Research on Engineering Structures & Materials
www.jresm.org

Research Article

Algorithm improvement and simulation verification for mobile robot path planning in complex environments

Qiuping Pan [*,a], Mengru Chen [b]

School of Electrical Eng., Zhengzhou Railway Vocational &Technical University, Zhengzhou, China

Article Info

Article History:

Received 28 July 2026

Accepted 31 Aug 2026

Keywords:

Mobile robot;
Path planning;
Artificial potential field method;
Multi-objective optimization;
ROS simulation

Abstract

Mobile robots are increasingly used, but navigation in complex dynamic environments suffers from issues like local optima, poor obstacle avoidance, and difficulty in coordination. This paper proposes a field swarm cooperative algorithm that integrates improved potential field and Particle Swarm Optimization. It uses simulated annealing to escape local optima, leverages the potential field force to enhance obstacle avoidance, and establishes a three-objective cooperative framework. Simulation outcomes indicate that in simple obstacle environments, the proposed algorithm successfully overcomes the local minima problem in U-shaped traps using the traditional Artificial Potential Field method. In the complex static library scenario, the path length of the algorithm proposed by the study is 218.7 m, the average safe distance is 0.56 m, and the smoothness is 3.89 rad. All the indicators have achieved balanced and superior results. Compared with the single path length optimization mode, the multi-objective collaborative optimization mode, at the cost of a slight increase in path length ($p>0.05$), has achieved significant improvements in the average safe distance and smoothness ($p<0.001$).The simulation results show that the algorithm proposed in this paper is significantly superior to the comparison methods, and multi-objective equilibrium, and has important practical value for improving the intelligence level and task execution efficiency of robots.

1. Introduction

With the swift advancement of AI, the Internet of Things and automation technology, mobile robots are gradually becoming core equipment in realms like intelligent manufacturing, logistics and distribution, medical services and home services [1-2]. Their autonomous navigation ability in complex environments, especially their ability to dynamically avoid obstacles and plan multi-objective collaborative paths, has become a key indicator for measuring the level of robot intelligence. In dynamic and unstructured environments, robots not only need to be able to quickly plan a collision-free path from the starting point (SP) to the target point (TP), but also need to achieve a balance between multiple objectives like path length, safety, smoothness and real-time performance to improve task execution efficiency, lower energy consumption and ensure operational safety [3-4].

However, traditional path planning (PP) algorithms still have significant shortcomings when dealing with complex environments. For example, the computational burden of the graph search-based A* algorithm increases sharply after grid refinement; the Artificial Potential Field (APF) method guides robot movement by constructing a virtual potential field (PF), but it is prone to getting trapped in local minima; while the Rapidly-exploring Random Tree (RRT) algorithm based on sampling performs path search by randomly sampling the state space, which can effectively handle high-dimensional space and complex constraints and has probabilistic completeness, but

*Corresponding author: panqiupinng@outlook.com
[a]orcid.org/0009-0009-9911-2179; [b]orcid.org/0000-0002-3552-2103
DOI: http://dx.doi.org/10.17515/resm2026-2067im0728rs

the planning results are random, the path is usually non-optimal and not smooth, and the sampling efficiency may decrease in complex environments [5]. Furthermore, existing research often focuses on a single objective (such as the shortest path), neglecting path safety, smoothness, and energy consumption coordination, making it difficult to achieve stable, efficient, and smooth navigation in real dynamic environments [6-7].

To address the aforementioned problems, this study focuses on PP for mobile robots in complex environments. An Artificial Potential Field and Swarm Optimization (ASO) fusion optimization algorithm is designed, combining Particle Swarm Optimization (PSO) with APF to optimize the path length, safety, and smoothness. To verify the algorithm's effectiveness and practicality, a simulation model of a Mecanum wheel omnidirectional mobile robot was built based on a Robot Operating System (ROS). Multi-scenario experiments were conducted in the Gazebo environment, and the results were compared with traditional methods.

This study addresses robot PP in complex environments by proposing an ASO, combining PSO with APF to optimize paths, using ROS modeling, and conducting multi-scenario experiments with Gazebo to compare with traditional methods.

2. Related Work

2.1 PP method based on graph search and sampling

Graph search and sampling methods are widely used in robot PP due to their clear structure and ease of implementation. T. Ni et al. proposed the adaptive weighted jump Theta algorithm, which combines the advantages of Theta algorithm and jump point search to improve the real-time performance and robustness of complex scenarios [8]. T. T. C. Giang et al. proposed the fast travel fireworks algorithm to solve the problems of multi-objective PP tree expansion conflict and time consumption, and reduce cost and time [9]. Y. Huang et al. proposed the multi-strategy bidirectional RRT algorithm, which integrates multiple strategies to shorten the execution time and number of nodes [10]. Y. Kim et al. studied the energy optimization PP of differential drive wheeled robots, derived the optimal conditions, and provided theoretical support [11].

2.2 PP methods based on intelligent optimization and machine learning

In recent years, intelligent optimization algorithms and deep reinforcement learning have become research hotspots. X. Wang et al. [12] introduced a multi-strategy improvement to the ant colony algorithm, introducing APF repulsion field and other methods to optimize indicators such as path length; B. Wang et al. [13] introduced a multi-strategy improvement to the golden jackal optimization algorithm, introducing nonlinear factors and other methods to improve global convergence and dynamic obstacle avoidance (OA) capabilities. In the field of deep reinforcement learning, C. Y. Sheng et al. [14] addressed the problem of low exploration efficiency of Deep Q-Network (DQN) in complex environments by introducing an improved action selection strategy and a multi-objective fusion reward function, achieving faster convergence speed and higher learning efficiency. J. P. Carvalho et al. [15] proposed a zero-shot coverage PP framework based on deep reinforcement learning. Through course learning and environment randomization, the agent could generalize to different map sizes and subtasks, and its performance far exceeded that of traditional heuristic algorithms. In view of the problems of long path, slow convergence, non-smoothness and oscillation in existing algorithms, S. Kumar et al. [16] introduced an intelligent sparrow search algorithm. By integrating Steiner midpoint smoothing strategy and predator distance neighborhood search, it performed well in terms of path smoothness and convergence speed. D. Lestari et al. [17] proposed a genetic algorithm with dynamic crossover and mutation rate to address the efficiency problem of path search for wheeled mobile robots in grid space. By adaptively adjusting the probability of genetic operation, the algorithm could better balance global exploration and local development in the search process, thereby improving the efficiency and effect of path search. Simulation results showed that the method could effectively shorten the path length and improve the convergence speed in different grid environments.

In summary, most PP algorithms focus on single-objective optimization such as minimizing path length or computation time, lacking a systematic trade-off mechanism for the contradictory

relationships between multiple indicators like path length, safety, and smoothness. Secondly, while existing improved algorithms perform well in static obstacle scenarios, their real-time response capability to dynamic obstacles is insufficient, making it difficult to achieve fast and smooth OA trajectory replanning in complex dynamic environments. To address the problems of local minima, insufficient dynamic OA capability, and difficulties in multi-objective collaborative optimization in mobile robot PP under complex environments, this research proposes a PP method that integrates improved APFs and multi-objective PSO: At the algorithm level, Simulated Annealing (SA) is introduced to solve the local minima problem of traditional APFs, and dynamic OA capability is enhanced by incorporating the PF force into the particle swarm velocity update equation; at the optimization mechanism level, an optimization system is constructed with path length, safety, and smoothness as objectives, and roulette wheel selection is used to achieve multi-objective collaborative optimization

3. Methods and Materials

3.1. Improved APF method

APF is a classic local PP method [18]. It realizes collision-free PP from the SP to the TP. In two-dimensional space, the gravitational PF function is typically formulated as a quadratic function that depends on the distance separating the robot's current location and the TP, as presented in Equation (1).

$$U_{att}(X) = \frac{1}{2} K_{att} \cdot \rho^2(X, X_g) \tag{1}$$

In Equation (1),$U_{att}(X)$ is the gravitational PF value of the robot at position$X$;$K_{att}$is a positive proportionality coefficient, called the gravitational gain constant;$\rho(X, X_g) = \|X - X_g\|$ is the Euclidean distance between the robot's current position $X$and the TPX_g. Gravity $F_{att}(X)$ can be obtained from the negative gradient of the gravitational PF, as shown in Equation (2).

$$F_{att}(X) = -\nabla U_{att}(X) = -K_{att} \cdot \rho(X, X_g) \cdot \frac{\partial \rho}{\partial X} \tag{2}$$

The repulsive PF function is usually designed to only work when the robot enters the range of the obstacle's influence, and its form is shown in Equation (3).

$$U_{rep}(X) = \begin{cases} \frac{1}{2} K_{rep} \left(\frac{1}{\rho(X, X_o)} - \frac{1}{\rho_0} \right)^2, & \rho(X, X_o) \le \rho_0 \\ 0, & \rho(X, X_o) > \rho_0 \end{cases} \tag{3}$$

In Equation (3),$U_{rep}(X)$ is the repulsive PF value; $K_{rep}$is the repulsive gain constant; $X_o$is the obstacle position; $\rho(X, X_o)$ is the distance from the robot to the obstacle; $\rho_0$is the maximum influence radius of the obstacle. The precise expression of the repulsive force (RF) vector is given by Equation (4).

$$F_{rep} = K_{rep} \left(\frac{1}{\rho(X, X_o)} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(X, X_o)} \nabla \rho, \quad \rho(X, X_o) \le \rho_0 \tag{4}$$

In Equation (4), $\nabla \rho$ is the unit direction vector pointing from the obstacle to the robot. The $F_{att}(X)$ direction points from the robot's current position to the TP.

When the robot is within the influence range of the obstacle, the repulsive PF generates an RF $F_{rep} = -\nabla U_{rep}(X)$ pointing in the opposite direction to the obstacle. The total PF$U(X)$ experienced by the robot in the environment is the sum of the gravitational PF and the repulsive PFs of all obstacles. The corresponding total resultant force $F(X)$ can also be expressed similarly, and the calculation method is presented in Equation (5).

$$\begin{cases} U(X) = U_{att}(X) + \sum_{j=1}^{n} U_{rep}^{j}(X) \\ F(X) = F_{att}(X) + \sum_{j=1}^{n} F_{rep}^{j}(X) \end{cases} \tag{5}$$

The robot moves along the direction of the total resultant force until it reaches the TP. Although APF is widely used due to its simple structure and high computational efficiency, it may become unreachable when there are dense obstacles or obstacles near the target [19-20]. Figure 1 summarizes three typical cases of this problem. In Figure 1(a), the robot stops moving because the RFs cancel each other out; in Figure 1(b), when there are obstacles near the TP, the attraction and repulsion forces may reach a balance, making the target unreachable; in Figure 1(c), when there are U-shaped obstacles in the environment, the robot may be trapped in the concave area and unable to escape.

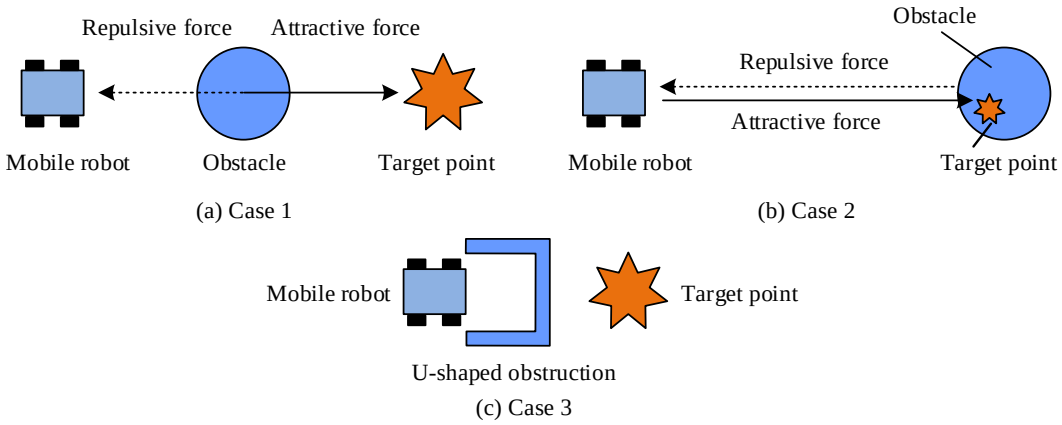


Fig. 1. The occurrence of local minima

To overcome the local minima problem, the SA algorithm was introduced into the APF method, and the SA-APF algorithm was introduced. In the SA-APF algorithm, when the robot is detected to be trapped in a local minimum (such as the position change being less than the threshold and the resultant force being close to zero in several consecutive iterations), the algorithm randomly generates multiple disturbance points near the current robot position [21]. Each disturbance point will exert a virtual RF on the robot, and the magnitude and direction of the RF are controlled by the temperature parameter of SA. When the initial temperature is high, the amplitude of the RF generated by the disturbance point is large, and the probability of the algorithm accepting unfavorable disturbances is also high, thus effectively breaking the original force balance state. As the iteration proceeds, the temperature decreases exponentially, as shown in Equation (6).

$$T_{k+1} = \alpha \cdot T_k \tag{6}$$

In Equation (6), T_k represents the temperature at the k -th iteration; α is the annealing rate constant. After the temperature decreases, the amplitude of the random perturbation gradually reduces, ensuring that the algorithm can converge stably in the later stage. The perturbation acceptance adopts the Metropolis criterion, with the acceptance probability being $P_{accept} = exp(-\Delta F/T_k)$, where $\Delta F = \|F_{perturbed}\| - \|F_{current}\|$ and $F_{perturbed}$ are the resultant forces acting on the robot after the perturbation, and $F_{current}$ is the current resultant force. When $\Delta F < 0$, the acceptance probability is 1. At each temperature, a maximum of 8 perturbations are attempted. If it is not accepted for 3 consecutive times, the temperature is reduced earlier. As the temperature decreases, the amplitude of the random perturbation gradually reduces, ensuring that the algorithm can converge stably in the later stage. The force analysis of the random disturbance point is shown in Figure 2. The direction of the resultant force has changed, guiding the robot away from the local minima.

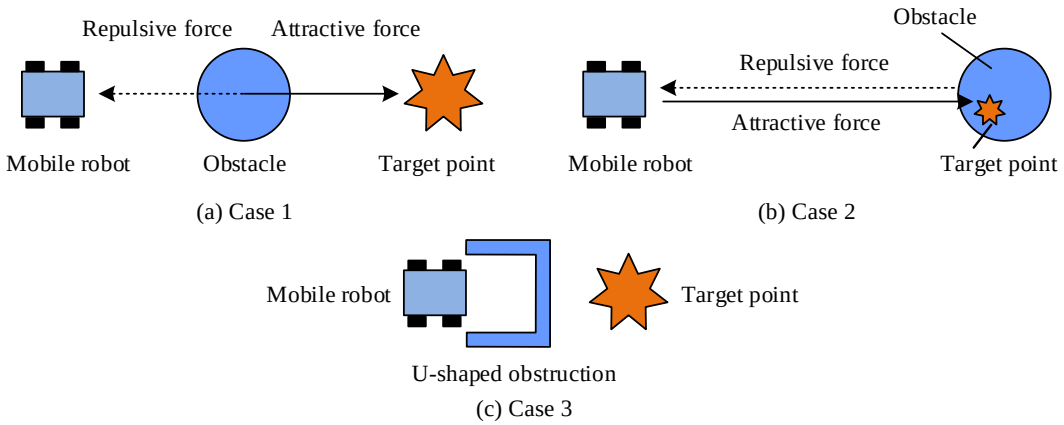


Fig. 2. Force analysis of random TPs

The complete process of the SA-APF algorithm is presented in Figure 3. The algorithm first initializes the robot's position, TP, obstacle information and SA parameters. In each iteration, the resultant force acting on the robot is calculated. If the resultant force is zero and the robot has not reached the TP, it is determined that the robot has fallen into a local minimum and the random perturbation point generation and temperature update mechanism are activated; otherwise, the robot moves forward in the direction of the resultant force [22]. When the perturbation is activated, the total resultant force is $F_{total}^{SA} = F_{att} + \sum F_{rep} + \sum F_{dist}$, where the magnitude of the repulsive force of the perturbation point is $\|F_{dist}\| = 0.5 \cdot T_k$ and the direction is from the perturbation point to the robot. The above process is repeated until the robot reaches the TP or reaches the maximum number of iterations.

The path generated by SA-APF may still have local inflections or unevenness, which can lead to frequent acceleration, deceleration, and turning during robot movement, increasing energy loss and mechanical wear. To address this, the study further introduces a moving spline smoothing algorithm to optimize the path post-processing. The moving spline smoothing algorithm first uses the moving average method to perform coarse smoothing on the original path points. For a sequence$P = \{p_1, p_2, \dots, p_n\}$ of $n$path points, the coordinates of the smoothed points are calculated by the average of the five adjacent points, as shown in Equation (7).

$$p_i^{\text{smoothed}} = \frac{p_{i-2} + p_{i-1} + p_i + p_{i+1} + p_{i+2}}{5}, \quad i = 3,4, \dots, n-2 \tag{7}$$

Boundary points are averaged with fewer points or the original value is retained. Then, the smoothed path points are used as control points, and cubic spline interpolation is used to generate a continuous smooth curve. The cubic spline function constructs a cubic polynomial between every two adjacent control points and ensures that the first and second derivatives are continuous at the connection points, thereby obtaining a globally smooth path [23]. The general form of the spline interpolation function is shown in Equation (8).

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad x \in [x_i, x_{i+1}] \tag{8}$$

In Equation (8),x_i is the x-coordinate of the$i$th path point; a_i, b_i, c_i, and d_i are undetermined coefficients, determined by the continuity of function values, the continuity of the first derivative, the continuity of the second derivative, and the natural boundary conditions (the second derivative at the TPs is zero). By solving the linear equation system, the spline functions of each segment can be obtained, thereby generating dense, smooth path points.

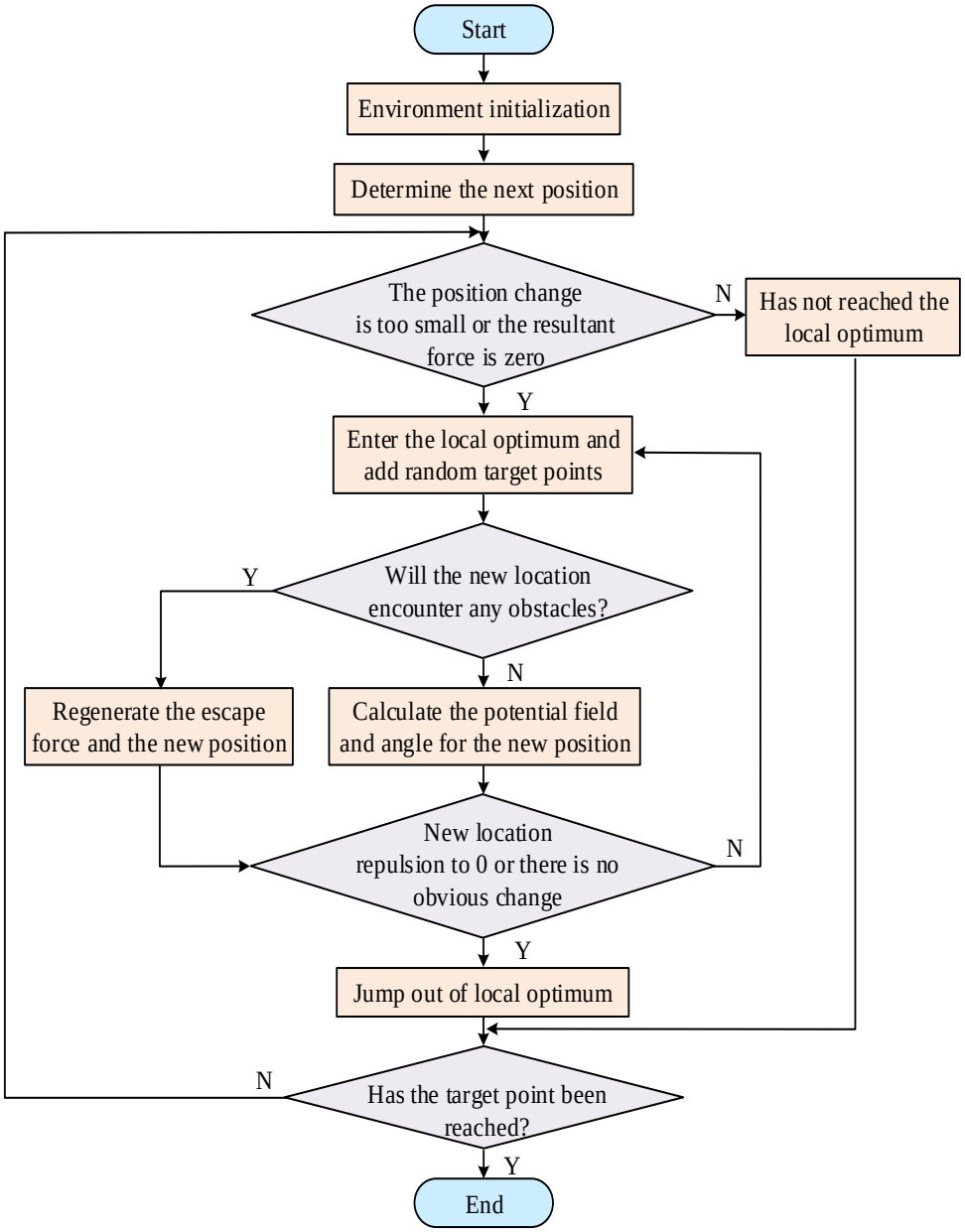


Fig. 3. Flowchart of the SA-APF algorithm

3.2 Field-Group Fusion Multi-Objective Optimization (MOO) Algorithm

Based on the improvement of the APF method to effectively overcome the local minima problem, in order to achieve the collaborative optimization among multiple objectives such as path length, safety and smoothness, a field swarm fusion algorithm integrating APF and PSO is further proposed. The fundamental concept behind this algorithm is to organically combine the local real-time OA capability based on physical PF in the APF algorithm with the global heuristic search capability based on swarm intelligence in the PSO algorithm. An MOO framework is established, and a population-based efficient solution search is introduced to obtain the optimal solution set for the balance of the three objectives, thus enabling the planning of optimal paths for robots in complex scenarios [24-25].

To quantitatively evaluate the merits of each candidate path (i.e., the path represented by the particle position) during the PSO iteration process, it is necessary to integrate the three objectives of path length, safety, and smoothness into a scalar fitness function. The study uses a linear weighted sum method to construct the fitness function, and its expression is shown in Equation (9).

$$\begin{cases} F = \omega_1 \cdot L + \omega_2 \cdot \dfrac{1}{S_1} + \omega_3 \cdot S_2 \\ L = \displaystyle\sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \\ S_1 = \displaystyle\sum_{i=1}^{n} d_i \\ S_2 = \displaystyle\sum_{i=2}^{n-1} \theta_i \end{cases} \tag{9}$$

In Equation (9), Frepresents the fitness value of the particle; the smaller the value, the better the overall performance of the corresponding path. $L$is the path length objective function, representing the total Euclidean distance of the path. $S_1$is the path safety objective function, representing the sum of distances $d_i$between each point on the path and the nearest obstacle. Since a larger safety value indicates a better path, and the fitness function aims to minimize it, its reciprocal is used for conversion. S_2 is the path smoothness objective function, representing the sum of all turning anglesθ_i in the path. ω_1,ω_2, and $\omega_3$are the weight coefficients for the corresponding objectives.

Due to the significant differences in the dimensions and numerical magnitudes among path length, path safety, and path smoothness, each target needs to be normalized to the maximum-minimum range before weighted summation, mapping each target to the interval [0,1] to eliminate the weight imbalance problem caused by the dimensional differences. After normalization, the linear weighted sum method is used to construct the fitness function, as shown in Equation (10).

$$\begin{cases} F = \omega_1 \cdot L^{'} + \omega_2 \cdot \dfrac{1}{S_1^{'} + \varepsilon} + \omega_3 \cdot S_2^{'} \\ f_k^{'} = \dfrac{f_k - f_{k,min}}{fk, min_{k,max}} \end{cases} \tag{10}$$

In Equation (10), F represents the fitness value of the particle, and the smaller the value, the better the comprehensive performance of the corresponding path. ω_1 , ω_2 , and ω_3 are the weight coefficients of the corresponding targets. L', S_1', and S_2' are the path length, path safety, and path smoothness targets after normalization. ε is a very small constant, set to 10^6, to prevent division by zero errors when S_1' is 0 after normalization. f_k' is the maximum-minimum normalization function, and $f_{k,min}$ and $f_{k,max}$ are the minimum and maximum values of the k th target within the population, which are updated dynamically during each generation of population iteration.

The traditional PSO algorithm guides particle flight through individual historical optimalityP_{best}^i and group global optimality g_{best} , but its update mechanism lacks perception of real-time environmental information [26]. To enhance the algorithm's local OA and real-time response capabilities in complex obstacle environments, the virtual force generated by the APFis directly introduced into the particle velocity update equation. The improved velocity update formula is shown in Equation (11).

$$v_i(t+1) = \omega \cdot v_i(t) + c_1 \cdot r_1 \cdot (P_{best}^i - x_i(t)) + c_2 \cdot r_2 \cdot (g_{best} - x_i(t)) + F_{att} + F_{rep} \tag{11}$$

In Equation (11),$v_i(t)$ and $x_i(t)$ represent the velocity and position vectors of the particle $i$in the tth iteration, respectively. $\omega$is the inertial weight, used to balance the particle's global exploration and local development capabilities. $c_1$and $c_2$are learning factors, controlling the step size of the particle's learning towards its individual optimal and global optimal positions, respectively. $r_1$and $r_2$are random numbers uniformly distributed in the interval [0, 1], used to increase the randomness of the search. Gravity F_{att} always points towards the global TP, causing particles to converge towards the target area; repulsion $F_{rep}$is generated by all obstacles in the environment, its direction is away from the obstacles, forcing particles away from dangerous areas.

The choice of g_{best} directly affects the convergence direction of the entire population. In MOO, there exists a Pareto front composed of numerous non-dominated solutions, rather than a single global optimum. To select a suitable guide from the non-dominated solution set while maintaining population diversity during the iterative process, this study uses fitness based reciprocal calculation of roulette wheel selection probability to dynamically determine g_{best}.

Specifically, in each iteration, all non-dominated solutions are extracted from the current population to construct a temporary elite solution set [27-28]. The Pareto dominance screening is performed based on the three original optimization objectives, namely, path length, path safety, and path smoothness, rather than the weighted-sum composite fitness F defined in Equation (10). The composite fitness F serves solely as an aggregated performance metric for executing selection operations on the obtained non-dominated candidate solutions. Following the minimization principle, a smaller value of F indicates better overall path quality. Therefore, the reciprocal of fitness is used to calculate the roulette wheel selection probability, thereby preventing inferior solutions with larger fitness values from being assigned excessively high selection probabilities. The selection probability of each individual in the elite set is computed by Equation (12).

$$P_i = \frac{1/F_i}{\sum_{j=1}^{N_e}(1/F_j)} \tag{12}$$

In Equation (12), N_e denotes the cardinality of the current elite non-dominated solution set; F_i represents the composite fitness value of the ith elite candidate. F_j denotes the composite fitness value of the jth elite candidate within the elite set for summation operation. P_i corresponds to the probability that the ith elite solution is selected as the global optimum g_{best} for the current iteration.

The algorithm generates a random number in the interval [0,1] and selects a solution from the elite solution set as the current iteration based on the aforementioned probability distribution g_{best}. This random selection mechanism allows the population to explore along different directions of the Pareto front, which helps to obtain a more evenly distributed set of non-dominated solutions, thus providing decision-makers with a series of optimization path schemes with different focuses.

3.3. ROS-based mobile robot system modeling

To confirm the capability and practicality of the introduced PP algorithm, a complete mobile robot simulation system was built based on the ROS system. The mobile robot designed in the research adopts a four-wheel independent drive Mecanum wheel structure and has omnidirectional mobility. To accurately describe the relationship between the robot's center of mass velocity and the velocities of each wheel, a kinematic analysis model of the robot, as shown in Figure 4, was established.

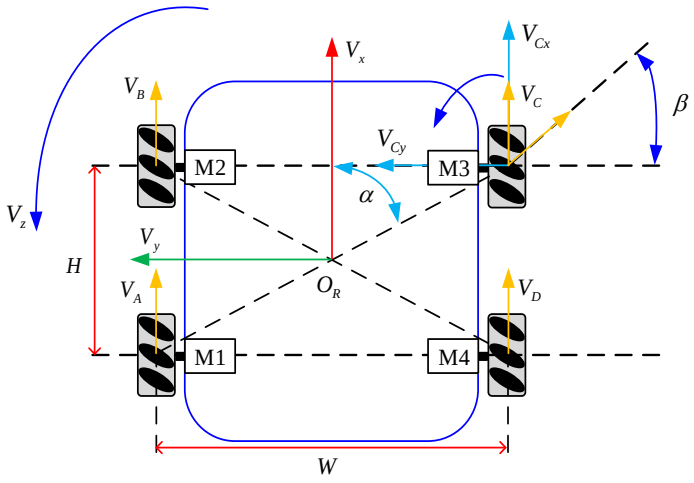


Fig. 4. Kinematic analysis diagram of the four-wheel drive wheel-driven robot

Figure 4, the robot's geometric center is set as the origin, and a robot coordinate system $X_R O_R Y_R$ is established. The robot's wheelbase is defined as H, the wheelbase as W, the robot's center of mass velocities in the X and Y directions as V_x and V_y, and the rotational angular velocity around the center of mass as V_z (counterclockwise is positive). The linear velocities of the four Mecanum wheels are V_A, V_B, V_C, and V_D. Taking wheel C as an example, assuming that there is no slippage during the robot's movement and that the center of mass is located at the geometric center, the velocity of the center of mass of wheel C can be decomposed according to Equation (13).

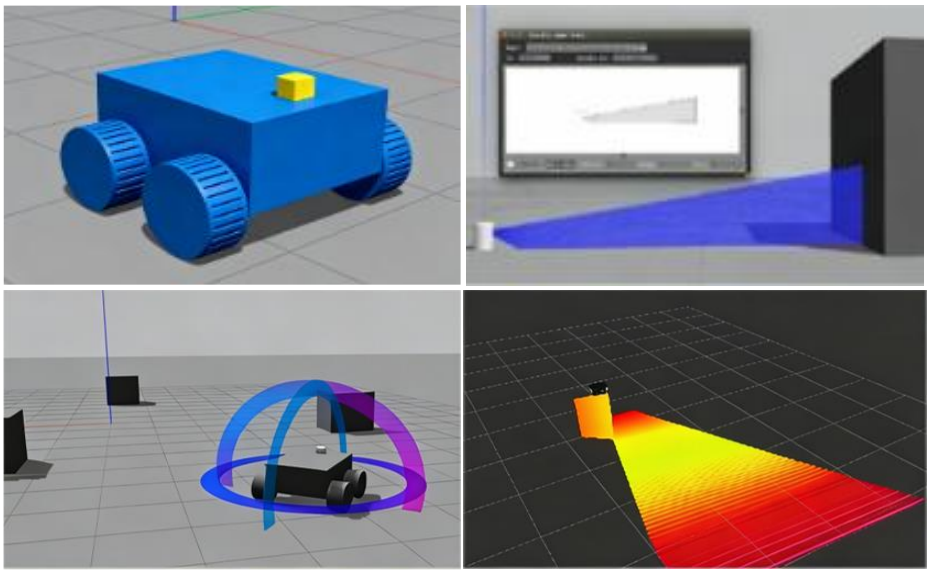
$$\begin{cases} V_{CX} = V_x + V_z \cdot I \cdot sin\,\alpha_C \\ V_{CY} = V_y + V_z \cdot I \cdot cos\,\alpha_C \end{cases} \tag{13}$$

In Equation (13), $I = \sqrt{\left(\frac{H}{2}\right)^2 + \left(\frac{W}{2}\right)^2}$ is the distance from the robot center to the wheel center, and α_C is the angle between wheel C and the robot coordinate system X_R axis. Considering the velocity synthesis of the contact point between the Mecanum wheel roller and the ground, Equation (14) is further obtained.

$$V_C = V_x + V_y + V_z \cdot \left(\frac{H}{2} + \frac{W}{2}\right) \tag{14}$$

Similarly, the velocity expressions of the other three wheels can be derived, and finally the inverse kinematics formula of the robot is obtained, as shown in Equation (15).

$$\begin{cases} V_A = V_x + V_y - V_z \cdot \left(\frac{H}{2} + \frac{W}{2}\right) \\ V_B = V_x - V_y - V_z \cdot \left(\frac{H}{2} + \frac{W}{2}\right) \\ V_C = V_x + V_y + V_z \cdot \left(\frac{H}{2} + \frac{W}{2}\right) \\ V_D = V_x - V_y + V_z \cdot \left(\frac{H}{2} + \frac{W}{2}\right) \end{cases} \tag{15}$$



(a) Robot URDF model (b) Lidar URDF model

Fig. 5. Construction of the simulation model for mobile robots (Image source: Self-drawn by the author)

At the hardware simulation level, the Unified Robot Description Format (URDF) is used to define the robot's physical structure, including chassis dimensions, wheel mounting positions, mass, inertial parameters, and joint types. The simulation model of the LiDAR sensor is described using

a separate URDF file and integrated into the robot model to simulate the point cloud data output of the real sensor. Figure 5 shows the robot simulation model visualized in Rviz.

The overall structure of the mobile robot motion control system is shown in Figure 6.In terms of software functionality, the system integrates core modules such as coordinate transformation, odometry, SLAM, PP, and navigation. The coordinate transformation module publishes coordinate system transformations through the "robot_state_publisher" node, unifying the coordinates of the LiDAR, robot base, and global map. The odometry module subscribes to the rotational speed information of each wheel, calculates the robot's pose changes by combining the kinematic model, and publishes this information via "nav_msgs/Odometry" messages. The SLAM module uses the gmapping algorithm, fusing LiDAR data and odometry information to build an environmental map in real time and simultaneously estimate the robot's pose. The navigation module, based on the "move_base" framework, integrates a global path planner and a local path planner, autonomously plans the path after receiving the TP, and controls the robot's movement.

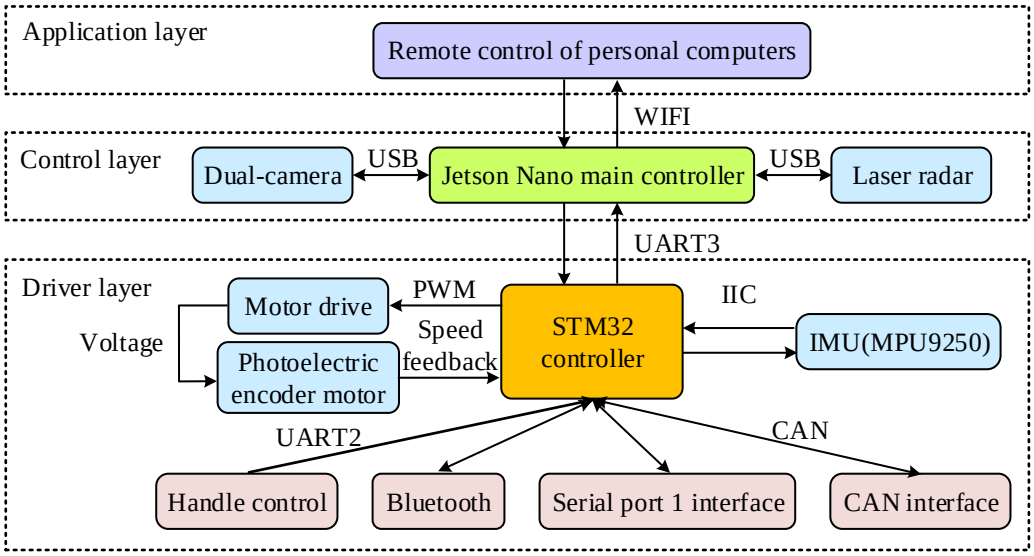


Fig. 6. Overall structure diagram of the robot motion control system

4. Experimental Results

4.1 Simulation experiment setup

The simulation experiments were conducted on a laptop equipped with an Intel Core i7-12700H processor (2.30GHz, 14 cores, 20 threads) and 16GB of DDR4 RAM, running Ubuntu 20.04 LTS. Algorithm development and verification relied on the following software platforms: MATLAB R2022a for algorithm prototype development and comparative verification in a 2D grid environment; ROS Noetic as the robot software framework, responsible for robot model description, sensor data transmission and reception, coordinate transformation management, and navigation function integration; Gazebo 11 as a 3D physical simulation environment, used to construct high-fidelity simulation scenes to simulate real-world dynamic characteristics and sensor noise; and Rviz for visualizing robot status, sensor data, planned paths, and map information, assisting in algorithm debugging and result display.

The detailed configuration of the ROS/Gazebo simulation environment was as follows: The ROS version was Noetic Ninjemys, and the workspace was built using catkin_tools; Gazebo 11 used the Open Dynamics Engine as the physics engine, and the FCL library performed collision detection. The simulation step size was set to 0.001 s, and the real-time factor was set to 1.0. The maximum linear velocity of the robot was0.5 m/s, the maximum angular velocity was 1.0 rad/s, the upper limit of linear acceleration was 0.2 m/s^2, and the upper limit of angular acceleration was 0.5 rad/s^2; The laser radar simulation used the RayPlugin in the gazebo_plugins, with an angle range of [-π,π], a sampling frequency of 10 Hz, and an effective measurement range of 0.1~30 m.

The particles of the PSO algorithm were encoded using a sequence of control points, and each particle corresponded to a complete path. The dimension of the particles was determined by the number of control points in the path. The population was initialized using a combination of two strategies: a portion of individuals were quickly generated feasible paths using APF as the initial particles; the other portion of individuals randomly sample control points in the free grid space to ensure the diversity of the initial population. All initialized particles were pre-executed for basic collision detection, and invalid individuals that collide with obstacles in the initial stage were eliminated. In the SA-APF algorithm, when the robot's position change was less than 0.15 m for 8 consecutive iterations and the magnitude of the resultant force was less than 0.08, it was determined that the robot had fallen into a local minimum, triggering the simulated annealing random perturbation mechanism.

After the moving spline smoothing was completed, dense sampling was carried out at a step size of 0.05 m on the smoothed path, and each sampling point was checked to see if it fell within the expansion layer of the obstacle (with an expansion radius of 0.3 m). If there were collision points, the path was returned to the original path point before the smooth section and the local path was re-smoothed with a smaller smoothing intensity until the entire path satisfied the no-collision constraint. If after 3 local retries still could not meet the safety requirements, the candidate path was abandoned and the smoothed path before the smooth operation was retained.

To ensure the reproducibility of the experiment, all algorithms involving randomness were set with fixed random seeds. In the MATLAB environment, rng (2024) was used to fix the random number generator state; in the Python/ROS environment, random.seed (2024) and np.random.seed (2024) were used to fix the random seeds. All comparative experiments were run under the same random seed and were independently repeated 20 times to calculate the mean and standard deviation. In the dynamic obstacle experiment, the movement trajectory of the obstacles was also generated under the fixed random seed to ensure that the trajectory was the same for each run.

To systematically evaluate the adaptability of the algorithm to environments with different complexities, three types of raster maps were designed.

- Simple environment map: Environment A (obstacles near the target): The grid size is 30×30. The SP is located at the bottom left corner (2,2), and the TP is located at the top right corner (25,25). 3-5 obstacles are randomly placed near the TP.
- Environment B (U-shaped obstacle): The size is 30×30 grid. A U-shaped trap area is set up with obstacles. The opening is facing downwards. The SP is located at the bottom left of the U (2,2) and the TP is located at the outside right of the U (25, 20).
- Complex static environment map: Environment C (simulated library scene): Based on the floor plan of a university library, the size is 200×200 grid, and includes various static obstacles such as bookshelves, tables and chairs, and pillars.
- Dynamic obstacle environment map: Environment D (Library scene with moving obstacles): Based on Environment C, set 2-3 dynamic obstacles (simulating pedestrians) that move in a straight line or back and forth in key areas of the path, with the movement speed set to 0.5m/s.

In all grid maps, black grids represent obstacles (occupancy probability ≥ 0.8), white grids represent free space, and gray grids represent unknown areas. In the simulation environment, the robot model size is set to a diameter of 0.5m, and the obstacle expansion layer radius is set to 0.3m to ensure a safe distance. The parameters of each comparative algorithm were determined based on recommended values from the literature or through pre-experimental optimization. The core parameter settings of the introduced ASO are presented in Table 1.

To fully quantify the performance of the algorithm, five indicators were used to compare and evaluate it with four algorithms: RRT [2 9], Rapidly-exploring Random Tree Star (RRT*) [30], a framework for multiple populations for multiple objectives based on ant colony optimization (MPMO-ACO) [31],and DQN [32].(1) Path length (unit:m): The total Euclidean distance of the path traversed by the robot from the SP to the TP. (2) Computation time (unit:s): The CPU time consumed by the algorithm from inputting the information of the SP and the TP to outputting the complete path. Each experiment was conducted 20 times, and the mean value was calculated to

minimize the impact of randomness. (3) Path safety (unit:m): Collect several discrete points along the planned path, calculate the Euclidean distance from each sampling point to the nearest obstacle, and then take the arithmetic mean of these distances.

Table 1. ASO algorithm parameter configuration

Parameter name	Symbol	Value
Gravitational gain coefficient	K_{att}	15.0
RF gain coefficient	K_{rep}	8.0
Obstacle influence radius	ρ_0	3.5
Initial temperature of SA	T_0	100.0
SA termination temperature	T_{min}	1.0
Annealing rate	α	0.92
Number of local minimum detection steps	N_{local}	8
Moving average window size	w_{smoot}	5
Spline interpolation density	N_{spline}	10
Particle swarm size	N_p	50
Maximum iteration count	N_{iter}	200
Inertia weight	ω	0.7~0.3
Learning factor	c_1, c_2	1.5, 1.5
Multi-objective weighting	$\omega_1, \omega_2, \omega_3$	0.4, 0.3, 0.3
Pareto solution set capacity	N_{arc}	100
Random seed	-	2024

Table 2. Comparison of PP results under different weight combinations

Weighted combination ($\omega_1, \omega_2, \omega_3$)		Path length (m)	Path safety degree (m)	Smoothness (rad)	Calculation time (s)
This study	(0.4, 0.3, 0.3)	218.7±2.1	0.56±0.04	3.89±0.11	1.43±0.05
	95% CI	(217.2, 220.2)	(0.54, 0.58)	(3.84, 3.94)	(1.41, 1.45)
Emphasizing length	(0.6, 0.2, 0.2)	215.3±1.9	0.48±0.05	4.21±0.13	1.39±0.04
	95% CI	(214.0, 216.6)	(0.46, 11.50)	(4.15, 4.27)	(1.37, 1.41)
Emphasizing safety	(0.2, 0.6, 0.2)	223.1±2.3	0.63±0.04	4.05±0.12	1.46±0.05
	95% CI	(221.5, 224.7)	(0.61, 0.65)	(3.99, 4.11)	(1.44, 1.48)
Emphasizing smoothness	(0.2, 0.2, 0.6)	221.8±2.0	0.45±0.04	3.82±0.10	1.45±0.04
	95% CI	(220.4, 223.2)	(0.43, 0.47)	(3.77, 3.87)	(1.43, 1.47)
Equal weight	(0.33, 0.33, 0.33)	220.5±2.2	0.52±0.05	3.95±0.11	1.44±0.05
	95% CI	(219.0, 222.0)	(0.50, 0.54)	(3.90, 4.00)	(1.42, 1.46)

The larger the average distance, the farther the entire path is from the obstacles, indicating higher safety. (4) Path smoothness (radians): The sum of the absolute values of the turning angles of adjacent line segments on the path. A smaller value indicates a smoother turn, which conforms to kinematic constraints. (5) OA success rate: The percentage of times the robot successfully reaches the target without collision under dynamic obstacles. 50 tests are conducted per scenario to record the success rate.

To verify the impact of the weight coefficient values on the planning results, five different weight combinations were set in environment C for comparative experiments. Each group of experiments was independently run 20 times, and the average values of each indicator were taken. The results are shown in Table 2. From Table 2, it can be seen that when the weight of the path length was increased, the path length was shortened to 215.3 m, but the safety and smoothness decreased. When the safety weight was increased, the average safety distance was raised to 0.63 m, but the path length increased accordingly, and when the smoothness weight was increased, the smoothness improved to 3.82 rad, but the safety decreased significantly. The weight combinations set in this paper achieved balanced and superior results in all indicators. The path length was 218.7 m, the average safety distance was 0.56 m, the smoothness was 3.89 rad, and the calculation time

was 1.43 s. The above results indicated that the algorithm was not sensitive to the weight coefficients within a reasonable range, and the weight selection had good robustness. The weight set in this paper could effectively balance the three conflicting optimization objectives.

4.2 Algorithm Performance Comparison in Simple Environments

Figure 7 shows the path length convergence curves of the four algorithms in environments A and B. In environment A, ASO converged first after about 25 iterations, with a path length of 28.3 meters, which was better than RRT (30.1 meters) and RRT* (29.5 meters), and the curve was smooth with slight fluctuations. In environment B, ASO stabilized after 30 iterations, with a path length of 10.4 meters, which was better than RRT (10.9 meters) and RRT* (10.7 meters), and converged the fastest. RRT and RRT* converged slowly in both environments and fluctuated greatly in the later stages. Overall, ASO had significant advantages in convergence efficiency and path quality, verifying its global search and optimization stability.

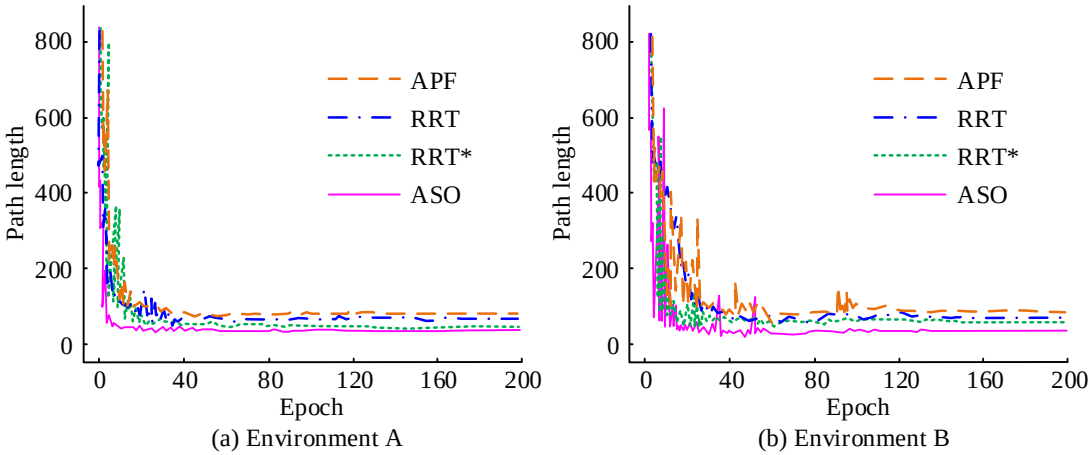


Fig. 7. Path length convergence curve

Table 3 indicates the experimental outcomes of PP in simple environments. From Table 2, the traditional APF algorithm failed to reach the TP in both scenarios due to local minima. The proposed ASO algorithm not only successfully completed all planning tasks but also significantly outperformed the comparative algorithms in regard to computational efficiency and path quality. In environment A, the computation time of ASO was 3.02 s; its path length was 28.3 m, which was 6.0% and 4.1% shorter than RRT and RRT*, respectively. In the more complex environment B, the advantages of ASO were even more pronounced. Its computation time was only 0.86 s; its path length was 10.4 m, which was 4.6% and 2.8% shorter than RRT and RRT*, respectively. This result fully verified the effectiveness of the SA mechanism and moving spline smoothing strategy in solving local minima and improving path quality.

Figure 8 shows the convergence curves of the fitness values of the four algorithms in environments A and B as a function of the iteration count. In Figure 8(a), ASO converged to the lowest fitness value first after about 40 iterations, with a smooth and stable curve, demonstrating strong global search capability; RRT converged quickly but had a slightly higher fitness value; RRT converged slowly and exhibited slight fluctuations in the later stages; APF had the highest fitness curve but large fluctuations, making it difficult to escape local optima. In Figure 8(b), ASO still converged to the lowest fitness value fastest, with a smooth curve; RRT and RRT* had lower convergence speeds and fitness values than ASO; the APF curve fluctuated at a high level, highlighting the limitations of the U-shaped obstacle environment. Overall, ASO significantly outperformed other algorithms in regard to convergence efficiency and optimization quality.

Table 3. Comparison of PP performance of different algorithms in simple environment

Environmental type	Algorithm	Calculation time (s)	Path length (m)	Whether successfully reached the target
Environment A	APF	2.96±0.18	29.8±0.6	No
	RRT	5.35±0.62	30.1±0.8	Yes
	RRT*	3.61±0.41	29.5±0.6	Yes
	ASO	3.02±0.28	28.3±0.5	Yes
Environment B	APF	1.61±0.17	11.4±0.2	No
	RRT	1.95±0.23	10.9±0.5	Yes
	RRT*	1.33±0.15	10.7±0.4	Yes
	ASO	0.86±0.09	10.4±0.3	Yes

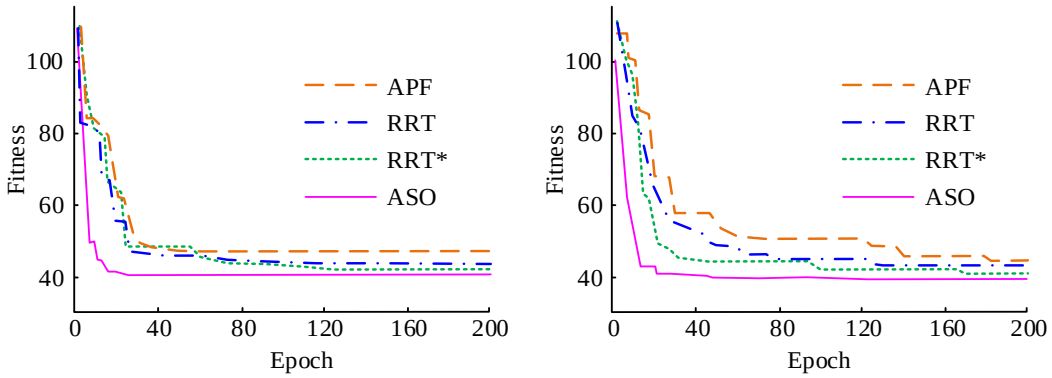


Fig. 8. Fitness iteration curve

4.3 Comparison of Algorithm Performance in Complex Static Environments

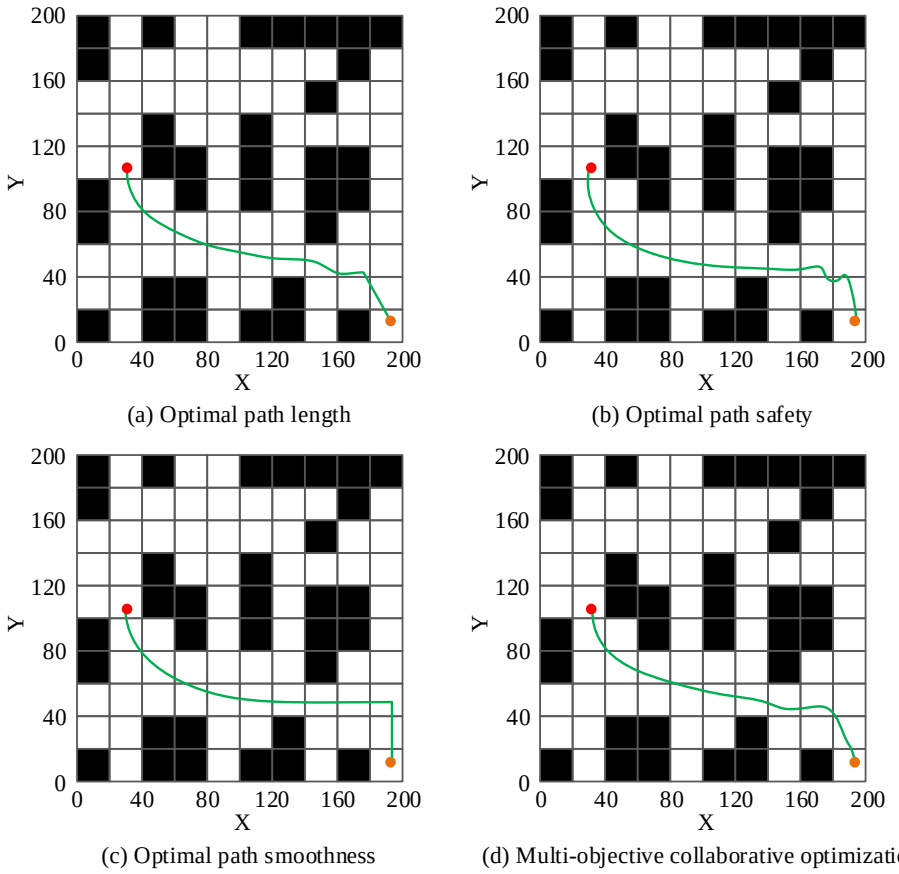
In the comparative experiment in environment C (indicators are shown in Table 4). All algorithms were run on the same hardware platform and with the same random seed (2024); for RRT and RRT, the maximum number of iterations for each planning was set to 5000. If no path was found, the process would terminate. DQN adopted a fixed training step size (100,000 steps) and was pre-trained on a static map identical to the test environment until convergence. After training, the network parameters were fixed for 20 tests. The environment maps, starting and ending positions, and obstacle distributions of all algorithms were exactly the same to ensure the fairness of the comparison. Each set of experiments was run independently 20 times. The ASO algorithm's computation time was only 1.43 seconds, far exceeding RRT (17.25 seconds) and RRT* (21.88 seconds), demonstrating high planning efficiency. Its path length was 218.7 meters, better than SA-APF, but slightly longer than MPMO-ACO, which sacrificed safety (its average safety distance was 0.39 meters, lower than ASO's 0.56 meters). Regarding path smoothness, ASO's cumulative turning angle was 3.89 radians, lower than RRT and RRT*. DQN's smoothness was similar to ASO, but its computation time (3.41 seconds) and path length (285.7 meters) were inferior. ASO achieved the best balance in computational efficiency, path length, safety, and smoothness.

Figure 9 illustrates the impact of four different optimization objectives on PP results in a complex static environment. Figure 9(a) shows the shortest path was close to the obstacle, which had a high risk of collision; Figure 9(b) shows the path was far from the obstacle, which was safe but longer; Figure 9(c) shows the path had gentle turns, but excessive smoothness made the detour longer; Figure 9(d) shows the path was moderate, safe and smooth when using multi-objective collaborative optimization to balance path length, safety and smoothness.

Table 4. Comparison of comprehensive performance of various algorithms in complex static environment

Algorithm	Calculation time (s)	Path length (m)	Path safety degree (m)	Path smoothness (rad)
APF	1.62±0.08a	285.3±4.2d	0.06±0.02d	8.52±0.35b
95% CI	(1.58, 1.66)	(283.3, 287.3)	(0.05, 0.07)	(8.36, 8.68)
RRT	17.25±1.86c	247.8±5.1c	0.53±0.06b	13.05±1.52c
95% CI	(16.38, 18.12)	(245.4, 250.2)	(0.50, 0.56)	(12.34, 13.76)
RRT*	21.88±2.34d	224.6±3.8b	0.57±0.05a	9.47±0.89b
95% CI	(20.78, 22.98)	(222.8, 226.4)	(0.55, 0.59)	(9.05, 9.89)
MPMO-ACO	1.45±0.06a	203.2±2.9a	0.39±0.04c	3.05±0.14a
95% CI	(1.42, 1.48)	(201.8, 204.6)	(0.37, 0.41)	(2.98, 3.12)
DQN	3.41±0.35b	285.7±5.8d	0.58±0.05a	3.93±0.23a
95% CI	(3.25, 3.57)	(283.0, 288.4)	(0.56, 0.60)	(3.82, 4.04)
ASO	1.43±0.05a	218.7±2.2b	0.56±0.04a	3.89±0.11a
95% CI	(1.41, 1.45)	(217.3, 220.1)	(0.54, 0.58)	(3.84, 3.94)

Note: Different superscript letters (a, b, c, d) in the same column indicate statistically significant differences between groups (Tukey HSD post-hoc test, α=0.05), while the same letter indicates no significant difference. The same as below.



(a) Optimal path length (b) Optimal path safety

(c) Optimal path smoothness (d) Multi-objective collaborative optimization

Fig. 9. Comparison of algorithm performance in environment C

The performance evaluation results of multi-objective optimization are presented in Table 5. The results of single-objective optimization exhibit typical target interdependence: when pursuing the optimal path length, the path length reaches the shortest 215.8 m, but the average safety distance was only 0.42 m, and the smoothness was 4.83 rad. When pursuing the optimal safety level, the average safety distance increased to 0.68 m, but the path length increased to 231.2 m, and the smoothness deteriorated to 5.92 rad. When pursuing the optimal smoothness, the smoothness improved to 3.67 rad, but the average safety distance dropped significantly to 0.35 m. The proposed

multi-objective collaborative optimization mode in this paper achieved a comprehensive result of 218.7 m for path length, 0.56 m for average safety distance, and 3.89 rad for smoothness within 2.86 seconds of computation time. Compared with the path length optimal mode, the multi-objective collaborative optimization only required a 2.9 m increase in path length as a trade-off, while achieving significant improvements in safety level and smoothness (p<0.001), fully verifying the effectiveness of the proposed multi-objective collaborative strategy in balancing various conflicting objectives.

Table 5. Comparison of PP results under different optimization objectives

Optimization objectives	Calculation time (s)	Path length (m)	Path safety degree (m)	Path smoothness (rad)
Optimal path length	2.68±0.06	215.8±2.3[a]	0.42±0.05[c]	4.83±0.15[c]
95% CI	(2.65, 2.71)	(214.4, 217.2)	(0.40, 0.44)	(4.76, 4.90)
Optimal path safety	2.71±0.07	231.2±2.6[b]	0.68±0.04[a]	5.92±0.18[d]
95% CI	(2.68, 2.74)	(229.5, 232.9)	(0.66, 0.70)	(5.84, 6.00)
Optimal path smoothness	2.73±0.06	229.7±2.4[b]	0.35±0.04[d]	3.67±0.12[a]
95% CI	(2.70, 2.76)	(228.2, 231.2)	(0.33, 0.37)	(3.61, 3.73)
Multi-objective collaborative optimization	2.86±0.08	218.7±2.2[a]	0.56±0.04[b]	3.89±0.11[b]
95% CI	(2.82, 2.90)	(217.3, 220.1)	(0.54, 0.58)	(3.84, 3.94)
ANOVA F-value (p-value)	F(3,76)=3.12, p=0.031	F(3,76)=9.85, p<0.001	F(3,76)=12.41, p<0.001	F(3,76)=8.67, p<0.001

4.4 Comparison of real-time OA performance of algorithms in dynamic obstacle environments

Figure 10 illustrates the PP results of the proposed algorithm in a complex environment D containing dynamic obstacles. Figure 10 shows that the robot followed the planned path through two mandatory points, demonstrating that the algorithm satisfied multi-task constraints. When encountering dynamic obstacles, the algorithm locally replanned its path to bypass them, and returned to the original path to reach the target after overcoming the obstacles. The entire process was collision-free and the trajectory was smooth, confirming the algorithm's excellent OA and path tracking capabilities in dynamic and unknown environments.

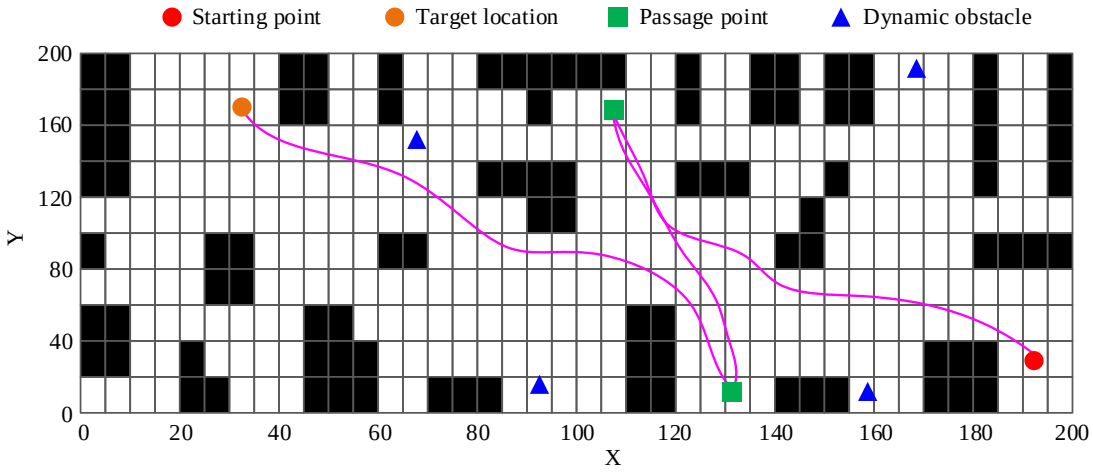


Fig. 10. Comparison of algorithm performance in environment D

The results in Table 6 showed that the performance indicators of the ASO algorithm in the dynamic OA environment were significantly superior to those of MPMO-ACO and APF. The success rate of ASO was 96.0%, with a collision rate of only 4.0%, while MPMO-ACO was 82.0% and 18.0% respectively, and APF was only 34.0% and 66.0%. The gap was extremely obvious. In terms of re-planning time, ASO only took 0.72 s, which was much faster than MPMO-ACO's 1.86 s and APF's

2.15 s, demonstrating its rapid response capability. The minimum obstacle clearance gap of ASO was 0.48 m, with the highest safety, traversal time of 28.6 s, the shortest path length of 226.5 m, and the smoothness of 4.12 rad being the best. One-way analysis of variance indicated that there were extremely significant differences in all indicators among the three algorithms, with *p*-values all less than 0.001. Tukey's post-hoc test further confirmed that pairwise comparisons between ASO and the other two algorithms were all at a significant level, confirming that ASO had the superiority of efficient, smooth and reliable real-time OA in dynamic environments.

Table 6. Performance comparison of various algorithms in dynamic obstacle environments

Algorithm	Success rate (%)	Collision rate (%)	Re-planning time (s)	Minimum obstacle clearance (m)	Traversal time (s)	Path length (m)	Smoothness (rad)
APF	34.0±6.8^{c}	66.0±6.8^{c}	2.15±0.32^{c}	0.08±0.03^{c}	45.6±5.2^{c}	312.5±18.4^{c}	11.23±1.54^{c}
95% CI	(31.1, 36.9)	(63.1, 68.9)	(2.06, 2.24)	(0.07, 0.09)	(44.2, 47.0)	(307.3, 317.7)	(10.80, 11.66)
MPMO-ACO	82.0±5.1^{b}	18.0±5.1^{b}	1.86±0.28^{b}	0.21±0.05^{b}	38.2±4.1^{b}	258.3±15.7^{b}	7.85±0.92^{b}
95% CI	(80.6, 83.4)	(16.6, 19.4)	(1.78, 1.94)	(0.20, 0.22)	(37.1, 39.3)	(253.9, 262.7)	(7.59, 8.11)
ASO	96.0±2.8^{a}	4.0±2.8^{a}	0.72±0.11^{a}	0.48±0.06^{a}	28.6±2.3^{a}	226.5±10.2^{a}	4.12±0.35^{a}
95% CI	(95.2, 96.8)	(3.2, 4.8)	(0.69, 0.75)	(0.46, 0.50)	(27.9, 29.3)	(223.6, 229.4)	(4.02, 4.22)

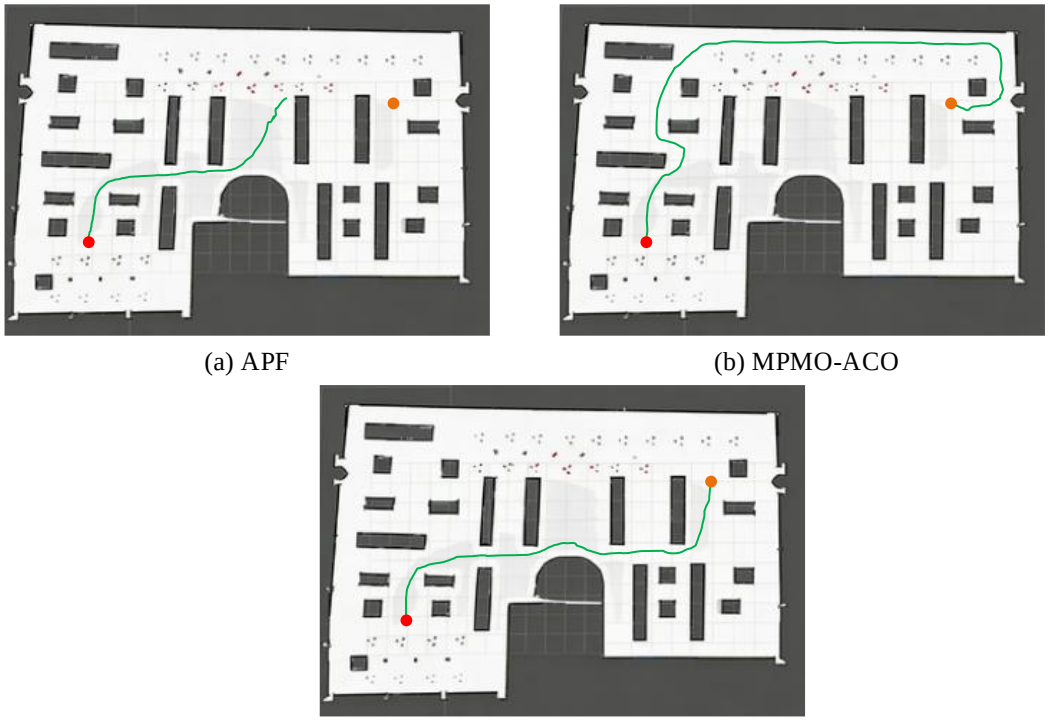
4.5. Performance verification of ASO algorithm PP

Figure 11 indicates a comparison of the PP performance of the three algorithms when facing dynamic obstacles in the Gazebo simulation environment. In Figure 11(a), the trajectory of APF showed obvious reciprocating oscillations. The robot repeatedly adjusted its direction near the dynamic obstacle but could not effectively avoid it, and eventually failed to plan its path due to getting trapped in a local minimum. In Figure 11(b), although MPMO-ACO could identify dynamic obstacles and complete OA, the path had many unnecessary detours and turns, and the overall smoothness was poor, reflecting its insufficient response speed and real-time adjustment capability. In Figure 11(c), ASO quickly identified and avoided dynamic obstacles with a smooth and continuous curve. The path length was the shortest and there were no obvious turning points. The entire motion process was stable and efficient.

Table 7. Ablation experimental results of ASO algorithm

Variation	Calculation time (s)	Path length (m)	Path safety degree (m)	Path smoothness (rad)
w/o SA	1.28±0.04^{a}	235.6±3.8^{c}	0.32±0.05^{e}	5.82±0.28^{d}
95% CI	(1.26, 1.30)	(233.8, 237.4)	(0.30, 0.34)	(5.69, 5.95)
w/o PF	1.52±0.06^{b}	225.4±3.2^{b}	0.48±0.04^{c}	4.45±0.21^{c}
95% CI	(1.49, 1.55)	(223.9, 226.9)	(0.46, 0.50)	(4.35, 4.55)
w/o PSO	1.35±0.05^{a}	241.8±4.5^{d}	0.41±0.05^{d}	6.18±0.32^{e}
95% CI	(1.33, 1.37)	(239.7, 243.9)	(0.39, 0.43)	(6.03, 6.33)
w/o Spline	1.41±0.05^{ab}	220.5±2.5^{ab}	0.54±0.04^{b}	5.86±0.24^{d}
95% CI	(1.39, 1.43)	(219.3, 221.7)	(0.52, 0.56)	(5.75, 5.97)
w/o Leader	1.45±0.06^{b}	222.8±2.8^{b}	0.50±0.05^{c}	4.22±0.18^{b}
95% CI	(1.42, 1.48)	(221.5, 224.1)	(0.48, 0.52)	(4.14, 4.30)
ASO	1.43±0.05^{ab}	218.7±2.2^{a}	0.56±0.04^{a}	3.89±0.11^{a}
95% CI	(1.41, 1.45)	(217.3, 220.1)	(0.54, 0.58)	(3.84, 3.94)

Table 7 reveals the differentiated contributions of each algorithm component to the performance of ASO through the results of the ablation experiments. After removing SA, the path length and safety distance significantly deteriorated by 7.7% (p<0.001) and 42.9% (p<0.001), respectively, indicating that the SA mechanism played an irreplaceable role in avoiding local minima. The removal of PF led to a 14.3% (p<0.01) decrease in the safety distance and a 14.4% (p<0.01) deterioration in smoothness, confirming the dual regularization effect of PF force on OA and trajectory smoothness. The absence of the PSO framework increased the path length by 10.6% (p<0.001) and the safety distance by 26.8% (p<0.001), verifying the necessity of it as the core engine for global search. The removal of spline smoothing significantly worsened the smoothness from 3.89 rad to 5.86 rad (p<0.001), highlighting the optimization effect of the post-processing stage on the continuity of path curvature. The removal of the leader selection mechanism led to a decrease in the safety distance from 0.56 m to 0.50 m (p<0.05) and a deterioration in smoothness from 3.89 rad to 4.22 rad (p<0.01). In summary, the complete ASO significantly out per for med any ablation variant in all indicators, verifying the effectiveness of the coordination of algorithm components and the necessity of the coupling between modules.



(a) APF (b) MPMO-ACO

(c) ASO

Fig. 11. Comparison of algorithm performance in environment D

5. Discussion

This study addressed the challenges of local minima trapping, insufficient dynamic obstacle-avoidance capability, and difficulties in multi-objective cooperative optimization for mobile-robot PP in complex environments. A field-swarm cooperative algorithm termed ASO, which integrated the improved APF and PSO, was proposed. The ASO algorithm achieved prominent performance advantages within simple-obstacle environments. In Environment A, where obstacles were distributed near the TP, ASO yielded a computation time of only 3.02s, which was shorter than the 5.35s consumed by the RRT algorithm and the 3.61s consumed by the RRT* algorithm; its resulting path length was 28.3m, outperforming the path-length results obtained from RRT and RRT*. More evident performance gains were observed for ASO in the U-shaped-obstacle scenario of Environment B. The computation time of ASO reached 0.86s, which was lower than the 1.95s obtained with RRT, and ASO produced a shorter path of 10.4m compared with the 10.9m path generated by RRT. It is noteworthy that the conventional artificial-potential-field method cannot reach the TP in both simple test scenarios owing to local-minima issues. Oscillation occurs around

the TP in Environment A, whereas the robot becomes trapped inside the U-shaped trap and forms a reciprocating oscillation loop in Environment B. These results verify that the embedded simulated-annealing mechanism effectively breaks the force-balance state of the APF. Once small positional variations over successive iterations and near-zero net force are detected, virtual repulsive forces produced by randomly generated perturbation points guide the robot to escape from local-minima regions. Exponential temperature decay enables stable convergence toward the TP in the later-stage iterations.

The comprehensive performance of the ASO algorithm was verified in a complex static environment experiment. In environment C, the ASO computation time was 1.43 seconds, far exceeding RRT (17.25 seconds) and RRT* (21.88 seconds), demonstrating high efficiency. The path length was 218.7 meters, better than SA-APF, but slightly longer than the safety-sacrificing Bezier smoothing algorithm (safety distance of 0.39 meters, lower than ASO's 0.56 meters). In terms of path smoothness, ASO's cumulative turning angle was 3.89 radians, lower than RRT and RRT*, better conforming to robot kinematic constraints. The moving spline smoothing algorithm first coarsely smooths the path points, then uses cubic spline interpolation to generate curves, eliminating sharp turns and preserving shape, which is beneficial for robot control.

Experiments in a dynamic obstacle environment validated the real-time OA capability of the ASO algorithm. In environment D containing moving obstacles, the robot started from the SP and passed through two pre-set necessary points in sequence. When a dynamic obstacle moved in a straight line and intruded into the planned path, the algorithm triggered a local replanning mechanism to adjust the course and detour around it. After passing the obstacle, the robot swiftly reverted to its initial path and arrived at the TP. No collisions occurred during the entire process, and the trajectory was smooth and continuous. Comparative experiments in the Gazebo environment further confirmed that ASO quickly identified and avoided dynamic obstacles with a smooth and continuous curve, achieving the shortest path length and no obvious turning points. In contrast, APF showed obvious reciprocating oscillations and got stuck in local minima, while MPMO-ACO, although able to complete OA, had several unnecessary detours and turns. This result is attributed to the design of directly introducing artificial potential force into the particle swarm velocity update equation. Gravity always points towards the global TP, causing particles to converge towards the target area, while RF is generated by all obstacles in the environment, forcing particles away from dangerous areas. This gives the algorithm both global heuristic search capability and local real-time OA capability.

The ASO algorithm has made progress but still has shortcomings: MOO relies on linear weighted integration, and the weight coefficients have a significant impact, potentially requiring dynamic adjustment when the environment changes; SA parameters are set based on experience, and adaptive adjustment could be studied in the future; currently, it mainly focuses on simulation verification and has not fully considered interference from actual robot systems. Subsequent experiments will be conducted on a real Mecanum wheel robot platform to verify its engineering practicality.

6. Conclusion

The research focused on the key issues of mobile robot PP in complex environments, such as the susceptibility to local minima, insufficient dynamic OA capability, and difficulties in multi-objective collaborative optimization. The ASO algorithm was proposed to address these problems. This algorithm introduced the SA mechanism to overcome the local optimal defect of the traditional APF, integrated the PF force into the particle swarm velocity update equation to enhance the adaptability to dynamic environments, and constructed a multi-objective framework for the collaborative optimization of path length, safety degree, and smoothness. The roulette wheel selection based on fitness was adopted to explore the Pareto frontier. A simulation platform for omnidirectional mobile robots with mechanic wheels was built based on the ROS system. Comparative experiments were conducted in grid maps and Gazebo environments. The results showed that the ASO algorithm effectively overcame the local minimum problem of APF. In the U-shaped obstacle environment, the calculation time was only 0.86s, the path length was 10.4m, and

it could successfully escape the U-shaped trap and plan a smooth path. In complex static library scenarios, the ASO algorithm could achieve a good coordination and balance among the three conflicting indicators of path length, security, and smoothness. Compared with the single pursuit of the optimal path length, the multi-objective collaborative optimization strategy, at the cost of a slight increase in path length, significantly improved security and smoothness, fully demonstrating the effectiveness of the proposed crowding distance roulette wheel selection mechanism in guiding the population to uniformly distribute along the Pareto frontier. This mechanism effectively avoided the population from prematurely converging to a specific target area by giving higher selection probabilities to non-dominated solutions in sparsely distributed regions, thereby ensuring the diversity and coverage of the solution set across the entire frontier. Moreover, in dynamic obstacle environments, the algorithm achieved effective real-time OA, with smooth trajectories and no collisions. This verified its excellent dynamic response capability under simulation conditions. The simulation results showed that ASO had excellent effectiveness and application potential in autonomous navigation under complex dynamic environments, providing a reliable algorithm foundation for the subsequent deployment of actual systems.

References

[1] Wei L, Xu Q, Hu Z. Mobile robot path planning based on multi-experience pool deep deterministic policy gradient in unknown environment. Int J Mach Learn Cybern. 2024;15(12):5823-5837. https://doi.org/10.1007/s13042-024-02281-6

[2] Moskalev PV, Stebulyanin MM, Myagkov AS. Impact of spatial resolution on mobile robot path optimality in two-dimensional lattice models. Comput Res Model. 2025;17(6):1131-1148. https://doi.org/10.20537/2076-7633-2025-17-6-1131-1148

[3] Ye F, Duan P, Meng L, Xue L. A hybrid artificial bee colony algorithm with genetic augmented exploration mechanism toward safe and smooth path planning for mobile robot. Biomim Intell Robot. 2025;5(2):57-71. https://doi.org/10.1016/j.birob.2024.100206

[4] Zhou T, Wei W. Mobile robot path planning based on an improved ACO algorithm and path optimization. Multimed Tools Appl. 2025;84(12):10899-10922. https://doi.org/10.1007/s11042-024-19370-x

[5] Hu Y, Chen X, Tang P, Zhang H, Jin J, Mao S. Path planning for mobile robots based on hybrid sampling and space-optimized RRT. IEEE Robot Autom Lett. 2026;11(3):2370-2377. https://doi.org/10.1109/LRA.2026.3653294

[6] Zhao H, Guo Y, Li X, Liu Y, Jin J. Hierarchical control framework for path planning of mobile robots in dynamic environments through global guidance and reinforcement learning. IEEE Internet Things J. 2025;12(1):309-333. https://doi.org/10.1109/JIOT.2024.3459918

[7] Greenberg B, González-Bravo U, Yi J, Feldman J. Effects of mobile robot passing-motion path curvature on human affective states in a hallway environment. Int J Soc Robot. 2025;17(12):3103-3113. https://doi.org/10.1007/s12369-025-01227-4

[8] Ni T, Liang J, Zhao Z, Zhao Y, Yang K. Adaptive weight optimization based jump point Theta* algorithm in mobile robot path planning for intricate environments. Discov Appl Sci. 2025;7(11):1343-1365. https://doi.org/10.1007/s42452-025-07932-z

[9] Giang TTC, Binh HTT, Luong HVD, Hoang NH, Huy DQ. Fast marching firework method for multi-goal mobile robot path planning in complex obstacle maps. Intell Serv Robot. 2025;18(6):1467-1484. https://doi.org/10.1007/s11370-025-00654-6

[10] Huang Y, Jiang W, Xu S. A multi strategy bidirectional RRT* algorithm for efficient mobile robot path planning. Sci Rep. 2025;15(1):1-19. https://doi.org/10.1038/s41598-025-13915-2

[11] Kim Y, Singh T. Energy optimal path planning for wheeled mobile robots with circular obstacle. Optim Control Appl Methods. 2025;46(2):583-601. https://doi.org/10.1002/oca.3208

[12] Wang X, Wang J, Cao J, Sun R. Research on mobile robot path planning based on multi-strategy improved ant colony algorithm. Robotica. 2025;43(10):3594-3614. https://doi.org/10.1017/S0263574725102610

[13] Wang B, Zhang Z, Andriukaitis D, Liu X, Hua D, Li Z, Vashishtha G, Chauhan S. MSGJO: a new multi-strategy AI algorithm for the mobile robot path planning. Int J Adv Manuf Technol. 2025;137(9):5081-5109. https://doi.org/10.1007/s00170-025-15462-6

[14] Sheng CY, An H, Nie J, Wang HX, Lu X. Improved M-DQN with ϵ-UCB action selection policy and multi-goal fusion reward function for mobile robot path planning. IEEE Trans Veh Technol. 2024;74(4):5358-5370. https://doi.org/10.1109/TVT.2024.3503552

[15] Carvalho JP, Aguiar AP. Deep reinforcement learning for zero-shot coverage path planning with mobile robots. IEEE/CAA J Autom Sinica. 2025;12(8):1594-1609. https://doi.org/10.1109/JAS.2024.125064

[16] Kumar S, Sikander A. An effective mobile robot path planner by optimizing multiple performance parameters inspired by sparrow's anti-predation exploration ability. Soft Comput. 2025;29(11):4641-4658. https://doi.org/10.1007/s00500-025-10686-w
[17] Lestari D, Sendari S, Zaeni IAE, et al. Improving efficiency and effectiveness of wheeled mobile robot pathfinding in grid space using a genetic algorithm with dynamic crossover and mutation rates. Int J Robot Control Syst. 2025;5(1):407-425. https://doi.org/10.31763/ijrcs.v5i1.1573
[18] Tong S, Liu Q, Ma Q, Qin J. Integrating deep reinforcement learning and improved artificial potential field method for safe path planning for mobile robots. Robot Intell Autom. 2024;44(6):871-886. https://doi.org/10.1108/RIA-01-2024-0011
[19] Wang Z, Zhao X, Zhang J, Yang N, Wang P, Tang J, Zhang J, Shi L. APF-CPP: an artificial potential field based multi-robot online coverage path planning approach. IEEE Robot Autom Lett. 2024;9(11):9199-9206. https://doi.org/10.1109/LRA.2024.3432351
[20] Gu X, Liu L, Wang L, Yu Z, Guo Y. Energy-optimal adaptive artificial potential field method for path planning of free-flying space robots. J Franklin Inst. 2024;361(2):978-993. https://doi.org/10.1016/j.jfranklin.2023.12.039
[21] Shan S, Shao J, Zhang H, Xie S, Sun F. Research and validation of self-driving path planning algorithm based on optimized A*-artificial potential field method. IEEE Sens J. 2024;24(15):24708-24722. https://doi.org/10.1109/JSEN.2024.3410271
[22] Yayla R, Üçgün H, Korkmaz OA. An embedded computer vision approach to environment modeling and local path planning in autonomous mobile robots. Comput Model Eng Sci. 2025;145(3):4055-4087. https://doi.org/10.32604/cmes.2025.072703
[23] Wu X, Wang X, Bai Z, Guo G. Synergistically integrated Gaussian artificial potential field and pruned rapidly exploring random tree for real-time path planning of autonomous vehicles in weakly regulated zones. J Braz Soc Mech Sci Eng. 2025;47(12):650-668. https://doi.org/10.1007/s40430-025-05963-6
[24] Li H, Li S, Wang Q, Huang X. AUV 3D path planning based on improved PSO. J Syst Eng Electron. 2025;36(3):854-866. https://doi.org/10.23919/JSEE.2025.000074
[25] Ge J, Wang T, Hu K, Wang J, Wu J, Wang J, Wu J. Optimization of power grid material warehousing and supply chain distribution path planning based on improved PSO algorithm. Sci Rep. 2025;15(1):1-19. https://doi.org/10.1038/s41598-025-33293-z
[26] Kamat GG, Yogeswarr S, Narahari, Parashivamurthy V. Comparison of bald eagle search algorithm with benchmark meta-heuristic algorithms (PSO, ABC and GWO) applied to robot path planning. Int J Reason-Based Intell Syst. 2025;17(4):226-237. https://doi.org/10.1504/IJRIS.2025.148027
[27] Sharma U, Rani S, Shabaz M. ACAMR: AI-enabled communication algorithm in ROS to improve mobile robot connectivity in Internet of Robotic Things for AMVs. J Intell Robot Syst. 2025;111(2):1-14. https://doi.org/10.1007/s10846-025-02269-6
[28] An X, Lin Y, Lin M, Wu C, Murase T, Ji Y. Federated reinforcement learning framework for mobile robot navigation using ROS and Gazebo. IEEE Internet Things Mag. 2025;8(5):45-51. https://doi.org/10.1109/MIOT.2025.3575929
[29] Wei Z, Zhang N, Yue Z, Zhou B, Li Y, Mu Z. An adaptive smoothing RRT method for path planning of concentric cable-driven manipulators. Mech Solids. 2025;60(4):2962-2979. https://doi.org/10.1134/S0025654425601417
[30] Xun X, Zhang R, Chai S, Chai R, Xia Y. Light reflection-guided RRT*: efficient path planning in narrow passages. IEEE Robot Autom Lett. 2025;10(11):11474-11481. https://doi.org/10.1109/LRA.2025.3606385
[31] Zhang Z, Lu S, Lu J, Tan S, Zou K. Autonomous underwater vehicle 3D path planning based on multiple populations for multiple objectives ant colony optimization. Cluster Comput. 2025;28(13):814-833. https://doi.org/10.1007/s10586-025-05544-1
[32] Zhang J, Chen H, Sun H, Xu H, Yan T. Convolutional neural network-based deep Q-network (CNN-DQN) path planning method for mobile robots. Intell Serv Robot. 2025;18(5):929-950. https://doi.org/10.1007/s11370-025-00613-1