constraints. Furthermore, by modifying the cost function to reduce path sharpness, the algorithm produces trajectories more consistent with human driving preferences. Li et al. [12] used particle swarm optimization to modify the turning radius of the Reeds–Shepp curve, achieving path planning for unmanned underwater vehicles and minimizing path length while satisfying kinematic constraints. However, this algorithm only adjusted the turning radius of the Reeds–Shepp curve. In spatially constrained vertical parking environments, it fails to address the inherent obstacle distance and intrinsic curvature issues of the traditional Reeds–Shepp curve. Furthermore, in narrow parking scenarios, a single Reeds–Shepp curve cannot generate an optimal path that fully avoids obstacles.

Lim et al. [13] adopted the standard straight-turn reference line as a guide to ensure the feasibility of the parking trajectory generated by sequentially connected quadratic Bézier curves. Guo [14] proposed an automatic perpendicular parking path-planning algorithm that considers the vehicle's initial heading angle, termed IHA-PPP. This method determines the feasible parking start point based on the vehicle's kinematic parameters and the target parking spot. The parking type (single-step or multi-step) is determined by collision-avoidance constraints, and a smooth path with continuous curvature is generated through the application of a quartic spline curve. Wang [15] and Nelson [16] utilized polynomial curves to address the issue of abrupt changes in path curvature, resulting in smoother paths. However, these geometric methods impose strict requirements on the initial conditions and exhibit limited adaptability to diverse scenarios. Zhang et al. [17] utilized a fifth-degree polynomial for path planning in parallel parking scenarios to ensure the continuity of curvature along the trajectory. Zhang et al. [18] formulated a dynamic optimization problem and discretized it into a finite-dimensional nonlinear programming problem to solve for collision-free paths. Daniali et al. [19] employed a sleeveoid-based sequence combined with Linear Programming (LP) for parking path planning, eliminating the need for steering adjustments during parking. Sim et al. [20] proposed an autonomous parking path-planning algorithm based on adaptive-resolution horizontal and vertical grids. By exploring the constructed feasible motion space in 16 discrete directions and subsequently applying cubic spline interpolation for optimization, the algorithm generates curvature-continuous, collision-free paths. Guo [21] integrated the grid method with the Ant Colony Optimization (ACO) algorithm to generate the shortest collision-free path, followed by Bézier curve smoothing. Despite their effectiveness, but these methods neglect the safety margin between the planned path and the garage boundaries, potentially resulting in proximity to obstacles and higher collision risks. Piao et al. [22] proposed a method for multi-sensor information integration and adaptive trajectory generation using parking space recognition, which also showed good results when vehicles were not initially in parallel. Du and Tan [23] designed parking paths by combining segmented arcs and straight lines tailored to specific scenario requirements. When direct optimization of the path was infeasible, they implemented an initial vehicle-position adjustment prior to path planning to guarantee feasible trajectory generation under arbitrary initial orientations. Sedighi et al. [24] proposed a fusion of the Hybrid A* search algorithm with visibility graph planning to generate the shortest nonholonomic parking path. However, the paths obtained by these methods all have the problem of curvature discontinuity, which increases the difficulty of subsequent path tracking. In summary, to address these issues, this paper proposes a parking method based on the Reeds–Shepp curve and an improved Particle Swarm Optimization algorithm. The proposed method can enable smooth path planning under various initial positions, improve planning flexibility, and guarantee an appropriate vehicle position after garage entry. The key contributions are threefold.

(1) To address the strict initial position requirements for parking in vertical garages, it is recommended to use two Reeds–Shepp curves to construct the parking path, which enhances flexibility in planning parking trajectories from different starting positions.

(2) To address the problem that the traditional Reeds–Shepp curve does not consider the distance to obstacles, the point connecting two segments of the path is defined as the shift point; the deviation

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

between the position when entering the garage with the target point and the total length of the path are weighted as the fitness function. By using an improved particle swarm optimization algorithm to search for the optimal shift point position and selecting the corresponding Reeds–Shepp curve pattern for the optimal shift point, the vehicle achieves better positioning upon entering the garage while avoiding the planned path from getting too close to the garage edge.

(3) To address the curvature discontinuity issue in geometric planning of parking paths, design a nonlinear optimization problem based on discrete path points. Use two Reeds–Shepp curves as the initial solution for this nonlinear optimization problem. By solving this nonlinear optimization problem, we obtain a path with continuous curvature.

2. Parking path analysis based on the Reeds–Shepp curve

2.1 Reeds–Shepp curve

The foundation of the Reeds–Shepp curve was laid in 1990 when J. A. Reeds and L. A. Shepp improved on Dubins' work by integrating backward motion into path planning. Eqs. (1), (3) provide a mathematical description of the positional changes that resulted from this groundbreaking work, which extended the vehicle's basic motions to three basic modes: left turns (l), straight movements (s), and right turns (r).

$$l_t^+ = \begin{cases} x_1 = x_0 + R\left[\sin\left(\theta_0 + t\right) - \sin\theta_0\right] \\ y_1 = y_0 - R\left[\cos\left(\theta_0 + t\right) - \cos\theta_0\right] \\ \theta_1 = \theta_0 + t \end{cases} \quad l_t^- = \begin{cases} x_1 = x_0 + R\left[\sin\left(\theta_0 - t\right) - \sin\theta_0\right] \\ y_1 = y_0 - R\left[\cos\left(\theta_0 - t\right) - \cos\theta_0\right] \\ \theta_1 = \theta_0 - t \end{cases} \tag{1}$$

$$r_t^+ = \begin{cases} x_1 = x_0 - R\left[\sin\left(\theta_0 - t\right) - \sin\theta_0\right] \\ y_1 = y_0 + R\left[\cos\left(\theta_0 - t\right) - \cos\theta_0\right] \\ \theta_1 = \theta_0 - t \end{cases} \quad r_t^- = \begin{cases} x_1 = x_0 - R\left[\sin\left(\theta_0 + t\right) - \sin\theta_0\right] \\ y_1 = y_0 + R\left[\cos\left(\theta_0 + t\right) - \cos\theta_0\right] \\ \theta_1 = \theta_0 + t \end{cases} \tag{2}$$

$$s_t^+ = \begin{cases} x_1 = x_0 + R\left(t\cos\theta_0\right) \\ y_1 = y_0 + R\left(t\sin\theta_0\right) \\ \theta_1 = \theta_0 \end{cases} \quad s_t^- = \begin{cases} x_1 = x_0 - R\left(t\cos\theta_0\right) \\ y_1 = y_0 - R\left(t\sin\theta_0\right) \\ \theta_1 = \theta_0 \end{cases} \tag{3}$$

Where l, s, and r denote a left turn, straight motion, and a right turn, respectively. $t$denotes either the arc's radian measure or its equivalent straight-line length, plus or minus sign denotes the direction of traveling, R denotes the turning radius of the vehicle, (x_0, y_0) and (x_1, y_1) denote the cartesian coordinates before and after the movement, while θ_0 and θ_1 represent the corresponding heading angles before and after the movement, respectively. The Reeds–Shepp curve generates optimal paths by combining three to five basic path segments, resulting in 48 standardized patterns. These patterns are represented using "C" to denote arc segments (left turn l or right turn r), "S" for straight segments, and "|" to indicate directional changes. The 48 patterns can be classified into nine distinct path types [25]. The initial and final points are set separately for the forward and reverse path segments during parking. The schematic of the resulting Reeds–Shepp curve planning is illustrated in Figure 1.

2.2 Vehicle model

The parking process is a low-speed driving process, and the lateral slip of the tires can be ignored. The vehicle information and parking space schematic during parking are shown in Figure 2. Without loss of generality, the vehicle is simplified as a rectangle here.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

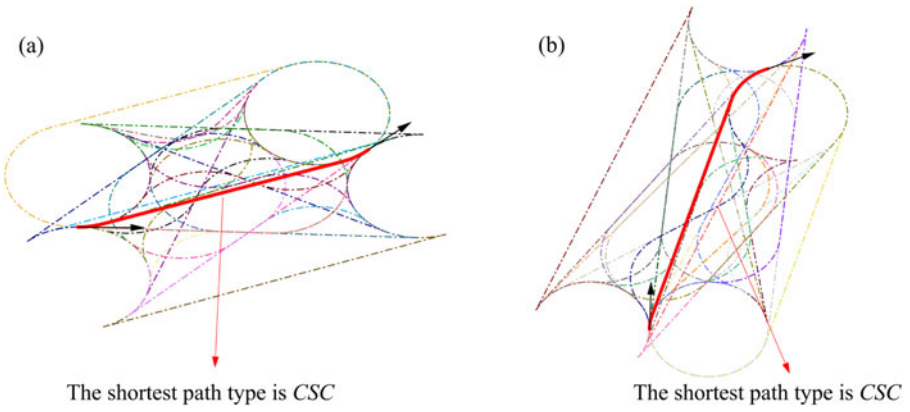


Figure 1. *Schematic diagram of the Reeds–Shepp curve planning for two paths. (a) Path-planning schematic for forward motion; (b) path-planning schematic for backward motion.*

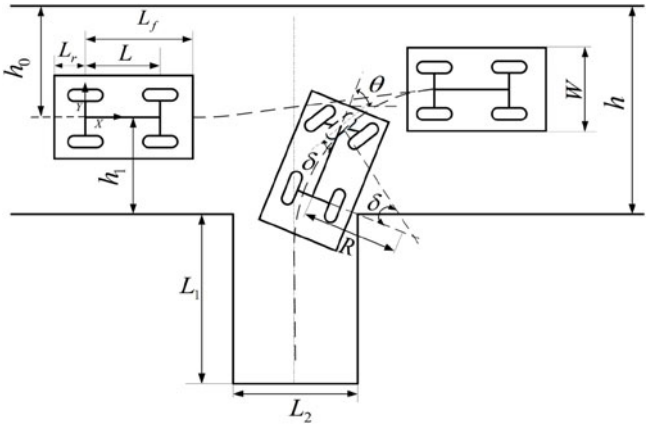


Figure 2. *Schematic diagram of parking in a vertical garage.*

Where L is the wheelbase of the vehicle, L_f and L_r are distances from the center of the rear axle to the front end of the vehicle and rear end of the vehicle, respectively. W is the width of the vehicle, h is the width of the lane, and h_0 is the longitudinal distance between the origin of the coordinates and the road boundary. h_1 is the longitudinal distance between the origin and the entrance of the garage. θ is the vehicle heading angle, and δ is the equivalent front wheel angle. R is the turning radius of the vehicle. Taking the center of the vehicle's rear axle as the reference point, both forward and reverse states during parking are considered. To improve the flexibility of parking path planning, the two-stage parking process is divided into forward and reverse phases, in which the Reeds–Shepp curve is employed to generate paths for each phase. However, not all combinations of Reeds–Shepp curves are applicable to parking scenarios. Therefore, suitable candidate Reeds–Shepp curves need to be identified separately for the forward and reverse phases.

2.3 Forward parking path analysis

In order to reduce the number of times a vehicle changes its direction of motion during forward movement when using the Reeds–Shepp algorithm for path planning, it is necessary to select path patterns with fewer basic actions and consistent forward direction. Among the nine path types of the Reeds–Shepp curve, the *CSC* (Circle-Straight-Circle) type satisfies the direction consistency requirement.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

Patterns with consistent forward motion direction include the following: However, in a perpendicular parking environment, the shift point is located to the upper left of the parking starting point. Consequently, executing a right turn-straight-right turn sequence from the parking starting point is impractical. The $r_t^+ s_u^+ r_v^+$ path pattern is excluded from this pattern category. The suitable path patterns candidates are $l_t^+ s_u^+ l_v^+$, $l_t^+ s_u^+ r_v^+$ and $r_t^+ s_u^+ l_v^+$, where t, u, v denote the radian measure of the arc segments or the lengths of the straight-line segments, respectively. The normalized initial position of the forward trajectory is defined as $(0, 0, 0°)$, and the target position coordinates are (x_a, y_a, θ_a). Applying the $l_t^+ s_u^+ l_v^+$ transformation to the normalized initial position in Eqs. (1)–(3) yields:

$$\begin{aligned} (u, t) &= R(x - \sin\theta_q, y + \cos\theta_q - 1) \\ v &= M(\theta_q - t) \end{aligned} \tag{4}$$

where the function $M(x)$ is used to restrict the range of x to $(-\pi, \pi)$. Solving the above equation yields a path length of:

$$l_{l_t^+ s_u^+ r_v^+} = |t| + |u| + |v| \tag{5}$$

From Eqs. (1)–(3), the equation obtained after applying the $l_t^+ s_u^+ r_v^+$ path pattern transformation to the normalized initial position is:

$$\begin{aligned} u\cos t &= x_q + \sin\theta_q - 2\sin t \\ u\sin t &= y_q - \cos\theta_q + 2\cos t - 1 \end{aligned} \tag{6}$$

Let (u_1, t_1) be

$$(u_1, t_1) = R(x + \sin\theta_q, y - 1 - \cos\theta_q) \tag{7}$$

If $u_1^2 < 4$, then $l_{l_t^+ s_u^+ r_v^+} = \infty$; otherwise

$$\begin{cases} u = \sqrt{u_1^2 - 4}, (T, \kappa) = R\,(u, 2) \\ t = M\,(t_1 + \kappa)\,, v = M\left(t - \theta_q\right) \end{cases} \tag{8}$$

The path length is

$$l_{l_t^+ s_u^+ r_v^+} = |t| + |u| + |v| \tag{9}$$

Since the $r_t^+ s_u^+ l_v^+$ path pattern is the reflection transformation of the $l_t^+ s_u^+ r_v^+$ path pattern, the related corresponding mathematical formulation can be systematically derived from Eqs. (6)–(9). Collision analysis for the three forward-moving path patterns identifies two potential collision scenarios during forward maneuvers.

Case I: As the vehicle approaches the shift point, both the $l_t^+ s_u^+ l_v^+$ and $r_t^+ s_u^+ l_v^+$ path patterns execute the final left-forward steering command. This results in an overall leftward and forward motion of the vehicle. When the center of the rear axle (indicated by the blue circle) reaches the transition point (indicated by the green circle), a potential collision between the left front vertex of the vehicle and the road boundary should be considered, as illustrated in Figure 3(a).

Case II: During the initial movement phase of the vehicle, the $l_t^+ s_u^+ r_v^+$ and $l_t^+ s_u^+ l_v^+$ path patterns first proceed diagonally forward to the left, while maintaining a relatively safe distance from the parking space. For the $r_t^+ s_u^+ l_v^+$ path pattern, the vehicle first moves diagonally forward to the right and then travels straight, as illustrated in Figure 3(b). In this process, the right vertex D of the vehicle is the closest to the parking space among all fixed points, which introduces a risk of collision with adjacent parked vehicles or obstacles (marked by red circles). After completing the right-turn segment and straight segment, the vehicle moves toward the left front. As indicated in Eq. (1), when the heading angle is negative, the vehicle still approaches the garage despite the left-forward motion. Therefore, the collision probability is analyzed as follows for the left-forward motion. If the path length at the end of the straight segment is short,

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

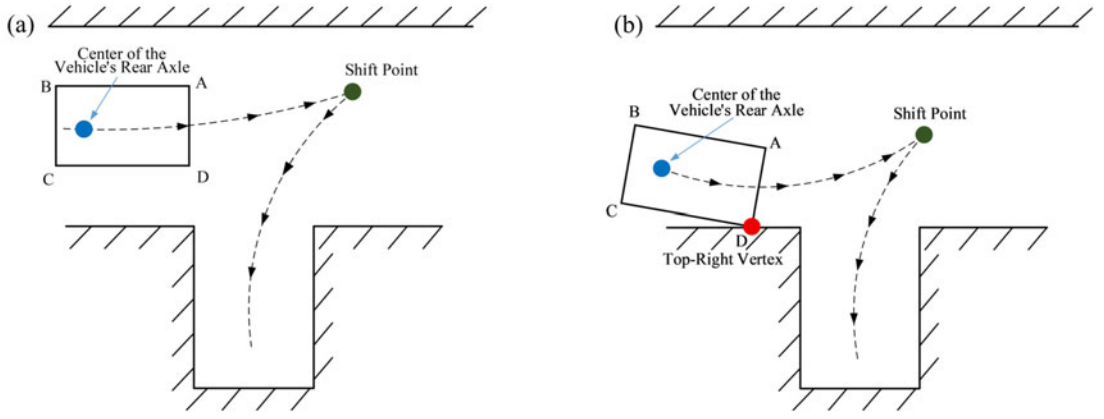


Figure 3. *Collision analysis during forward motion. (a) Case I; (b) Case II.*

the vehicle travels left-forward toward the centerline of the parking space. Even if the right-rear vertex C and right-front vertex D avoid collision in this phase, the endpoint of the forward path still cannot satisfy the spatial constraints for parking. If the vehicle has already entered the front area of the parking space by the end of the straight segment, it can safely turn left-forward inside the parking space. If the vehicle is near the right vertex of the parking space at the end of the straight segment, even if no collision occurs on the right side during the left turn, the endpoint of the forward path will be too far from the parking space with a small heading angle, which fails to meet the positional requirements for parking path planning. Therefore, for collision scenario two, only the constraint on the vehicle's right vertex D is retained after the straight segment is completed.

To prevent Case I from occurring, when the vehicle's rear axle reaches the shift point, the longitudinal coordinate of point A must satisfy $y_{\mathrm{A}} \leq h$. From the geometric relations, the coordinates of point A at this moment are given by:

$$\begin{cases} x_{\mathrm{A}} = x_z + L_f \cdot \cos\theta_z - \dfrac{W}{2} \cdot \sin\theta_z \\ y_{\mathrm{A}} = y_z + L_z \cdot \sin\theta_z + \dfrac{W}{2} \cdot \cos\theta_z \\ \theta_{\mathrm{A}} = \theta_z \end{cases} \tag{10}$$

To prevent Case II from occurring, the vertical coordinate of point D must satisfy $y_D > h_1$. Let the rear-axle position after completing the right-turn and straight-line maneuvers be denoted by (x_u, y_u, θ_u). The position of point D is obtained from the geometric relationship:

$$\begin{cases} x_{\mathrm{D}} = x_u + L_f \cdot \cos\theta_u + \dfrac{W}{2} \cdot \sin\theta_u \\ y_{\mathrm{D}} = y_u + L_f \cdot \sin\theta_u - \dfrac{W}{2} \cdot \cos\theta_u \\ \theta_{\mathrm{D}} = \theta_u \end{cases} \tag{11}$$

where (x_u, y_u, θ_u) can be obtained from the initial position (x_s, y_s, θ_s) by substituting into Eqs. (2)–(3).

2.4 Reverse parking path analysis

Similar to forward motion planning, backward path optimization in the Reeds–Shepp curve prioritizes minimal path patterns. Select path patterns with consistent direction from the nine path formats of the Reeds–Shepp curve. The qualified type is *CSC*, which includes the following backward path patterns:

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

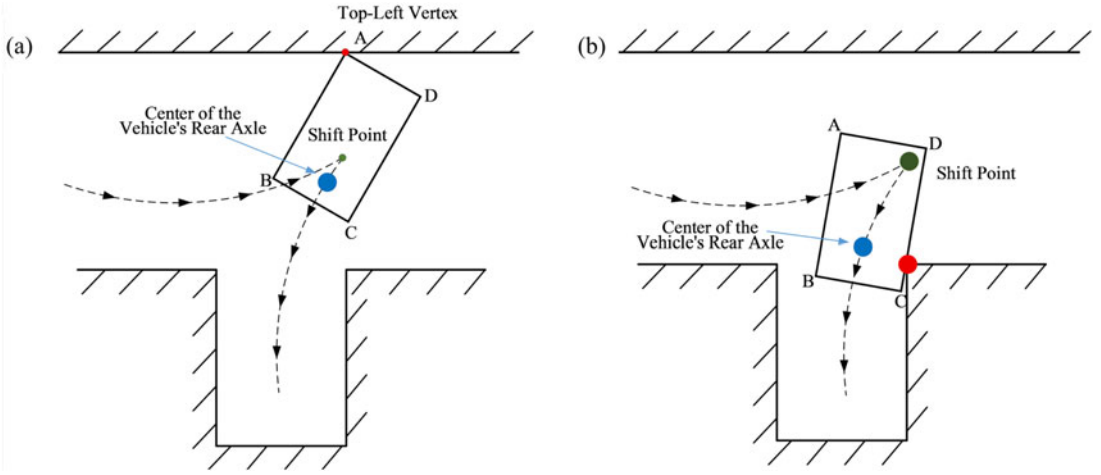


Figure 4. *Collision analysis during reverse motion. (a) Case I; (b) Case II.*

$r_t^- s_u^- r_v^-$, $r_t^- s_u^- l_v^-$, $l_t^- s_u^- r_v^-$, and $l_t^- s_u^- l_v^-$. However, in a vertical parking environment, the vertical parking destination is located to the rear right of the shift point. Consequently, turning left from the shift point is impractical. The selected patterns suitable for the backward movement of the parked vehicle are $r_t^- s_u^- r_v^-$ and $r_t^- s_u^- l_v^-$. Due to the symmetry between the $r_t^- s_u^- r_v^-$ and $l_t^+ s_u^+ l_v^+$ path patterns under time-flip (i.e., reversing the travel direction) and reflection transformation (i.e., swapping left and right turns), the path length of $r_t^- s_u^- r_v^-$ can be calculated using Eqs. (4), (5). Similarly, the $r_t^- s_u^- l_v^-$ path pattern is the time-flip and reflection transformation of the $l_t^+ s_u^+ r_v^+$ path pattern, and its path length is obtained from Eqs. (6)–(9). Collision analysis for these two backward-moving path patterns reveals two potential collision scenarios during reverse maneuvers:

Case I: In the selected reverse path pattern, the initial segment consists of backward right turning. During this maneuver, the left front corner of the vehicle is closest to the road boundary. Therefore, collision with the road boundary during the right-turn reverse maneuver must be taken into account (marked by the red circle), as illustrated in Figure 4(a).

Case II: During the parking maneuver, potential collisions between the vehicle and both sides of the parking space must be taken into account. When the shift point is positioned near the center of the parking space, the vehicle has a shorter distance available for pose adjustment in the forward segment, resulting in a smaller variation of the heading angle. Even if the vehicle body does not collide with the left side of the parking space, the limited clearance between the parking space and the road boundary prevents the vehicle from completing the parking maneuver at the minimum turning radius from this position. Therefore, in practical path planning, the shift point is preferably located near the right corner of the parking space. Therefore, for the second collision scenario when entering the garage, only the collision constraint between the vehicle's right side and the garage's right corner is considered, as illustrated in Figure 4(b).

To prevent Case I collisions during backward-right maneuvering of both $r_t^- s_u^- r_v^-$ and $r_t^- s_u^- l_v^-$ path patterns, the longitudinal coordinate of point A must satisfy $y_A \leq h$. Define the rear axle center pose as (x_r, y_r, θ_r). The angular parameter t of the arc is uniformly discretized with a sampling interval of 0.2 radians. Since the starting position of the backward path is the shift point, the coordinates of each sampled rear axle position are computed by substituting both the shift point location and the sampled arc's radian measure into Eq. (2). These coordinates are then input into Eq. (10) to derive the corresponding point A coordinates. To avoid collisions between point A and the road boundary, $y_A \leq h$ is satisfied for each sampling point.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

To prevent Case II collisions during backward-right maneuvering of $r_t^- s_u^- r_v^-$ and $r_t^- s_u^- l_v^-$ path patterns, the vehicle's right-side contour must remain above the right garage corner. This condition is enforced by requiring the line passing through points C and D to be positioned above the garage corner. The geometric relationships between points C and D and the rear axle center are as follows:

$$X_{\mathrm{C}} \begin{cases} x_{\mathrm{C}} = x_r - L_r \cdot \cos\theta_r + \dfrac{W}{2} \cdot \sin\theta_r \\ y_{\mathrm{C}} = y_r - L_r \cdot \sin\theta_r - \dfrac{W}{2} \cdot \cos\theta_r \\ \theta_{\mathrm{C}} = \theta_r \end{cases}, \quad X_{\mathrm{D}} \begin{cases} x_{\mathrm{D}} = x_r + L_f \cdot \cos\theta_r + \dfrac{W}{2} \cdot \sin\theta_r \\ y_{\mathrm{D}} = y_r + L_f \cdot \sin\theta_r - \dfrac{W}{2} \cdot \cos\theta_r \\ \theta_{\mathrm{D}} = \theta_r \end{cases} \tag{12}$$

The equation of the line passing through points C and D is derived from their coordinates as:

$$ax + by + c = 0 \tag{13}$$

where $a = y_{\mathrm{D}} - y_{\mathrm{C}}/x_{\mathrm{D}} - x_{\mathrm{C}}$, $b = -1$, and $c = y_{\mathrm{D}} - a \cdot x_{\mathrm{D}}$. Define the parking space's right-upper position as (x_j, y_j). The collision constraint between the vehicle edge CD and this corner is formulated as:

$$ax_j + by_j + c < 0 \tag{14}$$

During the backward-right maneuver from the shift point, it is essential to prevent a collision between the right side of the vehicle and the right garage corner. To achieve this, the arc angle t is uniformly sampled at 0.2-radian intervals. For each sampling point, the coordinates of the rear axle center are calculated by substituting the shift point location and sampled angles into Eq. (2). Subsequently, the coordinates of points C and D are obtained via Eqs. (12), (13), and the line passing through these two points is then determined. To ensure no collision occurs, this line must remain above the right garage corner at every sampling point.

Based on the selected path patterns of the Reeds–Shepp curve during the backward maneuver, the vehicle follows a straight line after completing the backward-right motion from the shift point. To ensure accurate garage entry, constraints are imposed on the vehicle state at the conclusion of this motion. Specifically, let the lateral deviation relative to the target position be defined as e_l the heading angle deviation as e_θ, and the overall entry deviation as E.

$$E = w_1 \cdot e_l + w_2 \cdot e_\theta \tag{15}$$

where w_1 and w_2 are the weight parameters that balance the lateral deviation and the heading angle deviation in the parking process.

3. Finding shift points and nonlinear optimization of paths

3.1 Optimal shift point search based on improved particle swarm algorithm

The Particle Swarm Optimization (PSO) algorithm [26] is a stochastic sample-based optimization method proposed by Kennedy. Its optimization process is similar to the way a flock of birds forages for food. Each particle has two attributes: position X and velocity v. In the optimization iteration, the positions of all particles are adjusted based on the individual best positions from the previous iteration and the current global best solution shared among all particles. The adjustment is formulated as follows:

$$\begin{cases} v_{i+1} = \omega v_i + c_1 \times r_1 \, (p_{best} - X_i) + c_2 \times r_2 (g_{best} - X_i) \\ X_{i+1} = X_i + v_i \end{cases} \tag{16}$$

where ω is the inertia weight, c_1 and c_2 represent the individual learning factor and social learning factor, respectively, p_{best} and g_{best} represent the individual optimal position and global optimal position, respectively, r_1 and r_2 are random numbers uniformly distributed within the interval [0, 1]. In the particle swarm optimization algorithm, these parameters collectively influence the optimization outcome.

The position of the shift point governs the length of the Reeds–Shepp curve planning path and the total deviation relative to the target point when entering the garage. Accordingly, the coordinates of

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

the shift point are adopted as the position information of the particle here. To ensure that the vehicle achieves an appropriate pose after entering the garage, it is essential to constrain the pose information upon completion of the right rearward movement from the shift point. Consequently, the total deviation between the position upon completing the backward-right movement and the target position, combined with the logarithmic weighted sum of the total lengths of the two segments, is used as the fitness function. Define l_q as the length of the parking track for forward movement and l_h as the length of the parking track for backward movement. Due to the two segments of path in the planning of a variety of different path patterns of the Reeds–Shepp curve, to ensure that the parking path in the garage provides better position information while minimizing path length, the length of the path is used as a criterion for forward movement, and the total deviation is used as a criterion for backward movement. The formula is:

$$\begin{cases} l_q = \min\left(l_{lsr}, l_{lsl}, l_{rsl}\right) \\ E_{\min} = \min\left(E_{rsl}, E_{rsr}\right) \\ l_h = \begin{cases} l_{rsl}, E_{\min} = E_{rsl} \\ l_{rsr}, E_{\min} = E_{rsr} \end{cases} \end{cases} \tag{17}$$

Due to the influence of lane width, the range of heading angle at the shift point is taken as $[0°, 45°]$, with uniform sampling every 5°. The heading angle with the minimum fitness value is chosen as the heading angle for that position. Since particles in the particle swarm optimization algorithm iterate as random samples, when a particle's position does not satisfy the forward movement constraints, the path length for the first segment should be set to a large value. When a particle's position does not satisfy the backward movement constraints, both the path length and deviation E for the second segment should be set to large values. Using parameter λ as a larger value, it indicates that the fitness at this location is poor. The target position is defined as a point on the vertical centerline of the parking space, aligned with the parking space's orientation. The fitness function is:

$$fitness = \begin{cases} w_3 \cdot E_{\min} + w_4 \cdot \log\left(l_q + l_h\right), \text{Constraint satisfaction} \\ w_3 \cdot E_{\min} + w_4 \cdot \log\left(\lambda + l_h\right), \text{Constraints not satisfied during forward pass} \\ w_3 \cdot \lambda + w_4 \cdot \log\left(l_q + \lambda\right), \text{Constraint when not satisfied during backtracking} \\ w_3 \cdot \lambda + w_4 \cdot \log\left(\lambda + \lambda\right), \text{None of the constraints are satisfied} \end{cases} \tag{18}$$

where w_3 and w_4 are the weight parameters, which are responsible for balancing the path length and the deviation associated with garage entry. Equations (17) and (18) demonstrate that different Reeds–Shepp combinations yield distinct performance in terms of path length and entry deviation. Accordingly, the process of optimizing the fitness function via the particle swarm optimization algorithm can be interpreted as determining the optimal shift point location and selecting the best-performing Reeds-Shepp curve type from the candidate shift point solutions. For each particle, the pseudo-code for the fitness function definition is given in Table I.

According to Eq. (16), the inertia weight ω, individual learning factor c_1, and social learning factor c_2 collectively govern the motion state of the particles. Geng [27] proposed a Linear Decreasing Weight Particle Swarm Optimization (LDWPSO) algorithm, which causes the inertia weight ω linearly decreases with each iteration, ensuring the dominance of population learning in the later stages. Van [28] proposed a Fermat-Weber Location Particle Swarm Optimization (FWL-PSO) algorithm, replacing the traditional optimal population location with the Fermat-Weber location to guide particles toward exploring more promising regions, thereby enhancing the algorithm's convergence. However, in more complex optimization problems, merely adjusting the inertial weight ω or altering calculation method for the global optimum position still leads to issues such as particles getting stuck in local optima, insufficient search capability, and premature convergence. To address this issue, this paper proposes an improved particle swarm optimization algorithm based on Bézier curves and nonlinear functions (named BCNPSO), which optimizes the algorithm from two aspects.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

Table I. *Fitness function of particle swarm algorithm.*

Algorithm1. Fitness function of particle swarm algorithm

Input X_z

Output $fitness, \theta_{bast}$

Initialize $X_{start}, X_{end}, fitness_{min}$
for $\theta = 0° to 45° do$
  $t_{lsl}u_{lsl}v_{lsl} = lsl(X_s, X_z)$
  if lsl satisfy constraint 1 and constraint 2 during forward movement
    $l_{lsl} = |t| + |u| + |v|$
  else
    $l_{lsl} = \lambda$
  $t_{lsr}u_{lsr}v_{lsr} = lsr(X_s, X_z)$, $t_{rsl1}u_{rsl1}v_{rsl1} = rsl1(X_s, X_z)$
  l_{lsr}, l_{rsl1} follows the same criteria
  $t_{rsl2}u_{rsl2}v_{rsl2} = rsl2(X_z, X_g)$
  if $= rsl2$ satisfy constraint 1 and constraint 2 during reversing
    $l_{rsl2} = |t| + |u| + |v|$
    $deviation_{rsl} = x_deviation_{rsl} + \theta_deviation_{rsl}$
  else
    $l_{rsl2} = \lambda, deviation_{rsl} = \lambda$
  $t_{rsr}u_{rsr}v_{rsr} = rsr(X_z, X_g)$
  $l_{rsr}, deviation_{rsr}$ follows the same criteria
  if $deviation_{rsl} > deviation_{rsr}$
    $fitness = w_3 \cdot deviation_{rsr} + w_4 \cdot \log(\min(l_{lsl}, l_{lsr}, l_{rsl1}) + l_{rsr})$
  else
    $fitness = w_3 \cdot deviation_{rsl} + w_4 \cdot \log(\min(l_{lsl}, l_{lsr}, l_{rsl1}) + l_{rsl2})$
  if $fitness < fitness_{\min}$
    $fitness_{\min} = fitness$
    $\theta_{best} = \theta$
  end

(1) Drawing on the concept of the Quadratic Bézier Curve, dynamically adjust the inertia weight and learning factors of the particle swarm optimization algorithm at different iteration periods.

(2) Introduce a speed adjustment factor α to reflect the degree of impact of the velocity term during the iteration process. A nonlinear function is designed to dynamically adjust the speed adjustment factor at different iteration periods.

Bezier curves exhibit different shapes based on changes in the positions of their control points. The formula for an nth-order Bezier curve is [29].

$$\vec{B}(m) = \sum_{i=0}^{n} \binom{n}{i} \vec{P}_m (1-m)^{n-i} m^i, m \in [0, 1] \tag{19}$$

where $\vec{B}(m) = [x(m), y(m)]^T$, $\vec{P}_m = [x_m, y_m]^T$. x_m, y_m are the coordinates of point i.

Using the properties of Quadratic Bézier curves, the control point coordinates are defined as P0, P1, and P2, as shown in Figure 5(a). Analysis reveals that as the horizontal coordinates increase, the vertical coordinates of the points T and P also increase, while that of point H decreases. The horizontal coordinates of the Quadratic Bézier Curve serve as a mapping to the number of iterations k, and the longitudinal coordinates of points T, P and H correspond to the values of social learning factor c_1, individual learning factor c_2, and inertial weight ω within the range [0, 1]. In the initial phase, larger inertia weights enhance the particles' global search capability [30]. As iterations progress, inertia weights

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

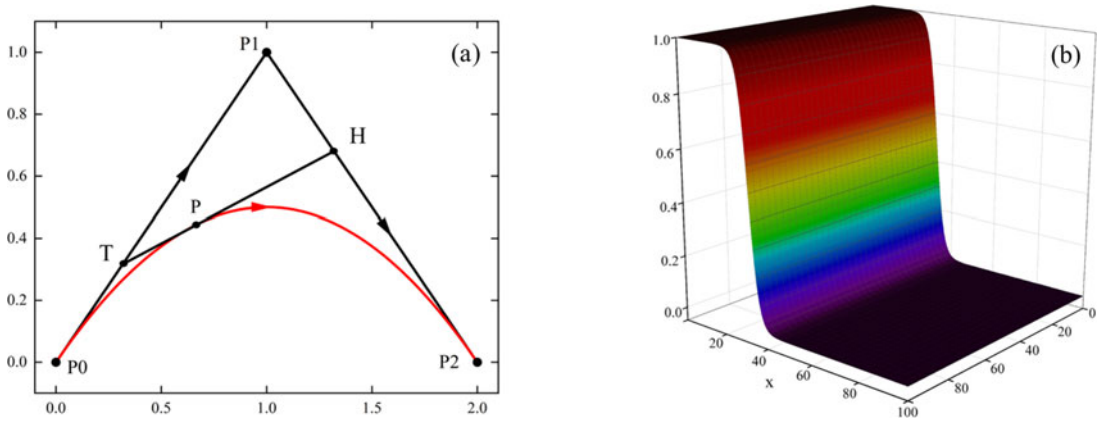


Figure 5. *Improved particle swarm optimization process. (a) Quadratic Bézier curve; (b) nonlinear function graph.*

gradually decrease while both individual learning and social learning factors increase simultaneously, helping particles escape the constraints of local optima. In the later stages of iteration, enhancing the social learning factor will be more conducive to particles converging toward the population's optimal position.

According to the parameter settings in references [31–35], the inertia weight is set within [0.4, 0.9], while the cognitive and social learning factors are generally within [1, 2]. The three control points of the Bézier curve, each corresponding to one of these parameters, are mapped from the normalized interval [0, 1] to their respective parameter ranges using the following formula:

$$\begin{cases} c_2(k) = 1 + \mathrm{T}(k) \\ \omega(k) = 0.4 + 0.5\mathrm{H}(k) \\ c_1(k) = 1 + \mathrm{P}(k) \end{cases} \tag{20}$$

Eq. (16) indicates that the velocity term affects the position of particles at the next moment. A velocity adjustment factor is introduced here to account for the influence of the velocity term during iteration. Therefore, the particle position update formula becomes:

$$X_{i+1} = X_i + \alpha v_i \tag{21}$$

To enhance the global search capability of particles in the early stages of iteration and the local search capability in the later stages, a nonlinear function is proposed to dynamically adjust the velocity adjustment factor. This nonlinear function is defined as:

$$b(k) = \eta \frac{1}{\sigma + e^{\beta(k-\mu)}} \tag{22}$$

where η is the gain factor, which determines the maximum attainable range of $b(k)$. σ is the offset adjustment factor, which affects the function's starting value and changes the position of the function's shape on the vertical axis. β is the growth factor, and it controls the steepness of the function's variation with k. μ is the translation factor, and it affects the dependence of the function value on the independent variable. As illustrated in Eq. (22) and Figure 5(b), the function value b decreases with increasing k, where the number of iterations and the speed adjustment factor are normalized to the ranges of [0, 100] and [0, 1], respectively. Consequently, at the beginning of iterations, a larger velocity adjustment factor enhances the particle's global search capability during position updates. As iterations progress, the velocity adjustment factor gradually decreases to strengthen the particle's local search capability. During later iterations, a smaller velocity adjustment factor facilitates finer-grained searches as particles approach the population's optimal solution, thereby ensuring the quality of the population's optimal

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

solution. Define the range of the speed adjustment factor as [0.5, 1]. The function value of the nonlinear equation maps from [0, 1] to this interval.

$$\alpha\ (k) = 0.5 + 0.5b\ (k) \tag{23}$$

The improved particle swarm algorithm iteration formula is as follows:

$$\begin{cases} v_{i+1} = \omega\ (k) \times v_i + c_1\ (k) \times r_1\ (p_{best} - X_i) + c_2\ (k) \times r_2(g_{best} - X_i) \\ X_{i+1} = X_i + \alpha\ (k)\ v_i \end{cases} \tag{24}$$

3.2 Nonlinear optimization of parking path

The optimal shift point is obtained using an improved particle swarm algorithm, and the path planning for vertical parking is completed using two Reeds–Shepp curves. Since the Reeds–Shepp curve is a path derived based on the vehicle's minimum turning radius, it satisfies the nonholonomic constraints of the vehicle. However, there is a sudden change in curvature at the connection between the arc and the straight line. As indicated by Eq. (25) of Ackermann's steering formula, when there is a discontinuity in curvature, the vehicle must stop to allow the front wheels to reach the desired angle before parking. Therefore, the path needs to be smoothed.

$$\delta = \arctan\ (L \cdot \kappa) \tag{25}$$

After obtaining the initial path, the front and rear segments are discretized separately using equidistant uniform sampling with a sampling step of 0.05 m. To minimize deviation from the original path while ensuring curvature continuity, the optimization objective is defined as the weighted sum of three factors: the coordinate offset between the smoothed path and the original path, the curvature change between adjacent points, and the total path curvature value. Constraints are set for the initial position, target position, and maximum curvature of path points. The objective function is expressed as follows.

$$\begin{aligned} &\min \textstyle\sum_{i=0}^{n} k_1\ (x_i - \tilde{x}_i)^2 + k_2\ (y_i - \tilde{y}_i)^2 + k_3{\kappa_i}^2 + \sum_{i=0}^{n-1} k_4\ (\kappa_{i+1} - \kappa_i)^2 \\ &s.t. \qquad X_0 = \tilde{X}_0 \\ &\qquad\quad X_n = \tilde{X}_n \\ &\qquad\quad |\kappa_i| < \kappa_{\max} i = 0, 1, \ldots n \end{aligned} \tag{26}$$

where k_1, k_2, k_3, and k_4 are weight parameters, which are used to balance the contributions of path deviation, curvature magnitude, and curvature variation. $(\tilde{x}_i, \tilde{y}_i)$ represents the original path point information, and κ_i is the curvature at point i, X_0 and X_n represent the initial position and target position, respectively. The poses of the initial and target configurations, together with curvature constraints, are incorporated as penalty terms into the objective function, thereby converting the original nonlinear constrained problem into an unconstrained one. The resulting objective function is given as follows:

$$\begin{aligned} &\min \textstyle\sum_{i=0}^{n} k_1\ (x_i - \tilde{x}_i)^2 + k_2\ (y_i - \tilde{y}_i)^2 + k_3{\kappa_i}^2 + \sum_{i=0}^{n-1} k_4\ (\kappa_{i+1} - \kappa_i)^2 \\ &+\mu_1 \left(X_0 - \tilde{X}_0\right)^2 + \mu_2 \left(X_n - \tilde{X}_n\right)^2 + \textstyle\sum_{i=0}^{n} \mu_3\ (\kappa_i - \kappa_{\max})^2 \end{aligned} \tag{27}$$

where μ_1, μ_2, and μ_3 are penalty factors, these parameters and constraints are used to penalize deviations when the start and end points do not meet requirements, and when curvature does not satisfy the constraints. The standard formula for calculating curvature is

$$\kappa_i = \frac{|\dot{x}_i \ddot{y}_i - \dot{y}_i \ddot{x}_i|}{\left((\dot{x}_i)^2 + (\dot{y}_i)^2\right)^{3/2}} \tag{28}$$

Regarding the curvature calculation for discrete points, the method described in reference [36] is employed. Let (x_0, y_0), (x_1, y_1), (x_3, y_3),. . ., (x_n, y_n) denote the intermediate path points. For any point

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

(x_i, y_i) other than the start and end points, the curvature formula for this point is

$$\begin{cases} \dot{x}_i = \dfrac{x_{i+1} - x_{i-1}}{2h}, \dot{y}_i = \dfrac{y_{i+1} - y_{i-1}}{2h} \\ \ddot{x}_i = \dfrac{x_{i-1} - 2x_i + x_{i+1}}{h^2}, \ddot{y}_i = \dfrac{y_{i-1} - 2y_i + y_{i+1}}{h^2} \end{cases} \tag{29}$$

where h corresponds to the sampling step size, set to 0.05 m here. For the starting point curvature, the steering angle is assumed to be zero at departure, implying that the curvature at the starting point is zero. For calculating the curvature at the endpoint, a third-order backward difference method is employed. Let the coordinates of the last four points be (x_{n-3}, y_{n-3}), (x_{n-2}, y_{n-2}), (x_{n-1}, y_{n-1}), and (x_n, y_n). The curvature formula for the endpoint is

$$\begin{cases} \dot{x}_n = \dfrac{-2x_{n-3} + 9x_{n-2} - 18x_{n-1} + 11x_n}{6h}, \dot{y}_n = \dfrac{-2y_{n-3} + 9y_{n-2} - 18x_{n-1} + 11y_n}{6h} \\ \ddot{x}_n = \dfrac{-x_{n-3} + 4x_{n-2} - 5x_{n-1} + 2x_n}{h^2}, \ddot{y}_n = \dfrac{-y_{n-3} + 4y_{n-2} - 5y_{n-1} + 2y_n}{h^2} \end{cases} \tag{30}$$

The curvature at each discrete point is obtained by substituting Eqs. (28) and (29) into Eq. (27). As indicated by Eqs. (28)–(30), the curvature at a discrete point X_i depends on its neighboring points X_{i-1} and X_{i+1}; consequently, computing the curvature at X_i requires evaluating the partial derivatives with respect to the positions of these two adjacent points, and extending this computation to all discrete points along the reference trajectory results in a complex process. Unlike the gradient-based method employed in ref. [37], The Nelder-Mead simplex method [38] is a gradient-free optimization algorithm whose solution logic does not rely on gradients or partial derivatives of the objective function; instead, it iteratively updates the search direction through four operations—reflection, expansion, contraction, and compression—by comparing function values at the simplex vertices. This method is well-suited for unconstrained optimization problems with penalty terms, making it a viable choice for the curvature smoothing problem addressed in this work.

3.3 Complexity analysis

To evaluate the computational efficiency of the BCNPSO Reeds–Shepp path-planning method proposed in this paper, a systematic analysis of its time complexity is performed using Big-O notation. The algorithm consists of three serially connected modules: the BCNPSO optimization module, the Reeds–Shepp curve generation module, and the nonlinear path optimization module. The overall time complexity is determined by the highest-order term among these three components. The complexity analysis for each module is presented below.

The time complexity of the BCNPSO algorithm is $O(N \times k \times C)$, where k is the maximum number of iterations and N is the population size, C denotes the computational cost of evaluating the fitness for a single particle. Since the fitness evaluation involves only constant-time operations (such as checking geometric constraints, calculating path length, and assessing deviations), the cost per particle is constant, i.e., $C = O(1)$. Therefore, the overall time complexity reduces to $O(N \times k)$.

The Reeds–Shepp curve generation employs a fixed-geometry method. Given a start and an end point, it selects the shortest path from 48 predetermined combinations. Due to its predefined maximum curvature and a fixed upper limit on the number of path segments, each candidate path can be computed in constant time, resulting in a time complexity of $O(1)$. Even when the selected path is subsequently uniformly sampled for discretization, this post-processing step does not alter its constant-time property. Therefore, the overall complexity of the Reeds–Shepp module remains $O(1)$.

The nonlinear optimization module first discretizes the initial path to obtain K path points. Subsequently, the Nelder–Mead simplex method is employed to solve the optimization problem with curvature continuity as the objective. The complexity of this algorithm primarily depends on the dimension d of the optimization variables and the number of iterations M. Since each path point comprises

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

Table II. *Simulation parameters.*

Parameters	Symbol	Value
Wheelbase	L	0.3 (m)
Rear axle to front end distance	L_r	0.4 (m)
Rear axle to rear end distance	L_r	0.1 (m)
Vehicle width	W	0.35 (m)
Minimum turning radius	R	0.52 (m)
Maximum front wheel steering angle	δ	$(-30°, 30°)$
Lane width	h	0.75 (m)
Vertical parking depth	L_1	0.62 (m)
Vertical parking width	L_2	0.45 (m)
Longitudinal distance from the origin to the road boundary	h_0	0.4 (m)
Longitudinal distance from the origin to the garage	h_1	0.35 (m)

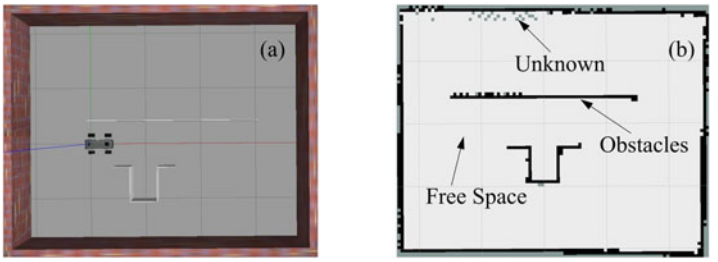


Figure 6. *Simulation environment setup. (a) Gazebo parking environment; (b) RViz 2D grid map.*

two coordinate dimensions (x and y), the total variable dimension is $d = 2K$. During each iteration, the objective function value must be computed for each of the $d + 1$ vertices of the simplex. A single objective function calculation requires traversing curvature, deviation, and other information across K path points, with a time complexity of $O(K)$. Thus, the overall complexity per iteration is $O(K \times d) = O(2K^2)$. Assuming the Nelder–Mead algorithm runs for M iterations, the overall complexity of the nonlinear optimization process can be expressed as $O(M \times 2K^2)$, which simplifies to $O(M \times K^2)$ after constant factors are simplified.

In summary, the proposed path-planning algorithm (BCNPSO Reeds–Shepp) exhibits a time complexity of $O(M \times K^2)$, where M represents the number of nonlinear optimization iterations, and K denotes the number of path discrete points. This algorithm completes computations within polynomial time, thereby meeting the engineering requirements for real-time path planning.

4. Experiment design and results discussion

4.1 Simulation environment establishment

Referencing the dimensional requirements for vertical garages in the International Building Code (IBC), a parking environment is established using the Gazebo simulation software, as shown in Figure 6(a). After creating the parking environment, map information based on a 2D grid map is generated using the Cartographer mapping algorithm [39] and displayed in RViz software, as shown in Figure 6(b). As shown in the figure, black represents the information about obstacles detected by the laser radar during mapping, light gray indicates the drivable areas, and dark gray denotes the uncertain areas.

To be consistent with actual vehicles, the maximum front wheel steering angle is limited to 30°. The specific parameters of the vehicle and the parking scene are detailed in Table II.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

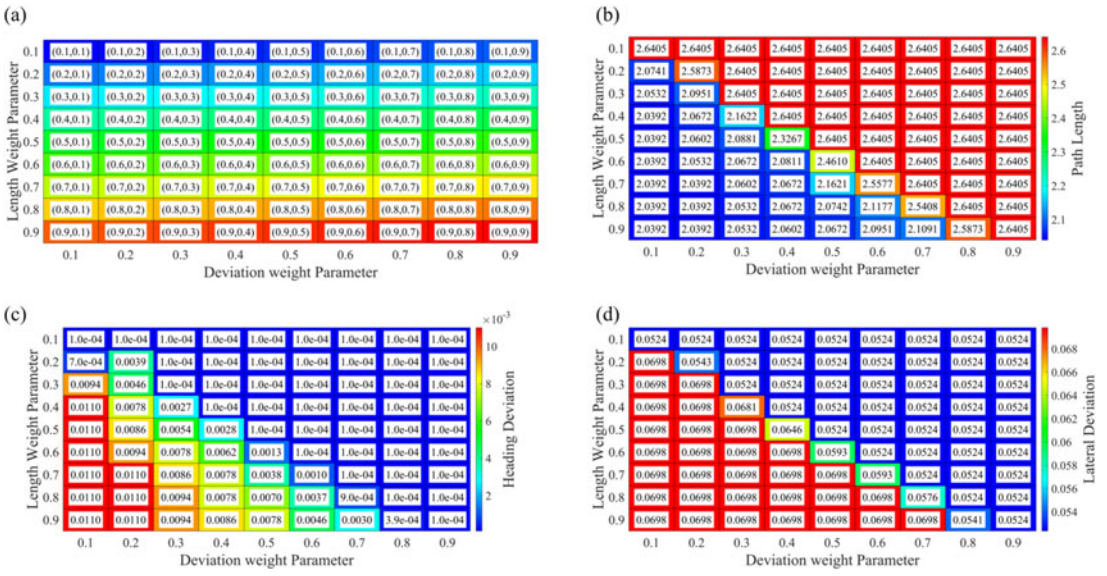


Figure 7. *Analysis of fitness functions with different weights. (a) Parameter combination; (b) average path length for different combinations; (c) average heading deviation for different combinations; (d) average lateral deviation for different combinations.*

4.2 Performance evaluation of the improved particle swarm algorithm

4.2.1 Parameter analysis

As defined in Eq. (18), the fitness function incorporates length weight parameter and deviation weight parameter, the adjustment of which shifts the optimization focus of the particle swarm optimization algorithm. To investigate the influence of these weights on algorithmic performance, 81 combinations of the two weights were evaluated, with each weight ranging from 0.1 to 0.9 in steps of 0.1 (see Figure 7a). In all experiments, the population size was set to 80, the maximum number of iterations to 100, the inertia weight to 0.65, and both the cognitive and social learning factors to 1.5. For each parameter combination, the PSO algorithm was run 30 independent times using Fitness Function 1, with an initial configuration of (0, 0, 0°) and a target of (1, −0.73, 90°). Figure 7 presents the averaged results in terms of path length, lateral deviation, and heading deviation.

Figure 7(b) illustrates the average path length under various weight combinations, while Figure 7(c) and (d) presents the average heading angle deviation and average lateral deviation, respectively. It can be observed from Figure 7(b)–(d) that a smaller length weight leads to an increase in total path length. Although the lateral and heading deviations decrease under this condition, the resulting path is suboptimal. Conversely, a smaller deviation weight reduces the total path length but increases lateral and heading deviations, compromising safety after parking. When the length-weight parameters are set to 0.4, 0.5, and 0.6, with corresponding deviation-weight parameters of 0.3, 0.4, and 0.5, the overall performance is favorable. Given that the parameter combinations (0.5, 0.4) and (0.4, 0.3) yield comparable heading angle and lateral deviations, while achieving a more pronounced reduction in path length compared to the (0.6, 0.5) combination, the length weight and deviation weight in the fitness function are set to 0.5 and 0.4, respectively.

Equation (18) reveals that when the constraint is violated, a relatively large value is assigned to parameter λ to indicate a poor fitness. Investigation of λ shows that an excessively small value may mislead the optimization direction and lead to algorithm divergence. Conversely, an overly large value may cause excessive discrepancies between fitness values in the early and late stages of iteration, hindering the analysis of optimal fitness variations. Five distinct values, i.e., [2–6], were selected as candidates for λ. Using identical parameter settings, the particle swarm optimization algorithm was run independently

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

Table III. *Parameters of the genetic algorithm and particle swarm optimization.*

Algorithm	Parameters
GA	Pc (Probability of Crossover) $=$ 0.85, Pm (Probability of Mutation) $=$ 0.25
PSO	$\omega = 0.65, c_1 = c_2 = 1.5$
LDWPSO (1.5)	$\omega = 0.9\text{-}0.4, c_1 = c_2 = 1.5$
LDWPSO (1.7)	$\omega = 0.9\text{-}0.4, c_1 = c_2 = 1.7$
FWL-PSO	$\omega = 0.9\text{-}0.4, c_1 = c_2 = 1.5$, p(Number of Elite Particles) $=$ 4, N_f(FWL Position Iteration Count) $=$ 25
BCNPSO	$\omega = \omega(k), c_1 = c_1(k), c_2 = c_2(k)$

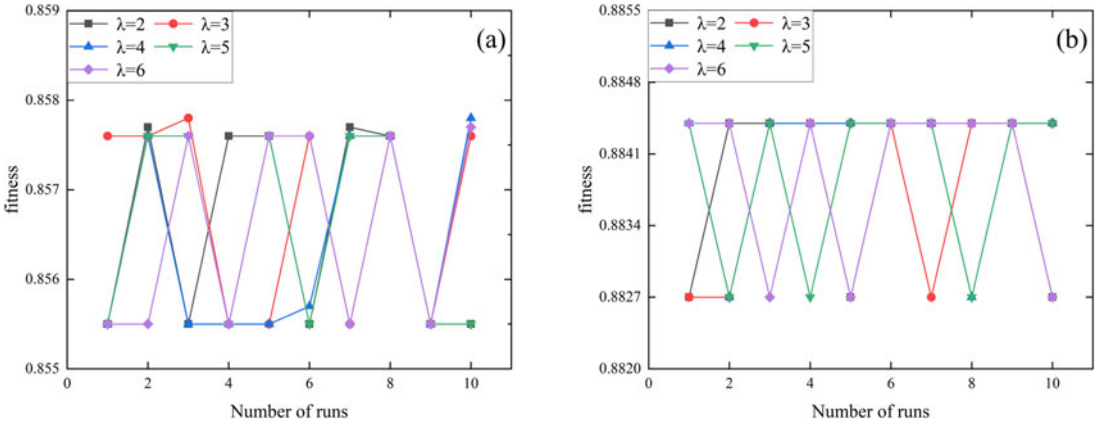


Figure 8. *Final fitness distributions for different λ values. (a) Final fitness distribution for FF1; (b) final fitness distribution for FF2.*

10 times for both Fitness Function 1 (FF1) (initial position: (0, 0, 0°), target point: (1, −0.73, 90°)) and fitness function 2 (FF2) (initial position: (0.1, 0.15, −30°), target point: (1, −0.73, 90°)). The final fitness values obtained from each run are presented in Figure 8.

Figure 8(a) shows the final fitness distribution for parameter λ at different values when using FF1. Figure 8(b) shows the final fitness distribution for parameter λ at different values when using FF2. The final fitness values for all parameter candidates, illustrated in Figure 8(a) and (b), converge to ranges of [0.855–0.858] and [0.8827–0.8848], respectively. Such convergence suggests that the particle swarm algorithm is capable of locating either the global optimum or a satisfactory local optimum when utilizing these parameter candidates. This means that choosing parameter λ at these values avoids misdirecting the optimization direction. The parameter λ is chosen to approximate five times the final fitness range, representing a poor fitness value when constraints are not satisfied. The parameter is set to 4.

4.2.2 Algorithm comparison results

To evaluate the performance of the proposed BCNPSO algorithm in identifying optimal shift points and other optimization problems, the FF1, FF2, Sphere function, and Ackley function were used as test functions. The BCNPSO was compared with the Genetic Algorithm (GA) [40], PSO [26], LDWPSO [27], and FWL-PSO [28] algorithms. Each algorithm is set with a population size of 80 individuals and a maximum iteration count of 100. GA parameters are referenced from refs. [41–43]. The parameters for particle swarm optimization are detailed in Table III.

The optimal fitness of the five algorithm populations was recorded at each iteration, and the results are shown in Figure 9.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

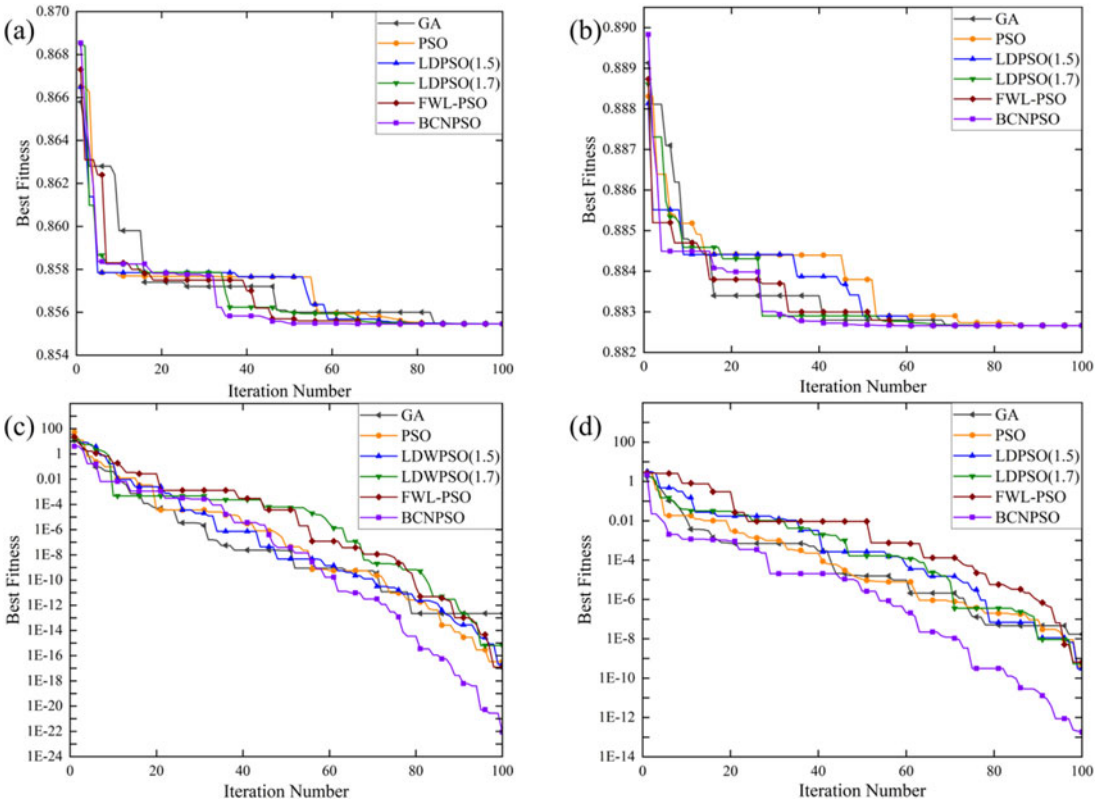


Figure 9. *Iteration process of different algorithms. (a) Iteration process of FF1; (b) Iteration process of FF2; (c) Iteration process of the Sphere function; (d) Iteration process of the Ackley function.*

As illustrated in Figure 9(a) and (b), LDWPSO (1.5), LDWPSO (1.7), and BCNPSO maintain larger inertial weights than standard PSO during the first 30 iterations, thereby strengthening global search capabilities. Meanwhile, GA identifies promising fitness positions by preserving high-fitness individuals, whereas FWL-PSO operates based on the Fermat-Weber location principle. However, between iterations 30 and 50, all five algorithms become trapped in local optima to varying degrees. Notably, the larger individual learning factor of BCNPSO allows particles to explore the search space more extensively based on their own experience, enabling effective escape from local optima. Furthermore, its rate adjustment factor remains consistently below one during this phase, prompting more refined local searches and facilitating the discovery of optimal solutions across multiple local optima. After 50 iterations, the social learning factor of BCNPSO gradually becomes dominant, guiding particles to conduct finer searches near the global optimum and thereby ensuring solution quality. Figure 9(c) and (d) further show that BCNPSO outperforms the other algorithms in optimization tasks. This advantage stems from the dynamic adjustment of three weight parameters across different iteration stages using Bézier curves: global search capability is enhanced in early iterations, while local search capability is strengthened in later iterations. By modulating the influence of velocity on particle movement, the algorithm achieves finer-grained search behavior, accelerates convergence, and improves overall solution quality.

To evaluate the robustness of BCNPSO in identifying optimal shift points and other optimization problems, FF1, FF2, sphere, and Ackley were used as test functions. Each function was executed 50 times independently under identical parameter conditions, and the results are presented in Table IV.

The experimental results show that, for test function FF1, the proposed BCNPSO algorithm achieves a standard deviation of 0.00100 after 50 iterations. Compared with the standard deviations of GA (0.001986), PSO (0.001988), LDWPSO (0.001117), and FWL-PSO (0.001166) under the same number of iterations, BCNPSO yields reductions of 49.6%, 49.7%, 10.5%, and 14.2%, respectively.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

Table IV. *Comparison of results from different algorithms.*

Test function	Algorithm	Optimal value	Worst value	Average	Standard deviation
FF1	GA	0.8555	0.8623	0.857824	0.001986
	PSO	0.8555	0.8619	0.856986	0.001988
	LDWPSO (1.5)	0.8555	0.8612	0.856814	0.001644
	LDWPSO (1.7)	0.8555	0.8589	0.856492	0.001117
	FWL-PSO	0.8555	0.8578	0.856439	0.001166
	BCNPSO	0.8555	0.8579	0.856096	0.001000
FF2	GA	0.8827	0.8876	0.885218	0.001815
	PSO	0.8827	0.8844	0.883678	0.000815
	LDWPSO (1.5)	0.8827	0.8844	0.883962	0.000739
	LDWPSO (1.7)	0.8827	0.8844	0.883922	0.000752
	FWL-PSO	0.8827	0.8844	0.883935	0.000752
	BCNPSO	0.8827	0.8844	0.883251	0.000732
Sphere	GA	2.2407×10^{-13}	2.9368×10^{-6}	1.7077×10^{-7}	4.7933×10^{-7}
	PSO	3.4307×10^{-17}	4.0202×10^{-14}	6.2364×10^{-15}	8.8343×10^{-15}
	LDWPSO (1.5)	1.1161×10^{-17}	6.5490×10^{-15}	6.8310×10^{-16}	1.0836×10^{-15}
	LDWPSO (1.7)	6.5329×10^{-16}	2.3394×10^{-13}	2.0377×10^{-14}	3.6051×10^{-14}
	FWL-PSO	7.8706×10^{-19}	5.9124×10^{-15}	4.6561×10^{-16}	1.0170×10^{-15}
	BCNPSO	8.7305×10^{-23}	5.7665×10^{-19}	6.8073×10^{-20}	1.1079×10^{-19}
Ackley	GA	1.7116×10^{-8}	4.2121×10^{-4}	3.7429×10^{-5}	9.4502×10^{-5}
	PSO	3.899×10^{-10}	2.3781×10^{-8}	6.7498×10^{-9}	4.9335×10^{-9}
	LDWPSO (1.5)	3.0362×10^{-10}	1.0136×10^{-8}	2.9708×10^{-9}	2.4605×10^{-9}
	LDWPSO (1.7)	5.3566×10^{-10}	8.7467×10^{-8}	1.9725×10^{-8}	1.6760×10^{-8}
	FWL-PSO	6.6103×10^{-10}	4.6555×10^{-8}	9.8926×10^{-9}	9.2013×10^{-9}
	BCNPSO	1.8163×10^{-13}	9.3465×10^{-11}	2.9765×10^{-11}	1.9750×10^{-11}

Furthermore, the range between the best and worst solutions obtained by BCNPSO is only 0.0024, which is smaller than that of the other four algorithms, further indicating its superior stability. When the test function was Sphere, BCNPSO achieved an average value of 6.8073×10^{-20} after 50 runs, maintaining a significant advantage over the best-performing FWL-PSO algorithm's 4.6561×10^{-16} among the other four algorithms. When the comparison function was replaced with the other two functions, the optimal value, worst value, and all four other metrics showed even better performance, fully validating its robustness.

4.3 Performance evaluation of the parking path-planning algorithm

To evaluate the proposed vertical garage parking path-planning algorithm, the initial position is set at $(0, 0, 0^\circ)$, and the target point is set at $(1, -0.73, 90^\circ)$. Simulation results are presented in Figure 10(a), where the blue curve represents the trajectory of the vehicle's rear axle during the parking process.

Figure 10(b) illustrates the optimized trajectory obtained via nonlinear optimization, consisting of 42 discrete path points. A comparison between the optimized and unoptimized paths (Figure 10c) shows that the transition at the junction between straight segments and circular arcs is significantly smoother after optimization. As indicated in Eq. (25), curvature discontinuities lead to abrupt changes in the reference steering angle, with the magnitude of these changes reflecting the degree of curvature variation. Figure 10(d) illustrates the reference steering angle profiles before and after optimization. The nonlinear optimization yields a considerably smoother reference steering angle, thereby reducing the excitation imposed on the steering actuator.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

Table V. *Path information for different initial positions.*

Initial position	Shift point's position	Initial path patterns	Optimized length (m)	Time (s)
(0.1, 0.15, −30°)	(1.52, 0.091, 0°)	$lsr + rsl$	2.52003	7.391
(−0.1, 0.1, 30°)	(1.257, 0.014, 30°)	$rsl + rsl$	2.19065	7.153
(0.5, 0.2, 0°)	(1.339, 0.054, 20°)	$rsl + rsl$	1.77213	6.954
(0.3, −0.15, 0°)	(1.211, 0, 35°)	$lsl + rsl$	1.70675	6.916

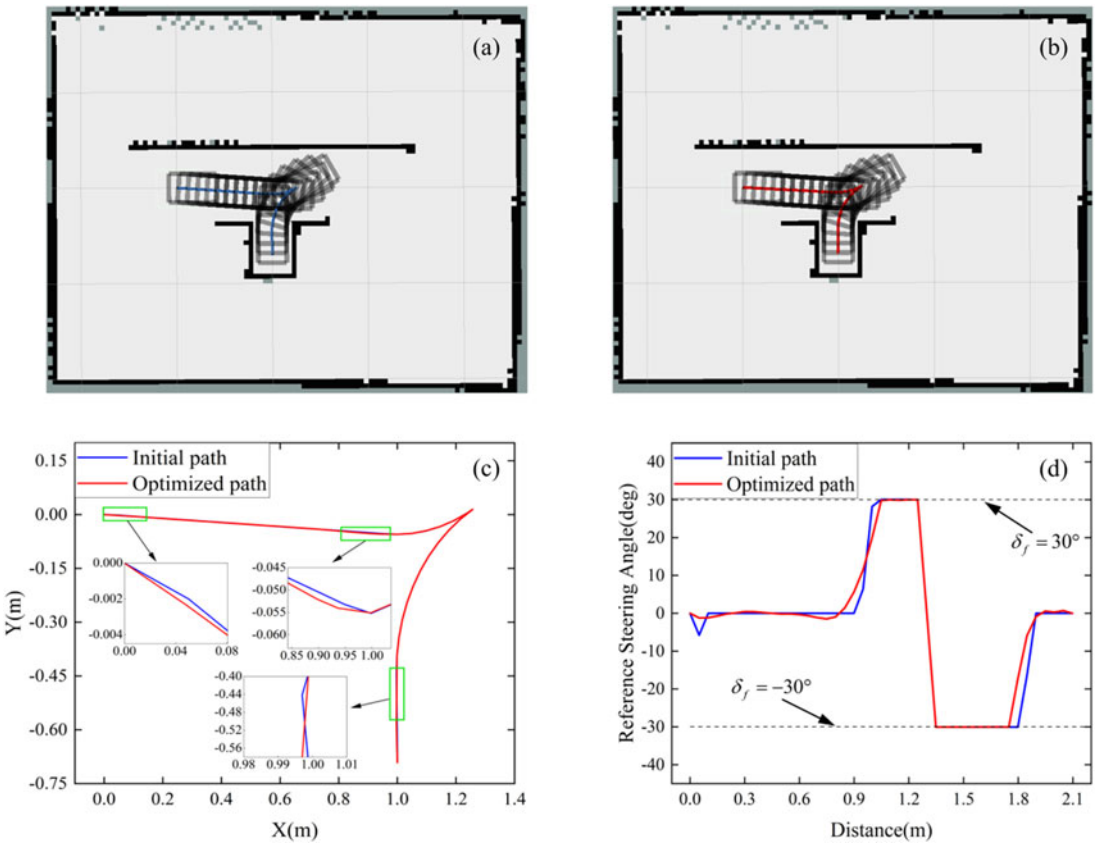


Figure 10. *Comparison of effects before and after optimization. (a) Initial path; (b) optimized path; (c) path comparison; (d) variation in equivalent front wheel steering angle.*

To evaluate the flexibility of the proposed path-planning algorithm, four initial positions were selected for parking simulation experiments. The corresponding results are presented in Figure 11(a), which demonstrates that the algorithm performs effectively across a variety of starting configurations. Figure 11(b) illustrates the variation of heading angle with path length at different initial positions. This demonstrates that the proposed algorithm plans a path with a small deviation of the heading angle from the target position after parking, even with different initial heading angles, ensuring better vehicle positioning post-parking.

Table V presents the planning results for different initial positions. Analysis indicates that the proposed BCNPSO Reeds–Shepp algorithm can generate smooth and feasible parking paths from multiple initial positions within a short timeframe.

To evaluate the path quality of the proposed BCNPSO Reeds–Shepp path-planning algorithm, comparative experiments were conducted under the same scenario with identical start and goal points. The

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

Table VI. *Comparison of path planning results using different algorithms.*

Algorithm	Maximum heading angle deviation after entering garage (rad)	Maximum lateral deviation after entering garage (m)
Hybrid A*	0.279	0.022
Bi RRT* Reeds–Shepp	0.069	0.015
IHA-PPP	0.071	0.018
BCNPSO Reeds–Shepp	0.001	0.003

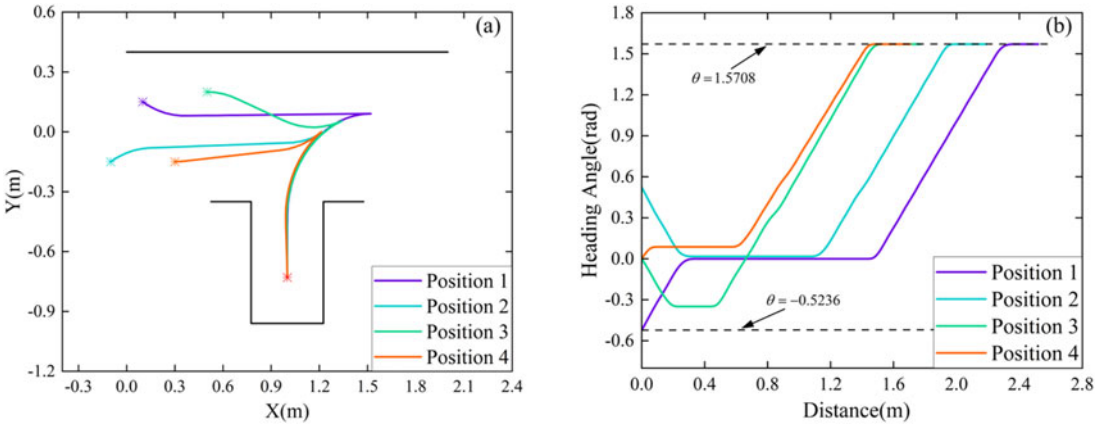


Figure 11. *Simulation results with different initial positions. (a) Paths from different initial positions;. (b) Variation in heading angle from different initial positions.*

proposed algorithm was compared with the classical Hybrid A* algorithm [44], the Bi-RRT-Reeds–Shepp algorithm [11], and the IHA-PPP algorithm [14].

Figure 12(a) compares parking paths generated by four algorithms. By identifying optimal shift points and corresponding Reeds–Shepp curve types via nonlinear optimization, the proposed BCNPSO Reeds–Shepp algorithm yields a smoother forward path than the three benchmarks. Moreover, during reverse parking, it achieves smaller parking deviation by integrating the garage entrance deviation into the fitness function. Figure 12(b) shows heading angle variations along the path. Unlike competing algorithms whose heading angles drift from the target after garage entry, the BCNPSO Reeds–Shepp algorithm reaches the target heading immediately upon entry and holds it until the destination. Figure 12(c) compares reference steering angles across methods. The proposed planner requires direction changes only at turning points, producing a smoother parking path than other algorithms.

Table VI compares the four algorithms in terms of maximum heading angle and lateral deviation after garage entry. In this comparison, the proposed algorithm reduces the maximum heading angle deviation by 0.278 rad and the maximum lateral deviation by 0.019 m relative to the Hybrid A* algorithm. Furthermore, relative to the Bi RRT* Reeds–Shepp algorithm and the IHA-PPP algorithm, our BCNPSO Reeds–Shepp algorithm achieves reductions in maximum heading deviation of 0.068 and 0.07 rad, respectively, and in maximum lateral deviation of 0.012 and 0.015 m, respectively. These results demonstrate its superior post-parking positioning accuracy and lower collision risk during the parking process.

In summary, this study proposes a vertical parking path-planning method using the BCNPSO Reeds–Shepp algorithm. Shift positions and Reeds–Shepp curve profiles are optimized via BCNPSO, followed by nonlinear optimization to ensure curvature continuity. A fitness function based on parking requirements is optimized using an improved particle swarm algorithm, whose optimality and convergence are

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

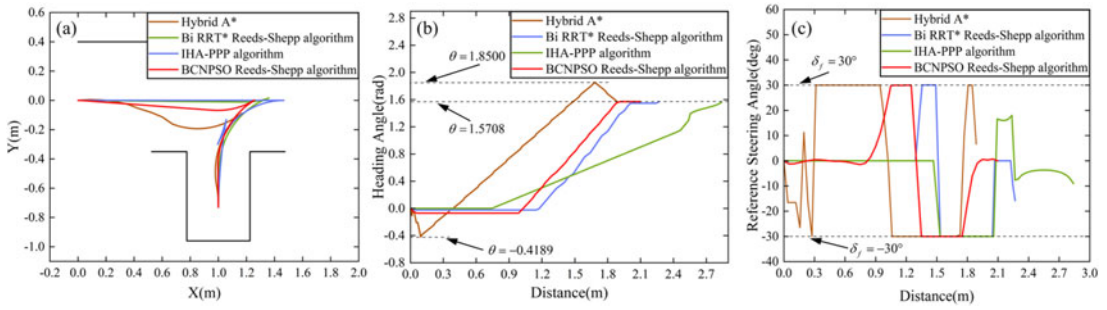


Figure 12. *Simulation results of different algorithms. (a) Parking path; (b) variation in heading angle; (c) variation in equivalent front wheel steering angle.*

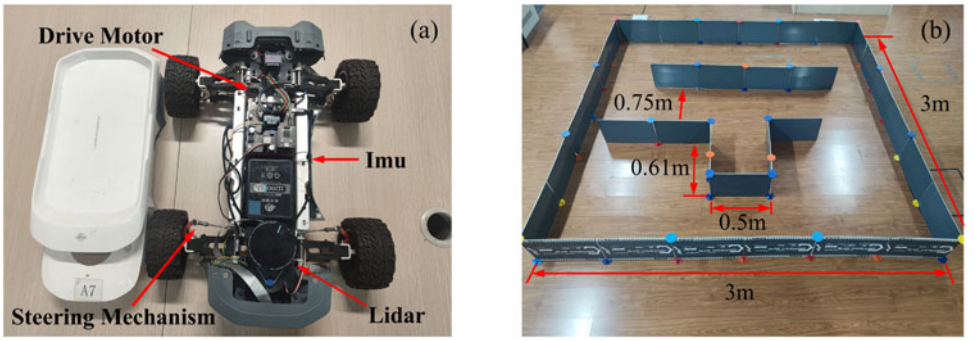


Figure 13. *Experimental platform and test field. (a) Experimental platform; (b) Test field.*

validated in Section 4.2.2. After obtaining optimal shift points, the path is generated with two Reeds–Shepp segments. Given initial and target poses, the shortest collision-free path is selected. Since each valid combination yields a unique solution, this combinatorial strategy ensures the overall path planning algorithm's optimality and convergence.

4.4 Real vehicle test

The experimental platform is based on an intelligent automated guided vehicle (AGV), as presented in Figure 13(a). The vehicle is designed with an Ackermann-type steering mechanism, in which the front wheels govern steering while the rear wheels deliver propulsion. Continuous modulation of the front-wheel steering angle enables directional control, whereas the rear wheels remain in a state of pure rolling, aligned parallel to the longitudinal axis of the chassis. The correspondence between turning radius and travel speed conforms to the Ackermann steering geometry. Furthermore, the AGV is integrated with a Lidar sensor, an inertial measurement unit (IMU), and an embedded Jetson Nano computing module, which together furnish the requisite hardware foundation for autonomous parking path planning. The mass of the vehicle is 5 kg, and all remaining key parameters are consistent with those in the simulation model. Detailed parameters are provided in Table II.

To account for potential wheel slippage during vehicle motion, the width of the vertical parking space in the simulated map was extended by 0.5 m, with all other parameters held constant. The updated layout is illustrated in Figure 13(b). Based on the data presented in Table II, the vehicle exhibits a length-to-width ratio of 1.428, whereas the parking space has a corresponding ratio of 1.24. Specifically, the vehicle length occupies 80% of the garage length, and its width accounts for 70% of the garage width. This configuration reflects a typical and challenging real-world scenario, offering a representative operational environment for evaluating autonomous parking maneuvers.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

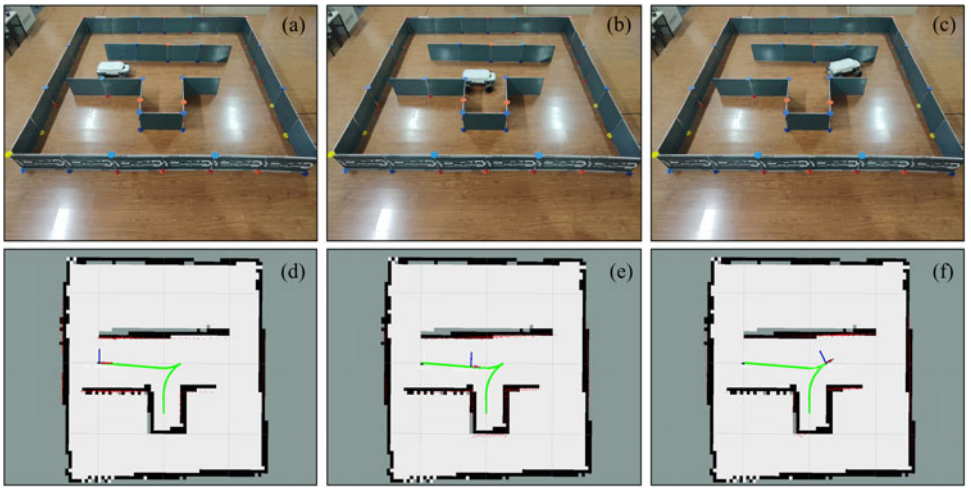


Figure 14. *Forward movement. (a) State 1 in a real-world scenario; (b) State 2 in a real-world scenario; (c) State 3 in a real-world scenario; (d) visualization of State 1; (e) visualization of State 2; (f) visualization of State 3.*

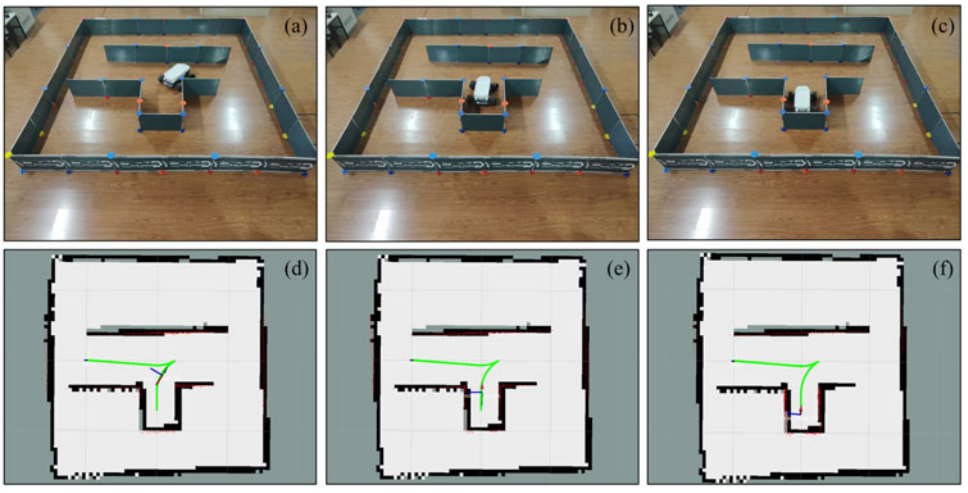


Figure 15. *Backward movement. (a) State 1 in a real-world scenario; (b) State 2 in a real-world scenario; (c) State 3 in a real-world scenario; (d) visualization of State 1; (e) visualization of state 2; (f) visualization of State 3.*

To execute the parking maneuver, the longitudinal velocity was set to 0.1 m/s for forward motion and –0.1 m/s for reverse motion. A pure pursuit controller [45] was employed for trajectory tracking. The initial position is given as (0, 0, 0°), and the target position is (1, −0.73, 90°). The parking process in the real scenario is shown in Figures 14 and 15.

As illustrated in Figure 16(a), the trajectory of the rear axle center remains within a reasonable margin of error relative to the reference path shown in Figures 13 and 14, demonstrating the successful completion of the perpendicular parking task. Figure 16(b) presents the variation in front wheel steering angle with respect to driving distance, confirming that the parking maneuver was executed within the allowable steering angle limits. Furthermore, Figure 16(c) shows the evolution of the vehicle's heading angle over distance; the heading angle remains stable after entry into the parking space, thereby validating the final orientation of the vehicle.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

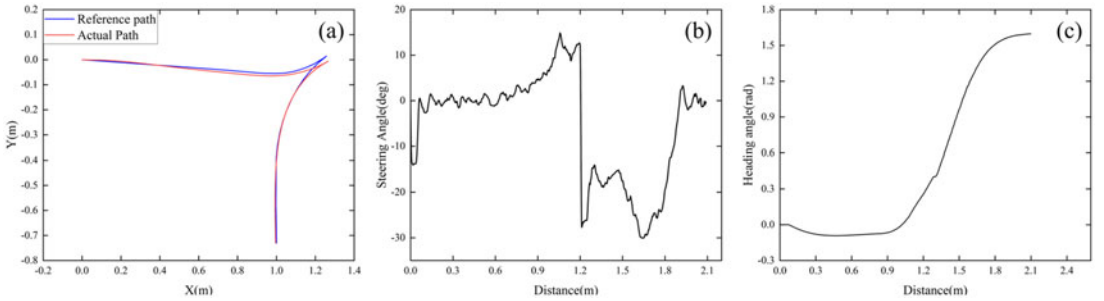


Figure 16. *Experimental results in real-world scenarios. (a) Comparison between the actual path and the reference path; (b) change in steering angle; (c) change in heading angle.*

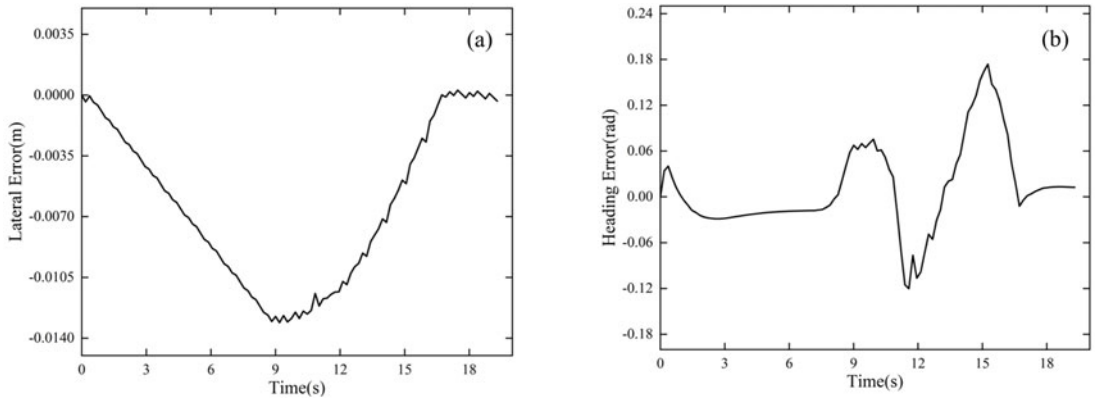


Figure 17. *Path-tracking errors in real-world scenarios. (a) Lateral error during tracking; (b) heading error during tracking.*

The lateral and heading errors during the parking maneuver are presented in Figure 17(a) and (b), respectively. The lateral tracking error peaks at 0.013 m, indicating precise path following within a tight tolerance, while the heading angle error settles at 0.012 rad at the endpoint, confirming the vehicle's ability to achieve the desired final orientation.

To evaluate the flexibility of the BCNPSO Reeds–Shepp parking path-planning algorithm, the initial pose of the vehicle was set to (0.1, 0.15, –30°), while the destination remained unchanged. The trajectory tracking results are shown in Figure 18(a). The results demonstrate that the vehicle successfully completed the parking task under this starting configuration. Figure 18(b) presents the lateral tracking error curve during parking. Lateral error analysis reveals minimal deviation throughout the process, with a maximum error of 0.011 m for this configuration. As illustrated in Figure 18(c), the heading angle error remains within an acceptable range, peaking at 0.105 rad despite a nonzero initial condition. In summary, the proposed path-planning algorithm exhibits strong adaptability and robustness to variations in initial poses, confirming its feasibility and effectiveness in parking scenarios.

5. Conclusion

To address the challenges of strict initial position constraints, insufficient clearance between the path and garage boundaries, and discontinuous curvature in traditional path planning for vertical parking scenarios, this paper presents a novel vertical parking path-planning method based on the Reeds–Shepp curve and an Improved Particle Swarm Optimization algorithm. The main conclusions are as follows:

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

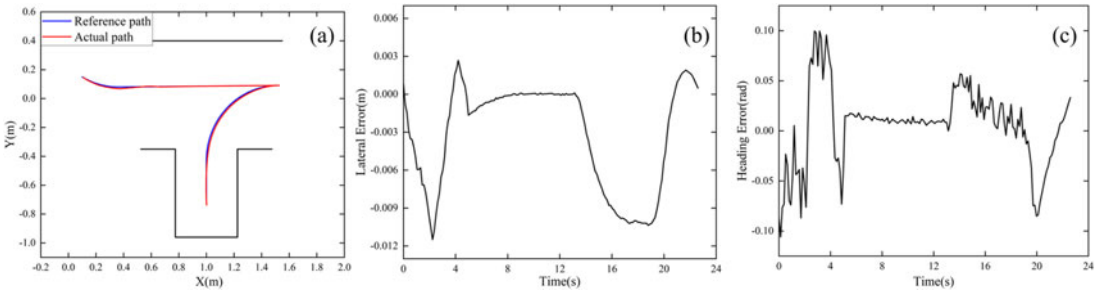


Figure 18. *Parking process with an initial pose of (0.1, 0.15, −30°). (a) Comparison between the actual path and the reference path; (b) lateral error during tracking; (c) heading error during tracking.*

(1) By segmenting the vertical parking process into two distinct phases and using the Reeds–Shepp curve to plan the path separately, the flexibility of path planning from various initial positions is enhanced. The shift point is treated as a position variable in the particle swarm algorithm, using the deviation during garage entry and path length as the fitness function. After determining the optimal shift point with the improved algorithm, the path is formed by two Reeds–Shepp curve segments, effectively addressing the proximity issue to the garage boundary upon parking completion. (2) By discretizing the paths derived from two Reeds–Shepp curves and solving a formulated constrained nonlinear optimization problem, the issue of curvature discontinuity at the junctions can be effectively addressed. (3) Through multiple experimental sets, the proposed perpendicular parking path-planning method demonstrated its ability to generate feasible paths from various initial locations. Compared to the Hybrid A* algorithm, it reduced the maximum heading angle deviation after garage entry by 0.278 rad and the maximum lateral deviation by 0.019 m. Compared to the Bi RRT* Reeds–Shepp algorithm [11] and the IHA-PPP algorithm [15], the proposed BCNPSO Reeds–Shepp algorithm also exhibited lower deviations. The feasibility of the generated paths was validated through real-vehicle experiments, offering a novel reference solution for perpendicular parking path planning.

Although the proposed algorithm has been validated in simulated parking environments, the path-planning algorithm presented in this paper relies on positioning accuracy. In future research, we will deepen our investigation into high-precision positioning technologies and try to deploy the planning algorithm in outdoor parking lots that more closely resemble real-world scenarios to further validate its practical effectiveness.

Author contributions. Guangqi Hu: proposed the research idea and designed the study. Likun Cui: funding acquisition, project administration, writing, review, and editing. Hao Wang: validation of correlation algorithms. Guangqi Hu: completed the manuscript writing. Xinle Li: reviewed and refined the paper.

Financial support. This research is supported by the Natural Science Foundation of Shaanxi Province (Grant No. 2023-JC-YB-018).

Competing interests. The authors declare no conflicts of interest exist.

References

[1] W. Wang, Y. Song, J. Zhang and H. Deng, "Automatic parking of vehicles: A review of literatures," *International Journal of Automotive Technology*. **15**(6), 967–978 (2014).

[2] Y. Wang and M. P. Cartmell, "Autonomous vehicle parallel parking design using function fitting approaches," *Robotica*. **16**(2), 159–170 (1998).

[3] R. H. Hamada and P. Sebastian, "Vision-Based Automated Parking System," **In:** *Proceedings of the 10th International Conference on Information Science*, Funchal, Portugal (2010) pp. 125–130.

[4] M. Khalid, K. Wang, N. Aslam, Y. Cao, N. Ahmad and M. K. Khan, "From smart parking towards autonomous valet parking: A survey, challenges and future works," *J Netw Comput Appl*. **175**, 102935 (2021).

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

[5] J. Tian and Q. Ye, "A review of automated parking development and motion planning research progress," *Sci Technol Eng*. **24**(21), 8825–8836 (2024, (in Chinese).
[6] K. Nesrine, N. Khraief and S. Belghith, "Reinforcement learning with modified exploration strategy for mobile robot path planning," *Robotica* **41**(9), 2688–2702 (2023).
[7] G. Gao, D. Li, K. Liu, Y. Ge and C. Song, "A study on path-planning algorithm for a multi-section continuum robot in confined multi-obstacle environments," *Robotica*.**42**(10), 3324–3347 (2024).
[8] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Am J Math*. **79**(3), 497–516 (1957).
[9] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pac J Math*. **145**(2), 367–393 (1990).
[10] M. Jiang, Y. Li, S. Zhang, S. Chen, C. Wang and M. Yang, "HOPE: A reinforcement learning-based hybrid policy path planner for diverse parking scenarios," *IEEE Trans Intell Transp Syst.* **26**(5), 6130–6141 (2025).
[11] J. H. Jhang, F. L. Lian and Y. H. Hao, "Human-like motion planning for autonomous parking based on revised bidirectional rapidly-exploring random tree* with Reeds–Shepp curve," *Asian J Control*. **23**(3), 1146–1160 (2021).
[12] H. Li, D. Zhu, M. Chen, T. Wang and H. Zhu, "An improved Reeds–Shepp and distributed auction algorithm for task allocation in multi-AUV system with both specific positional and directional requirements," *J Mar Sci Eng*.**12**(3), 486 (2024).
[13] W. Lim, J. Kim, K. Jo, "Turning Standard Line (TSL) Based Path Planning Algorithm for Narrow Parking Lots," **In:** *SAE Technical Paper 2015-01-0319* (2015).
[14] H. Guo. Intelligent vehicle perpendicular parking path planning and tracking control. **In:** *Proc Int Symp Intell Robot Syst* (2022) pp. 86–96.
[15] F. Wang and S. Gu, "Autonomous parking path planning based on improved β-spline theory," *J Jiangsu Univ Nat Sci Ed*. **43**(1), 15–23 (2022) (in Chinese).
[16] W. Nelson. Continuous-curvature paths for autonomous vehicles. **In:** *Proc IEEE Int Conf Robot Autom* (1989), pp. 1260–1264.
[17] S. Zhang, M. Simkani and M. H. Zadeh. Automatic vehicle parallel parking design using fifth degree polynomial path planning. **In:** *Proc. IEEE Veh. Technol. Conf*, 2011) pp. 1–4.
[18] Z. Zhang, S. Lu, L. Xie, H. Su, D. Li, Q. Wang and W. Xu, "A guaranteed collision-free trajectory planning method for autonomous parking," *IET Intell Transp Syst*. **15**(2), 331–343 (2021).
[19] S. M. Daniali, A. Khosravi, P. Sarhadi and F. Tavakkoli, "An automatic parking algorithm design using multi-objective particle swarm optimization," *IEEE Access*. **11**, 49611–49624 (2023).
[20] K. Sim, J. Kim and J. Gim, "A kinematics-constrained grid-based path planning algorithm for autonomous parking," *Appl Sci*. **15**(20), 11138 (2025).
[21] L. H. Guo, "Automatic parking path planning based on ant colony optimization and the grid method," *J Sensors*. **2021**, 8592558 (2021).
[22] C. Piao, J. Zhang, K. H. Chang, Y. Li and M. Liu, "Multi-sensor information ensemble-based automatic parking system for vehicle parallel/nonparallel initial state," *Sensors*. **21**(7), 2261 (2021).
[23] 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*. **16**(3), 1225–1237 (2014).
[24] S. Sedighi, D. V. Nguyen and K. D. Kuhnert. Guided hybrid A-star path planning algorithm for valet parking applications. **In:** *Proc 5th Int Conf Control Autom Robot.* (2019) pp. 570–575.
[25] H. J. Sussmann and G. Tang, "Shortest Paths for the Reeds–Shepp Car: A Worked Out Example of the use of Geometric Techniques in Nonlinear Optimal Control," **In:** *Rutgers Center for Systems and Control Technical Report*, (1991) pp. SyS–10–01.
[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," **In:** *Proc Int Conf Neural Netw.* **4**, 1942–1948 (1995).
[27] X. J. Geng, L. K. Cui, X. Y. Feng, "Research on active suspension performance optimized by linear decreasing weight PSO algorithm and LQR control," *Mod Manuf Eng*. **01**, 43–49 (2023, (in Chinese).
[28] L. Van Nguyen, N. M. Kwok and Q. P. Ha, "Fermat-Weber location particle swarm optimization for cooperative path planning of unmanned aerial vehicles," *Appl Soft Comput*. **167**, 112269 (2024).
[29] Y. Hu, D. Li, Y. He and J. Han, "Path planning of UGV based on Bézier curves," *Robotica*. **37**(6), 969–997 (2019).
[30] C. Zhou, B. Bai and N. Ye, "Reliability prediction of engineering systems using support vector regression optimized by adaptive particle swarm optimization," *J Mech Eng*. **59**(14), 328–338 (2023) (in Chinese).
[31] X. Xia, L. Gui, G. He, B. Wei, Y. Zhang, F. Yu, H. Wu and Z.-H. Zhan, "An expanded particle swarm optimization based on multi-exemplar and forgetting ability," *Inf Sci*. **508**, 105–120 (2020).
[32] G. Xiao, Y. Ma and Y. Li, "Energy-efficient trajectory optimization for planetary surface manipulators via heuristic multi-objective particle swarm optimization algorithm," *Robotica*. **43**(4), 1411–1432 (2025).
[33] S. Zhang, A. Li, J. Ren and R. Ren, "Kinematics inverse solution of assembly robot based on improved particle swarm optimization," *Robotica*. **42**(3), 833–845 (2024).
[34] L. Wang, L. Hong, H. Fu, Z. Cai, Y. Zhong and L. Wang, "Adaptive distance-based multi-objective particle swarm optimization algorithm with simple position update," *Swarm Evol Comput*. **94**, 101890 (2025).
[35] J. Fusic and R. Sitharthan, "Self-adaptive learning particle swarm optimization-based path planning of mobile robot using 2D Lidar environment," *Robotica*. **42**(4), 977–1000 (2024).
[36] R. Hamming. *Numerical Methods for Scientists and Engineers[M]* (McGraw-Hill, New York, 1962).

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use , available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737

[37] K. Kurzer, "Path Planning in Unstructured Environments: A Real-Time Hybrid A* Implementation for Fast and Deterministic Path Generation for the KTH Research Concept Vehicle," Master's Thesis, KTH Royal Institute of Technology (2016).
[38] M. Selvam, M. Ramachandran and V. Saravanan, "Nelder–Mead simplex search method-a study," *Data Anal Artif Intell*. **2**(2), 117–122 (2022).
[39] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," **In:** *2016 IEEE International Conference on Robotics and Automation (ICRA),* (2016) pp. 1271–1278.
[40] J. R. Sampson, "Adaptation in natural and artificial systems (John H. Holland)," *SIAM Rev*. **18**(3), 529–530 (1976).
[41] A. C. Khosroshahi, S. T. Afshord, B. Zarei and B. Arasteh, "Influence maximization in social networks using improved genetic algorithm," *IEEE Access*. **13**, 127490–127503 (2025).
[42] T. Rasul and K. Mukherjee, "Data-driven controller adaptation and parameter estimation in autonomous underwater vehicles by genetic algorithm enhanced unfalsification approach," *Robotica*. **43**(8), 3030–3057 (2025).
[43] X. Hong-Yang, Y. Ming, L. Jing-Jing, H. Xi and X. Wei, "LQT-based energy-efficient control for intelligent vehicles optimized by adaptive genetic algorithm," *IEEE Access*. **13**, 96800–96812 (2025).
[44] D. Dolgov, S. Thrun, M. Montemerlo, and Diebel J, "Practical search techniques in path planning for autonomous driving," **In:** *Proc AAAI Conf Artif Intell.* (2008) pp. 18–80.
[45] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," *Tech. Rep.,* Carnegie Mellon University, Robotics Institute (1992).

Cite this article: G. Hu, L. Cui, H. Wang and X. Li (2026). "Vertical parking path planning based on Reeds–Shepp curve and an improved particle swarm optimization algorithm", Robotica **44**, 1804–1829. https://doi.org/10.1017/S0263574726103737

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 13 Nov 2026 at 19:50:47, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103737