

Article

An Improved Three-Dimensional RRT Path Planning Method Incorporating Path-Aware Whale Optimization

Zhaoyang Wang, Da Xu * and Yuze Ma

Army Arms University (Beijing), Beijing 100072, China; wzy626@yeah.net (Z.W.)

* Correspondence: zxyxd2022@sina.com

Abstract

Complex three-dimensional path planning requires a planner to generate collision-free, short, and smooth paths within limited computation time, but traditional RRT-based methods often suffer from unguided sampling, repeated expansion failures in dense obstacle regions, redundant initial paths, and collision-prone post-processing. To address this problem, this study defines the planning task as efficient path generation in a bounded three-dimensional obstacle space and proposes an environment feedback hybrid sampling bidirectional RRT method integrated with a path-aware improved whale optimization algorithm. In the initial search stage, the algorithm uses the collision rate of each random tree to switch among open-space exploration, heuristic convergence, and blocked region escape sampling. Local obstacle density estimation is further introduced to fuse the sampling direction, goal direction, opposite tree attraction, and obstacle repulsion, while adaptive dual step sizes, backtracking safe step size adjustment, and local rewiring reduce invalid expansions and improve the quality of the first feasible path. In the post-processing stage, the whale optimization algorithm is used to optimize key path nodes rather than all nodes, with path corridor constraints, dynamic fitness weighting, collision repair, elastic band refinement, and B-spline smoothing to shorten the path and improve smoothness while maintaining feasibility. Tested independently 100 times in each of four MATLAB three-dimensional obstacle environments and compared with the best-performing comparison algorithm in each environment, the proposed method reduced planning time by 64.4%, 83.4%, 80.1%, and 39.5%, respectively, and shortened path length by 4.9%, 7.1%, 13.4%, and 10.1%. The success rate reached 100% in the first three environments and 97% in the most complex dense obstacle environment. These results show that the proposed framework improves search efficiency, path quality, and robustness for three-dimensional collision-free path planning under complex obstacle constraints.

Keywords: path planning; RRT; whale optimization algorithm; B-spline smoothing; adaptive step size



Academic Editor: José
Manuel García-Nieto

Received: 10 June 2026

Revised: 3 July 2026

Accepted: 7 July 2026

Published: 8 July 2026

Copyright: © 2026 by the authors.
Licensee MDPI, Basel, Switzerland.
This article is an open access article
distributed under the terms and
conditions of the [Creative Commons
Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

With the accelerating integration of the new technological revolution and industrial transformation, intelligent equipment with autonomous navigation and manipulation capabilities has been widely used in industrial manufacturing, transportation, and special operations [1,2]. In high-end equipment manufacturing, multi-joint robotic arms have become important end effectors in automotive welding, electronic assembly, and medical surgical robotics because of their high precision and flexible motion control capability [3]. In transportation, automated driving is progressing toward higher levels of

autonomy, which places stricter requirements on the reliability and real-time performance of trajectory planning in complex dynamic traffic scenes [4]. Large-scale applications of unmanned aerial vehicles in power line inspection, precision agricultural mapping, and urban instant logistics also depend heavily on obstacle avoidance and planning in three-dimensional environments with dense obstacles. The main challenges it faces are computational efficiency and path quality [5,6]. These typical applications point to a common technical problem: how to efficiently generate safe, executable, and collision-free paths in complex environments.

To address this problem, path planning algorithms have evolved from classical analytical methods to modern intelligent algorithms. Early studies focused mainly on deterministic graph search methods. For example, Dijkstra's algorithm can guarantee optimality in discrete grid environments [7], whereas A* reduces the search space through heuristic cost estimation [8]. Sampling-based planning is another important framework, with representative methods including probabilistic roadmaps and rapidly exploring random trees. These methods randomly sample in high-dimensional state spaces and expand tree structures, avoiding explicit geometric modeling of obstacle regions, thereby improving the solution efficiency and applicability under complex constraints when applying planning algorithms to practical scenarios [9,10]. In addition, swarm intelligence methods such as genetic algorithms, particle swarm optimization, and ant colony optimization have been introduced into path planning to handle multi-objective optimization and local minimum problems [11].

Among these path planning methods, RRT is particularly suitable for obstacle avoidance trajectory generation in three-dimensional continuous spaces and high-dimensional configuration spaces because of its probabilistic completeness and relatively high computational efficiency [12]. However, standard RRT has two inherent limitations. First, random sampling lacks environmental feedback, which makes the search direction blind and slows the convergence of the first feasible path. Second, the generated path usually contains redundant nodes and sharp turns, so its quality is often insufficient for subsequent tracking requirements [13,14].

To overcome these limitations, recent studies have improved RRT from the perspectives of sampling guidance, tree structure optimization, and path smoothing. Tu et al. addressed trap environments composed of large indoor spaces and small exits by simplifying the cross test and using point cloud clustering to locate narrow regions; they also preset root nodes and used parallel multi-tree expansion to accelerate passage through narrow channels [15]. Feng et al. proposed DBVS-APF-RRT*, which combines directional bias and variable step sampling and constructs a heuristic sampling function from the global obstacle distribution to accelerate generation of the first feasible path [16]. Ganesan et al. evaluated Hybrid-RRT* in three 384*384 two-dimensional simulation environments and reported that its average convergence rate was 76.14% higher than that of RRT*, 24.00% higher than that of Informed-RRT*, and 3.33% higher than that of RRT*-N. The number of explored nodes was also reduced by 48.53% compared with RRT* and by 40.83% compared with Informed-RRT* [17]. Xiong et al. proposed C-RRT*, which dynamically classifies nodes according to connectivity with the start and goal and optimizes the tree structure through customized expansion, parent selection, and ordered rewiring [18]. Uwacu et al. showed that HAS-RRT, which uses workspace skeleton guidance, reduced runtime by up to 91% and generated trees at least 30% smaller than competing RRT variants while maintaining competitive path cost; its average runtime improvement over DR-RRT ranged from 59% to 91% across the tested environments [19]. For path quality optimization, Wei et al. reported that KDB-RRT* reduced average planning time by up to 97.9%, shortened path length by 5.5–21.4%, and decreased the number of inflection points by 40–90.5% compared with Opti-

mized RRT*, RRT*-Connect, and Informed-RRT* [20]. Liu et al. introduced a turning point penalty into the IA-RRT cost function to reduce the frequency of path turns [21]. Zhu et al. combined a variable steering angle strategy, Kalman filtering, and quadratic Bezier curves to suppress invalid growth and improve path smoothness [22]. Jung et al. proposed DC-RRT*, which integrates a Dubins path model with curvature-aware expansion to generate curvature-continuous flight trajectories [23]. Guan et al. improved bidirectional APF-RRT* by optimizing the dual-tree connection mechanism, triangular inequality pruning, and B-spline smoothing, thereby shortening the path and improving kinematic trackability for vehicles [24]. Overall, existing improvement strategies mainly serve two goals. The first is to find a path more quickly by shortening the time required to obtain the first collision-free connection and improving the success rate in complex environments. The second is to optimize the obtained path by reducing length, redundancy, and turning intensity through online rewiring or post-processing. However, most existing RRT variants emphasize only one part of this pipeline. Sampling-guided and potential-field-guided methods improve search directionality but may still generate redundant or obstacle-adjacent paths, whereas smoothing or post-processing methods improve path quality after a feasible path exists but usually depend strongly on the quality of the initial path.

Based on the above analysis, this study proposes an environment feedback hybrid sampling bidirectional RRT path planning method integrated with a path-aware repair whale optimization algorithm, as shown in Table 1. In the initial path search stage, the algorithm maintains the sampling count and collision count for each random tree and adaptively switches the sampling mode accordingly to improve the match between the sampling process and the environmental structure. Local obstacle density estimation is then introduced to weight and fuse the sampling direction, goal direction, opposite tree attraction direction, and obstacle repulsion direction. Combined with adaptive dual step sizes and a backtracking safe step size, this design enables fast tree expansion in open regions while reducing the step size and strengthening obstacle avoidance in dense or narrow regions. The algorithm also introduces local parent selection and rewiring and removes redundant nodes through greedy pruning. In the path post-processing stage, a path-aware repair whale optimization algorithm is proposed. Only key nodes with large turning angles, low safety margins, or high local redundancy are selected as optimization variables. Path corridor constraints, a dynamically weighted fitness function, variable neighborhood local search, and elastic band local refinement are then combined to improve the proportion of feasible solutions and the path optimization capability under limited population size and limited iterations. Finally, B-spline curves are used to generate shorter, smoother, collision-free three-dimensional planned paths.

Table 1. Relationship between research gaps and proposed mechanisms.

Manuscript Gap	Proposed Mechanism	Expected Effect
Blind sampling in dense or narrow regions	Collision-rate-driven hybrid sampling	Higher probability of sampling useful open or passage regions
Invalid expansion caused by local obstacle structure	Obstacle-density-aware fused expansion and bounded backtracking	Fewer collision-causing expansion attempts
High redundancy in feasible RRT paths	Local rewiring, greedy pruning, and key node selection	Lower optimization dimension and lower initial path cost
Collision-prone post-processing	Path corridors, collision repair, elastic band refinement, and checked B-spline smoothing	Shorter and smoother paths while preserving feasibility

2. Algorithmic Foundations

2.1. RRT and Informed-RRT* Algorithms

RRT is a single-query path planning algorithm based on random sampling and was proposed by LaValle in 1998 [25]. The algorithm builds a random tree rooted at the start point q_{start} . In each iteration, a random sample q_{rand} is uniformly generated in the state space, and the nearest node q_{near} in the tree is identified. The tree then expands from q_{near} toward q_{rand} with a fixed step size, generating a new node q_{new} . If the line segment between q_{near} and q_{new} is collision-free, q_{new} is added to the random tree. This process is repeated until the tree reaches the neighborhood of the goal point q_{goal} , after which a feasible path is obtained through backtracking, as shown in Figure 1.

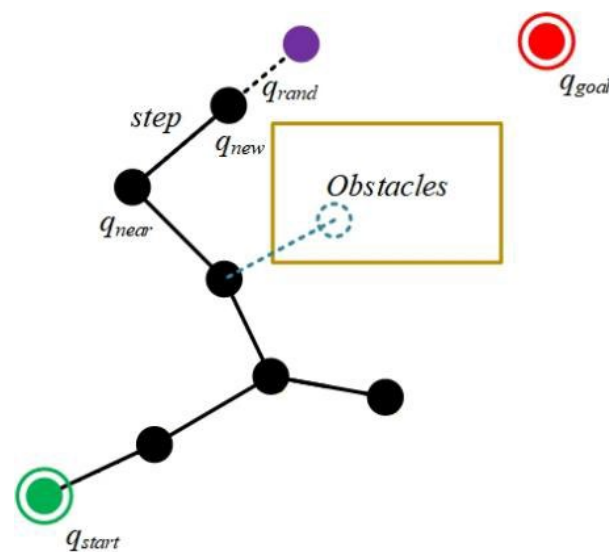


Figure 1. Schematic diagram of the RRT algorithm.

Informed-RRT* improves convergence efficiency while preserving the asymptotic optimality of RRT* by constraining the sampling space [26]. After obtaining the first feasible path, the algorithm records the current best path length c_{best} and restricts subsequent sampling to an ellipsoid whose foci are q_{start} and q_{goal} and whose major-axis length is c_{best} . This ellipsoid represents the set of all samples that may improve the current best solution. As the path is continuously optimized, c_{best} gradually decreases, the ellipsoid volume shrinks, and the sampling region becomes increasingly concentrated near the potential optimum.

2.2. Whale Optimization Algorithm

The whale optimization algorithm (WOA), proposed by Mirjalili and Lewis in 2016, is a swarm intelligence optimization algorithm that simulates the bubble-net feeding behavior of humpback whales [27]. The WOA position update equations are as follows:

$$X(t+1) = X^*(t) - A|C \cdot X^*(t) - X(t)| \text{ if } p < 0.5 \text{ and } ||A|| < 1 \quad (1)$$

$$X(t+1) = X_{rand} - A \cdot |C \cdot X_{rand} - X(t)| \text{ if } p < 0.5 \text{ and } ||A|| \geq 1 \quad (2)$$

$$X(t+1) = \left| X^*(t) - X(t) \right| \cdot e^{bl} \cdot \cos(2\pi l) + X^*(t) \text{ if } p \geq 0.5 \quad (3)$$

Here, $X(t)$ denotes the position vector of the current individual, $X^*(t)$ denotes the current global best solution, and X_{rand} denotes the position of an individual randomly selected from the population. $A = a(2r_1 - 1)$ and $C = 2r_2$, where a decreases linearly from 2 to 0 and r_1 and r_2 are random vectors in $[0, 1]$. In addition, p is a random number

in $[0, 1]$, l is a random number in $[-1, 1]$, and b is the spiral-shape constant. WOA has a simple structure, few parameters, and easy implementation, and it has been widely used in path planning and engineering optimization. However, under small population sizes and few iterations, the fixed linear decay strategy can produce an imbalance between exploration and exploitation, affecting convergence accuracy and stability.

In the proposed framework, WOA is not used to replace the RRT search process. Its purpose is to refine the feasible but redundant path generated by the bidirectional RRT stage. The WOA population represents candidate positions of selected key nodes, and the optimization objective is to reduce path length while preserving collision-free feasibility, safety distance, and smoothness.

3. Improved Algorithm Design

3.1. Environment Feedback Hybrid Sampling Bidirectional RRT

3.1.1. Collision-Rate-Driven Hybrid Sampling Strategy

Traditional RRT uses global uniform random sampling and is probabilistically complete, but it can generate many invalid expansions when the goal region is far away or obstacles are dense. To improve the match between sampled points and environmental structure, this study maintains a sampling count and a collision count for each random tree and defines the collision rate as follows:

$$P = \frac{N_{\text{collision}}}{N_{\text{sample}}} \quad (4)$$

Here, N_{sample} is the number of samples and $N_{\text{collision}}$ is the number of collisions. The collision rate P cumulatively reflects the degree to which the current tree is blocked during search. A low collision-rate threshold P_{low} and a high collision-rate threshold P_{high} are set. According to the value of P , sampling is divided into three modes: open-space exploration, heuristic convergence, and blocked region escape.

When $P < P_{\text{low}}$, the current random tree has a high expansion success rate, indicating that the local space is relatively open. In this mode, the algorithm uses a Halton low-discrepancy sequence as the main sampling method and periodically introduces fixed heuristic region sampling. This obtains more uniform spatial coverage under a limited number of samples and reduces sample clustering and large unsampled voids. To avoid completely unguided exploration, a fixed goal bias is retained in all sampling modes, allowing the tree to maintain global coverage while gradually converging toward the connected region between the start and goal.

When $P_{\text{low}} \leq P < P_{\text{high}}$, the current random tree is moderately blocked. In this state, continuing broad directionless exploration reduces efficiency, whereas excessive sampling near obstacle boundaries may increase collisions. Therefore, the algorithm uses ellipsoidal heuristic sampling with a relatively high probability, concentrating samples in a fixed heuristic region determined by the roots of the two trees and an amplification factor. A proportion of Halton sampling is retained to preserve the ability to escape local regions. The major axis of the ellipsoidal sampling region is determined by the distance between the start and goal, and the search range is adjusted by the amplification factor to balance path length optimization and feasibility.

When $P \geq P_{\text{high}}$, the current random tree collides frequently, indicating that obstacle boundaries, narrow passages, or locally enclosed structures may exist nearby. In this case, the algorithm first performs bridge sampling. Bridge sampling generates one point inside an obstacle and then generates a second point through random perturbation. If both endpoints are inside obstacles and their midpoint is in free space, the midpoint is more likely to lie near a narrow passage or a gap between obstacles. If bridge sampling

fails within a limited number of attempts, the algorithm switches to obstacle boundary sampling, generating candidate points outside the radius of a sphere or outside the surface of a cuboid to increase the probability that the random tree bypasses obstacles along their boundaries. The algorithm then selects between boundary sampling and fixed heuristic region sampling according to a preset probability. The schematic diagram is shown in Figure 2.

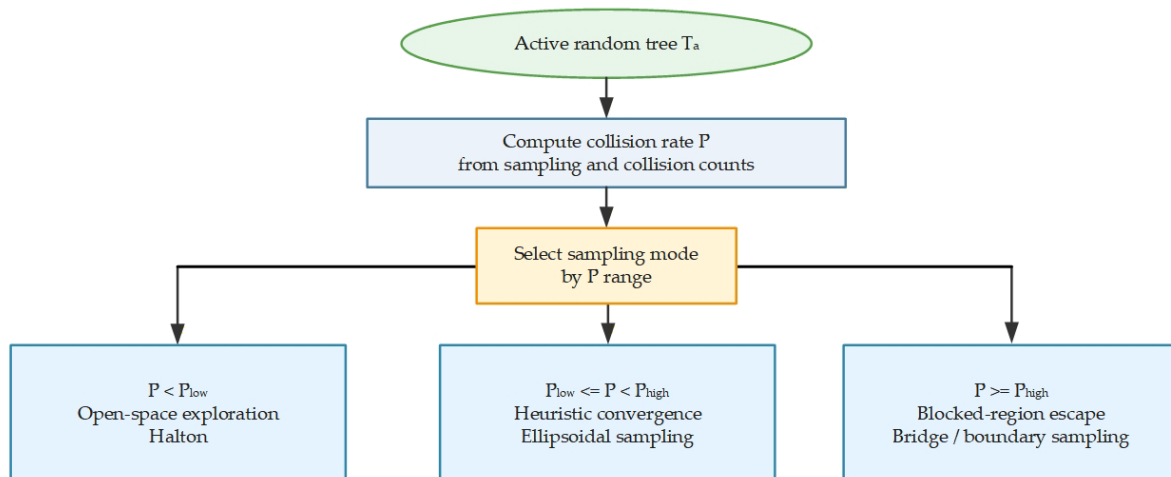


Figure 2. Collision-rate-driven hybrid sampling strategy.

3.1.2. Local Obstacle-Density-Aware Expansion

In a three-dimensional obstacle environment, the straight line direction from the nearest node to the sampled point may pass through obstacles. This is especially common in regions with dense spherical obstacles or bent passages, where expanding only along the sampling direction can cause repeated failures. Therefore, this study introduces local obstacle density estimation in the expansion stage and uses it for direction fusion and step size adjustment. For the nearest node x_{near} , 14 probing directions are set in its neighborhood, including the positive and negative directions of the three coordinate axes and eight spatial diagonal directions. A probing point is generated in each direction, and the line segment from the current node to the probing point is checked for obstacle intersection. Let K be the total number of probing directions and $n_{collision}$ be the number of colliding probes. The local obstacle density ρ is defined as follows:

$$\rho = \frac{n_{collision}}{K} \quad (5)$$

When ρ is low, step size attenuation is weak, and the random tree can maintain a large basic step size for rapid expansion. When ρ is high, the node is close to obstacles or located in a narrow passage, so the step size should be reduced and the obstacle avoidance tendency should be strengthened. The expansion direction is designed as a weighted fusion of four direction types. The first is the sampling direction u_s , which preserves the random exploration capability of RRT. The second is the goal direction u_g , which accelerates convergence toward the goal region. The third is the opposite tree attraction direction u_c , which enhances the tendency of the two trees to connect. The fourth is the obstacle repulsion direction u_r , which reduces the probability of expansion into obstacles, as shown in Figure 3. The composite direction is expressed as follows:

$$u = w_s u_s + w_g u_g + w_c u_c + w_r u_r \quad (6)$$

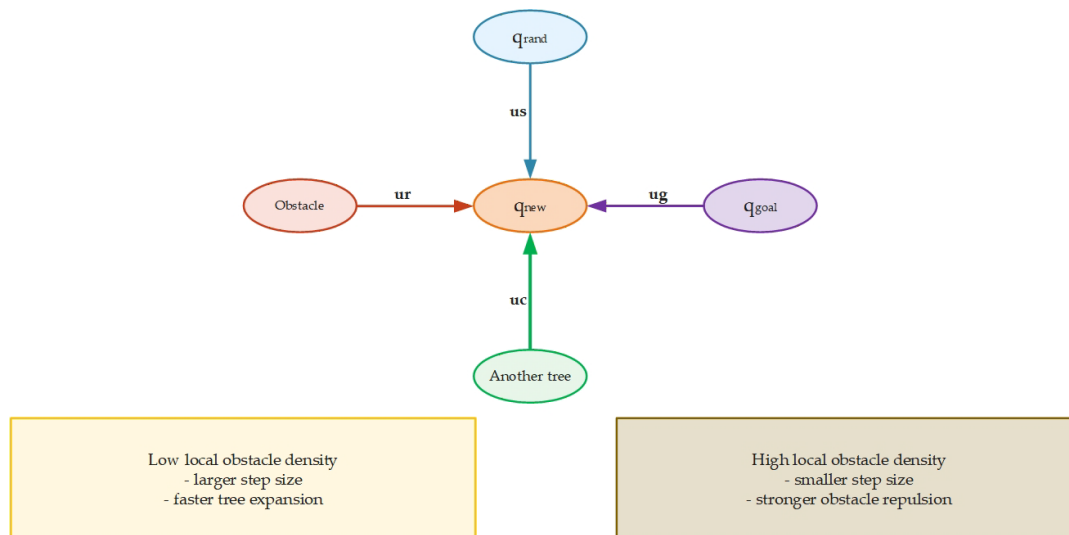


Figure 3. Direction fusion.

Here, u_s , u_g , u_c , and u_r denote the sampling direction, goal direction, opposite tree attraction direction, and obstacle repulsion direction, respectively. The corresponding weights are w_s , w_g , w_c , and w_r . If the previous expansion succeeds, w_s and w_g are gradually increased to accelerate search. If the previous expansion fails, they are decreased to reduce the collision risk caused by overly long expansions. The value of w_c increases as the need to connect the two trees becomes stronger, whereas w_r increases with the local obstacle density ρ and the collision rate P . When ρ or P is large, the current region is difficult to traverse, so the repulsion term is significantly strengthened and the expansion direction tends to move away from obstacles. The expansion step size is adjusted according to the local obstacle density as follows:

$$S = \max(S_{\min}, S_F \cdot (1 - c_1 \cdot \rho)) \quad (7)$$

Here, $S_{\min}$ is the minimum step size, and S_F is the basic step size. S_F is dynamically updated according to whether the previous expansion succeeds: it increases after successful expansion and decreases after failed expansion, while remaining within the preset step size range. The parameter c_1 is the local obstacle density limitation parameter. The local density term can only maintain or shorten the step size and cannot make the final step size exceed the current basic step size. This mechanism shortens the step size as the local obstacle density increases, producing finer-grained node expansion in dense obstacle regions. In open regions, the step size is mainly determined by the basic step size and the expansion state, allowing faster search. The final new node is generated as follows:

$$x_{new} = x_{near} + S \frac{d}{\|d\|} \quad (8)$$

Through fused expansion directions and adaptive step sizes, the exploration process of the random tree is no longer determined by a single sampled point. Instead, it jointly considers random exploration, goal convergence, dual-tree connection, and obstacle avoidance safety. The sampling direction provides global randomness and prevents the algorithm from degenerating into greedy search. The goal direction strengthens convergence toward the goal region. The opposite tree attraction direction uses the structural advantage of bidirectional RRT to reduce the connection distance between the two trees. The obstacle repulsion direction plays a larger role when the collision rate and local density are high,

reducing the probability of expansion into obstacles or deeper along the normal direction of obstacle boundaries.

3.1.3. Backtracking Safe Step Size

In complex three-dimensional spaces, even when the fused expansion direction is generally reasonable, the initial step length may still be too large, causing the local line segment to pass through obstacles. If this sampling point is discarded directly, the directional information already obtained will be wasted and the number of repeated sampling attempts will increase. To address this, a fallback-based safe step length strategy is adopted. It gradually shortens the step length while keeping the expansion direction unchanged, attempting to obtain a smaller feasible expansion in that direction.

The algorithm first generates an initial step length S_0 , whose value is jointly determined by the current dynamic step size parameters k_1, k_2 and the local obstacle density ρ . S_{min} and S_{max} are the minimum and maximum step lengths, respectively.

$$S_0 = \max(S_{min}, (k_1 + l_1 \cdot k_2) \cdot (1 - l_2 \cdot \rho)) \quad (9)$$

Here, l_1 and l_2 are control factors. A candidate new node is generated along the guidance direction with step length S_0 , and the line segment between the two nodes is checked for intersection with obstacles. If no collision occurs, the node is accepted; if a collision occurs, the step length is multiplied by a fallback coefficient η , and a candidate node is regenerated along the same direction. That is, the step length after the k -th fallback is:

$$S_k = S_0 \cdot \eta^k \quad (10)$$

The fallback process terminates successfully when a collision-free feasible step length is found, and terminates with failure when the step length is reduced below S_{min} but still cannot avoid obstacles. Meanwhile, to prevent endless fallbacks in a completely blocked direction, a maximum number of fallbacks K_{max} is set. When the current expansion fails, the collision count of the current tree is updated. The fallback step length and the collision rate feedback form a complementary relationship: if a valid node can be obtained after a fallback, it indicates that the original expansion direction is not entirely invalid but that the initial step length was too large; the algorithm retains this direction for further expansion. If multiple fallbacks still fail, the direction is judged to be infeasible, and this influences the subsequent sampling mode through the collision count.

3.2. Auxiliary Path Optimization

To improve the quality of the initial path, this study introduces RRT*-style local parent selection and rewiring within a limited neighborhood in bidirectional RRT. This mechanism corrects obviously low-quality connections within a small neighborhood so that the initial path has lower redundancy before entering post-processing. Greedy pruning is then used to remove unnecessary intermediate nodes and reduce the optimization dimension of PAWOA.

After PAWOA optimization, both the path length and node positions are improved, but discontinuous turns may still appear at local corners. Therefore, local B-spline smoothing is applied to each intermediate node. Local control points are determined by analyzing the preceding and following path segments, generating a smooth curve segment. Each smoothed B-spline segment is discretized with a fixed resolution of 40 interpolation points, and collision detection is performed between every two adjacent discrete points: if a sampled segment collides with an obstacle, the smoothing distance is multiplied by 0.6 and that segment is rechecked; if the collision-free constraint still cannot be satisfied after

several distance reductions, the original polyline node is retained. This method enhances path smoothness while ensuring safety.

3.3. Path-Aware Repair Whale Optimization Algorithm

After the pruned path is obtained, directly using standard WOA to optimize all intermediate nodes would lead to high optimization dimensionality, many invalid individuals, and collision paths that can only be passively eliminated by penalty functions. Therefore, this study proposes a path-aware repair whale optimization algorithm (PAWOA), which explicitly embeds path geometry and obstacle constraints into the WOA search process, as shown in Figure 4.

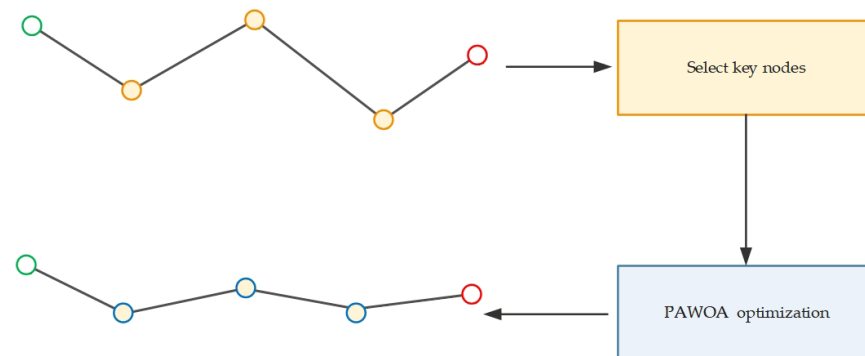


Figure 4. Schematic diagram of PAWOA optimization effect.

3.3.1. Selection of Key Path Nodes

The path generated by the improved RRT algorithm and processed by rewiring and pruning is already basically feasible, but it may still contain nodes with large turning angles, proximity to obstacle boundaries, or high contribution to total path length. If all intermediate nodes are used as WOA optimization variables, the search dimension increases linearly with the number of path nodes, making stable convergence difficult under limited population size and iteration count. In addition, many non-key nodes may destroy the original feasible path after global perturbation, increasing the proportion of colliding individuals. Therefore, this study adopts a key node selection strategy and optimizes only the nodes that have substantial influence on path shape and safety margin.

Let the pruned path be $L = (p_0, p_1, \dots, p_n)$, where p_0 and p_n are the start and goal points, respectively. For any intermediate node p_i , the change in direction between adjacent path segments is first calculated to characterize the turning degree at that node:

$$\theta_i = \arccos \left(\frac{(p_i - p_{i-1}) \cdot (p_{i+1} - p_i)}{\|p_i - p_{i-1}\| \|p_{i+1} - p_i\|} \right) \quad (11)$$

When θ_i is large, a pronounced corner exists near the node, and optimizing the node helps improve path smoothness. Next, the minimum distance d_i from the node to obstacles is calculated. If d_i is small, the node is in a low-safety-margin region, where a small perturbation may cause collision but may also produce a safer path through proper adjustment. Finally, the local contribution of the node to path length is considered by comparing the length of the two polyline segments with the direct connection length:

$$\Delta L_i = \|p_i - p_{i-1}\| + \|p_{i+1} - p_i\| - \|p_{i+1} - p_{i-1}\| \quad (12)$$

If ΔL_i is large, the node causes an obvious detour and has high optimization value. By integrating turning angle θ_i , safety distance d_i , and local redundancy ΔL_i , the algorithm selects the set of key nodes and uses only their three-dimensional coordinates as WOA

decision variables. This strategy reduces the WOA search dimension, enabling effective optimization under small populations and limited iterations. It also preserves nodes that are already favorable or safe in the original path, avoids unnecessary perturbations that could destroy feasibility, and improves consistency between population search directions and path improvement objectives.

3.3.2. Path Corridor Constraints and Elite Opposition Initialization

Standard WOA usually initializes the population and updates individual positions within global search boundaries. In three-dimensional path post-processing, such global updates can move key nodes far away from the original feasible path, causing adjacent path segments to cross obstacles over large regions and generating many infeasible individuals. Because the pruned path is already a collision-free feasible path, the goal of post-processing is not to restart global path search but to find a shorter and smoother local improvement near the original path while satisfying safety constraints. Therefore, this study constructs path corridor constraints centered on the pruned path.

For each key node, a local search range is established around its current position, and the corridor radius is adjusted according to the safety distance from the node to obstacles. If the node is close to obstacles, the corridor is narrowed to reduce collision risk. If the node is in an open region, the corridor is appropriately enlarged to preserve the opportunity for path shortening. The lower and upper bounds of the k -th coordinate dimension of a key node are expressed as follows:

$$lb_{ik} = \max(0, p_{ik} - r_i) \quad (13)$$

$$ub_{ik} = \min(1000, p_{ik} + r_i) \quad (14)$$

$$r_i = \min(R_{max}, \max(R_{min}, \alpha \cdot d_i + \beta)) \quad (15)$$

Here, r_i is the corridor radius of node p_i , R_{max} and R_{min} are the upper and lower bounds of the corridor radius, α and β are the clearance scaling coefficient and the base offset, respectively, and d_i is the Euclidean distance from the key node i to the nearest obstacle surface, and the planning space boundary is $[0, 1000]$. This constraint shrinks the search space from the global cube to a local feasible region near the original path, making it easier for WOA individuals to maintain path continuity.

During population initialization, this study further adopts an elite opposition-based learning strategy. The algorithm first generates candidate individuals around the original key nodes within the path corridor and then constructs their opposite individuals:

$$X_i^{opp} = lb + ub - X_i \quad (16)$$

Here, X_i is a candidate individual, and lb and ub are the local lower and upper bounds of the corresponding dimension. The algorithm then calculates the fitness values of the candidate and opposite individuals and retains the better one in the initial population. This strategy improves the diversity and quality of the initial solutions without increasing population size and reduces invalid early WOA iterations in low-quality regions. Path corridor constraints and elite opposition initialization jointly form the path-aware initialization mechanism of PAWOA. The former keeps the search range close to the original feasible path, whereas the latter improves initial exploration quality under a limited population, thereby alleviating the tendency of standard WOA to generate invalid individuals in path post-processing.

3.3.3. Dynamically Weighted Fitness Function

Path post-processing does not merely pursue the minimum path length. If only path length is optimized, individuals may approach obstacle boundaries or even remain collision-free only marginally through penalty terms, thereby reducing actual tracking safety. If safety distance and smoothness are overemphasized, the path may become too long. Therefore, this study constructs a multi-objective weighted fitness function composed of path length, smoothness, safety distance, and collision penalty:

$$F = L + w_{st}S + w_{ct}C + M \quad (17)$$

Here, L denotes path length, S denotes the smoothness penalty, C denotes the safety distance penalty, M denotes the collision penalty, w_{st} and w_{ct} are dynamic weights that vary with iteration count. The path length term is defined as follows:

$$L = \text{sum} \|p_{i+1} - p_i\| \quad (18)$$

The smoothness term is used to suppress sharp turns in the path. For each intermediate node, the angle between adjacent path segments is calculated; a larger angle produces a higher smoothness penalty:

$$S = \text{sum} (1 - \cos\theta_i) \quad (19)$$

The safety distance term prevents the path from approaching obstacles excessively. Let d_i be the distance from a node or path sample to the nearest obstacle, and let d_{safe} be the desired safety distance. The safety penalty can be expressed as follows:

$$C = \text{sum} \max(0, d_{\text{safe}} - d_i) \quad (20)$$

The collision penalty M has the highest priority. If any path segment intersects an obstacle, a large penalty is imposed on the fitness value so that colliding paths are rapidly eliminated during selection. Unlike methods that rely only on collision penalties, this study also introduces a safety distance term, so the path actively moves away from obstacle boundaries even when no collision occurs. Dynamic weights regulate the optimization emphasis at different iteration stages. In early iterations, when the population distribution is not yet stable, the algorithm increases the weights of smoothness and safety distance so that individuals preferentially enter feasible and safe regions. In later iterations, as the proportion of feasible solutions increases, the soft constraint weights are gradually reduced, allowing the search to focus more on path length compression. This strategy forms a staged balance among feasibility, safety, and smoothness.

3.3.4. Collision-Aware Repair and Elastic Band Local Refinement

Standard WOA usually handles infeasible solutions through fitness penalties. In path planning scenarios with dense obstacles, this passive elimination mechanism can leave many individuals in colliding states for many iterations, reducing the effective search ratio. Therefore, this study introduces a collision-aware repair operator after each individual position update to actively repair key nodes that cause collisions.

After key node p_i is updated, the algorithm checks whether the adjacent path segments (p_{i-1}, p_i) and (p_i, p_{i+1}) collide. If a collision exists, the midpoint of the adjacent nodes is used as the base position:

$$p_i^{\text{mid}} = \frac{p_{i-1} + p_{i+1}}{2} \quad (21)$$

The algorithm then superimposes an obstacle repulsion direction and random perturbation to generate several candidate repair points. Each candidate point must satisfy both the boundary constraints and the collision-free constraints of adjacent line segments, and

the best repair point is selected through a comprehensive evaluation of local path length and safety distance. If no valid repair point can be found, the original feasible structure is retained, or the variable is constrained back into the path corridor. This active repair mechanism can increase the proportion of feasible paths in the population, allowing WOA iteration resources to be used more for optimizing path quality rather than repeatedly evaluating obviously colliding individuals.

In addition to collision repair, elastic band length refinement is embedded in the WOA iteration process. The elastic band concept treats the path as a discrete curve pulled by adjacent nodes, allowing intermediate nodes to contract toward the line direction between adjacent nodes under collision-free constraints. For an intermediate node in the current best path, the ideal contraction direction is given by the midpoint of its adjacent nodes:

$$p_i^{new} = p_i + \alpha \left(\frac{p_{i-1} + p_{i+1}}{2} - p_i \right) \quad (22)$$

Here, α is the local contraction step size. If the adjacent path segments after contraction are collision-free, the update is accepted; otherwise, the step size is reduced or the original node is retained. The algorithm performs lightweight elastic band refinement on the current best path every several generations, enabling the local path structure to evolve continuously toward a shorter and smoother form. The pseudo code of the algorithm is shown in Table 2.

Table 2. Algorithm pseudo code.

```

1: Initialize two random trees Tstart and Tgoal.
2: for k = 1 to Nmax do
3:   Select the active tree and compute its collision rate.
4:   Generate grand using the environment feedback hybrid sampling strategy.
5:   Find the nearest node qnear in the active tree.
6:   Estimate the local obstacle density around qnear.
7:   Generate qnew using the fused expansion direction and adaptive step size.
8:   if the path segment from qnear to qnew is collision-free then
9:     Add qnew to the active tree and perform local rewiring.
10:    Try to connect qnew with the opposite tree.
11:    if the two trees are connected then
12:      Extract and prune the initial feasible path Pinit.
13:      break.
14:    end if
15:  end if
16:  Swap the two trees.
17: end for
18: if no feasible path is found then return failure.
19: Select key nodes from Pinit as optimization variables.
20: Construct path corridor constraints around the selected key nodes.
21: Optimize the key nodes using PAWOA
22: B-spline smoothing.
23: return the final optimized path Popt.

```

4. Simulation Experiments and Analysis

To comprehensively verify the performance of the proposed algorithm, a three-dimensional obstacle space of size 1000*1000*1000 was constructed in MATLAB R2019a, and four obstacle maps were established. Three improved algorithms based on Informed-RRT* were selected for comparison: Algorithm *a* adds a goal bias strategy, Algorithm *b* adds bidirectional expansion and goal bias, and Algorithm *c* adds an adaptive artificial potential field and goal bias. The main parameters were set as follows: maximum number of iterations, 15,000; connection threshold, 30; goal bias probability, 0.2; basic step size, 20,

constrained to the range 10–70; WOA population size, 14; and maximum number of WOA iterations, 45. All simulations were conducted under the same hardware environment, and each group of experiments was run independently 100 times to reduce the influence of randomness. In the figures, the results correspond sequentially to Algorithms *a*, *b*, *c*, and the proposed improved Algorithm *d*.

Figures 5–8 show the planning results of the four algorithms in the four environments. Environment 1 is dominated by large plate-like obstacles and has narrow traversable space. Algorithm *a* shows severe tree diffusion and an obvious detour, Algorithm *b* converges faster but still exhibits backtracking, Algorithm *c* has stronger directionality but still oscillates, and Algorithm *d* uses the fewest sampled points and produces the simplest path. In Environment 2, the obstacles are relatively scattered, but the feasible passages are interlaced. Algorithms *a* and *b* generate many search branches near the middle obstacle region and show obvious invalid exploration. Guided by the artificial potential field, Algorithm *c* tends to advance along obstacle boundaries and produces a more compact search tree than the first two algorithms. Algorithm *d* generates fewer search nodes, a more concentrated distribution, and a route closer to the effective connection between the start and goal. Environment 3 requires selecting a reasonable detour direction in three-dimensional space rather than simply avoiding regular obstacles. Algorithm *a* has a large search tree coverage range and a long final detour. Algorithm *b* benefits from bidirectional growth but still forms many branches near spherical obstacles. Algorithm *c* produces a more directional search process around obstacles, yielding better path quality than the first two algorithms. Algorithm *d* obtains a smoother spatial detour trajectory while maintaining fewer expansions. Environment 4 has dense obstacles and narrow passages and is the most demanding for the planners. Algorithms *a* and *b* show large-scale search divergence, Algorithm *c* improves the result but still has an obvious detour, whereas Algorithm *d* maintains fewer sampled points and good path continuity and feasibility. Further observation shows that although the pruned path has removed many redundant nodes, its nodes still originate from sampled points or line segment intersections in the original RRT search tree. The overall trend is therefore constrained by the geometry of the initial random tree, and the path retains detouring turns near obstacle boundaries and sharp corners. In contrast, after improved WOA and local B-spline curve optimization, the path is no longer restricted by the positions of the initial discrete sampled points and is adjusted globally toward a shorter and straighter form under collision-free constraints. The optimized path has a more uniform node distribution, fewer turning points, and smoother turns.

As shown in Table 3, Algorithm *d* demonstrates a comprehensive advantage across the four types of environments. Compared with the best-performing comparison algorithm in each environment, Algorithm *d* reduces planning time by 64.4%, 83.4%, 80.1%, and 39.5%, and shortens path length by 4.9%, 7.1%, 13.4%, and 10.1%, respectively. In terms of average performance, the average planning time of Algorithm *d* is only 0.85 s, which is lower than the 4.13 s of Algorithm *a*, the 2.75 s of Algorithm *b*, and the 4.58 s of Algorithm *c*; its average path length is 1922.07, also the shortest among the four algorithms. Regarding the success rate, Algorithm *d* maintains a 100% success rate in Environments 1 to 3 and still achieves 97% in Environment 4—the one with the densest obstacles and the narrowest passages—which is 13 percentage points higher than that of Algorithm *c*, the comparison algorithm with the highest success rate. The success rate in Environment 4 is lower than 100% because this environment has the most complex obstacle distribution among the four test cases. Although RRT-type algorithms are probabilistically complete in the asymptotic sense, this property does not guarantee successful planning in every independent run when the maximum number of iterations is limited [28]. The above results indicate that

the environment feedback sampling strategy can effectively reduce invalid expansions and improve search efficiency, while the PAWOA post-processing further improves path length, feasibility, and planning stability, enabling the algorithm to achieve better comprehensive planning performance in complex three-dimensional obstacle environments. Figures 9 and 10 further validate the comprehensive advantages of the proposed Algorithm d from a statistical distribution perspective. As can be seen from Figure 9, in the four types of environments, the planning time boxes of Algorithm d are all located in the lowest range, and both the boxes and whiskers are relatively short overall, indicating that it not only has the lowest average time consumption but also exhibits small time fluctuations over multiple independent runs. Even in Environment 4, which features the densest obstacles and the narrowest passages, Algorithm d still maintains a planning time distribution significantly lower than those of algorithms a, b, and c. Figure 10 shows that Algorithm d likewise achieves the overall shortest path lengths across the four environments, with a more concentrated distribution in most cases. Overall, Figure 6 reflects the improvement in search efficiency brought by Algorithm d, while Figure 10 demonstrates its enhancement in path quality and stability. Together, they illustrate that the proposed method can simultaneously achieve fast planning and high-quality path generation in complex 3D obstacle environments.

The shorter paths obtained through the proposed method are mainly caused by two coupled mechanisms. First, the environment feedback RRT stage suppresses blind expansion by increasing obstacle boundary and bridge sampling when the collision rate is high and by strengthening local obstacle repulsion when the obstacle density increases. This reduces ineffective branches and provides a lower redundancy feasible path for post-processing. Second, PAWOA does not perturb all path nodes globally. It selects only high-value key nodes, constrains them within safety-aware corridors, repairs collision-causing individuals, and gradually shifts the optimization emphasis from feasibility and smoothness to path length compression. As a result, the final path is not restricted to the original sampled nodes and can become shorter while preserving collision-free feasibility.

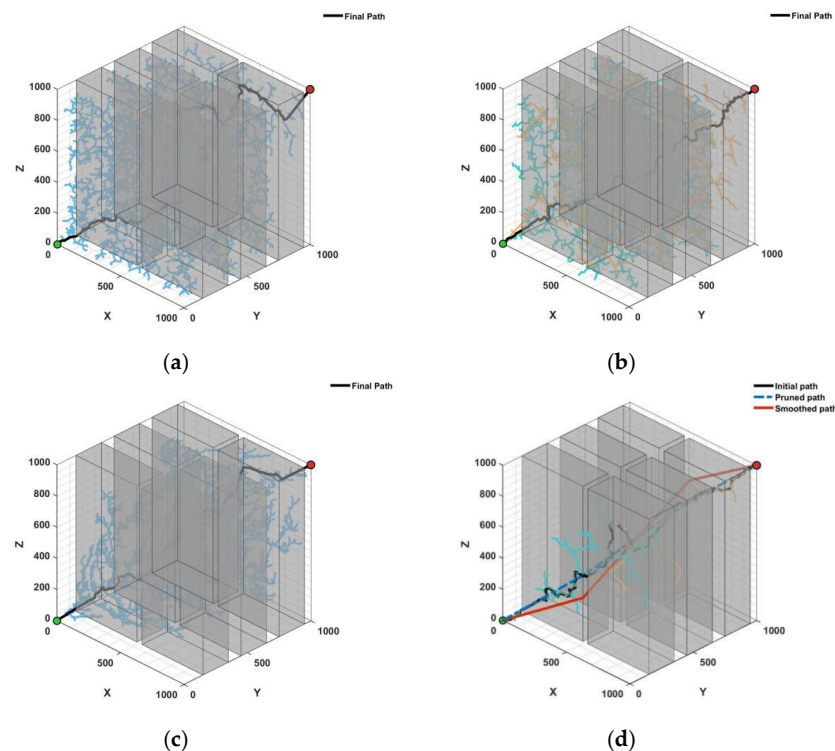


Figure 5. Environment 1. Subfigures (a–d) correspond to Algorithms a, b, c, and d, respectively.

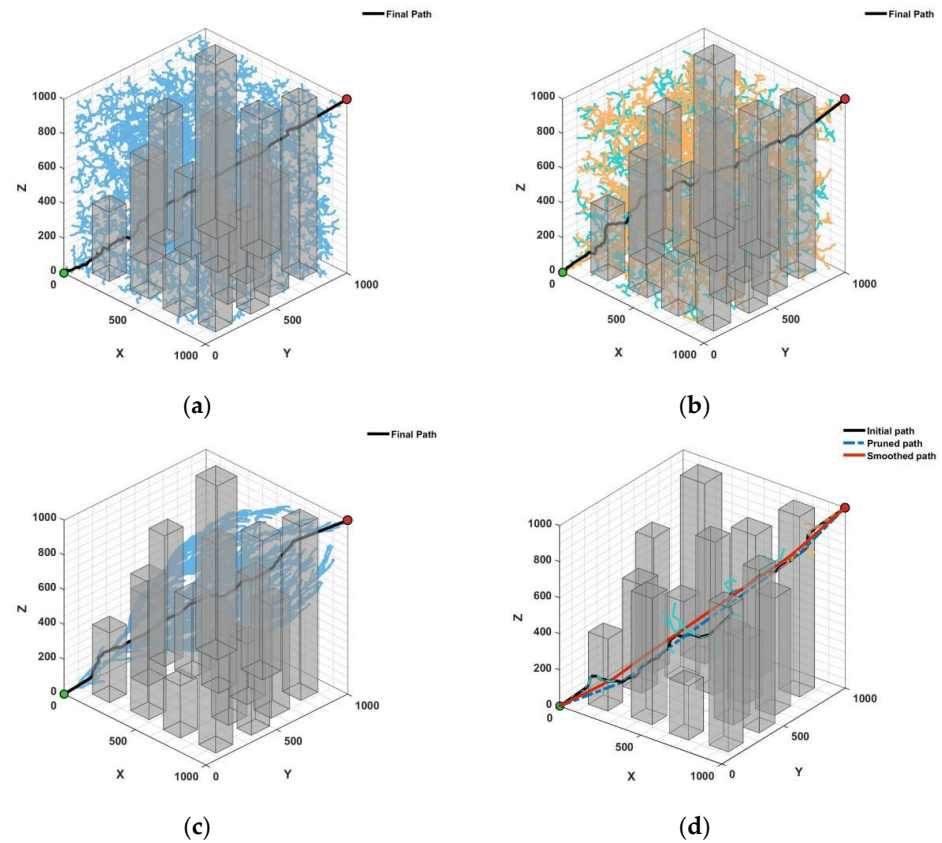


Figure 6. Environment 2. Subfigures (a–d) correspond to Algorithms a, b, c, and d, respectively.

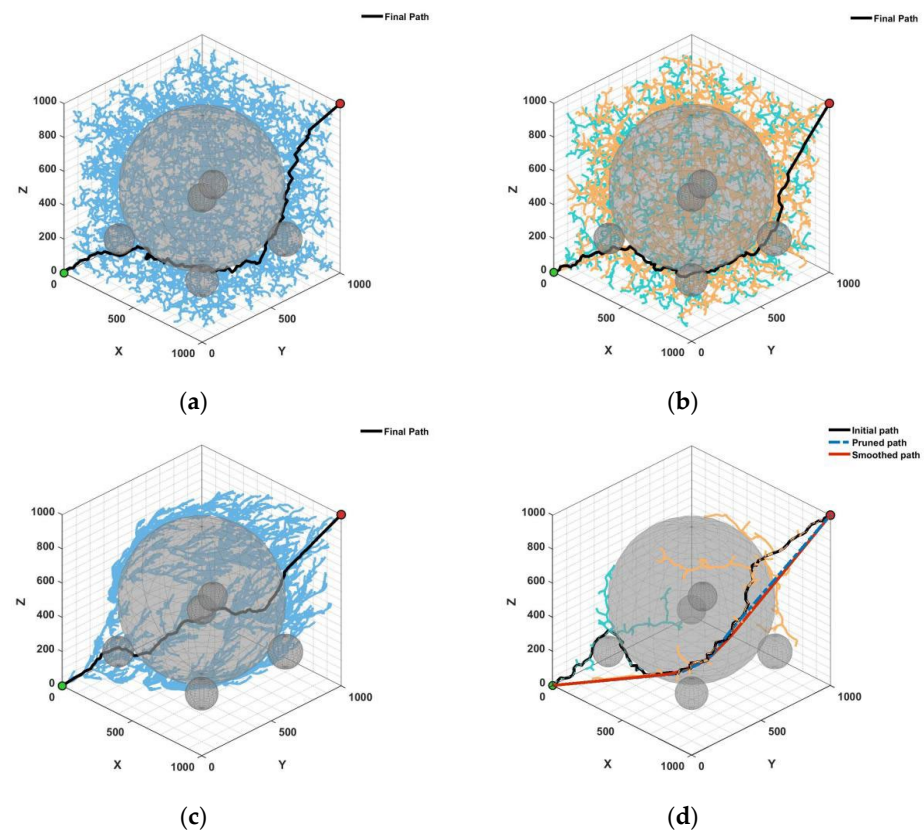


Figure 7. Environment 3. Subfigures (a–d) correspond to Algorithms a, b, c, and d, respectively.

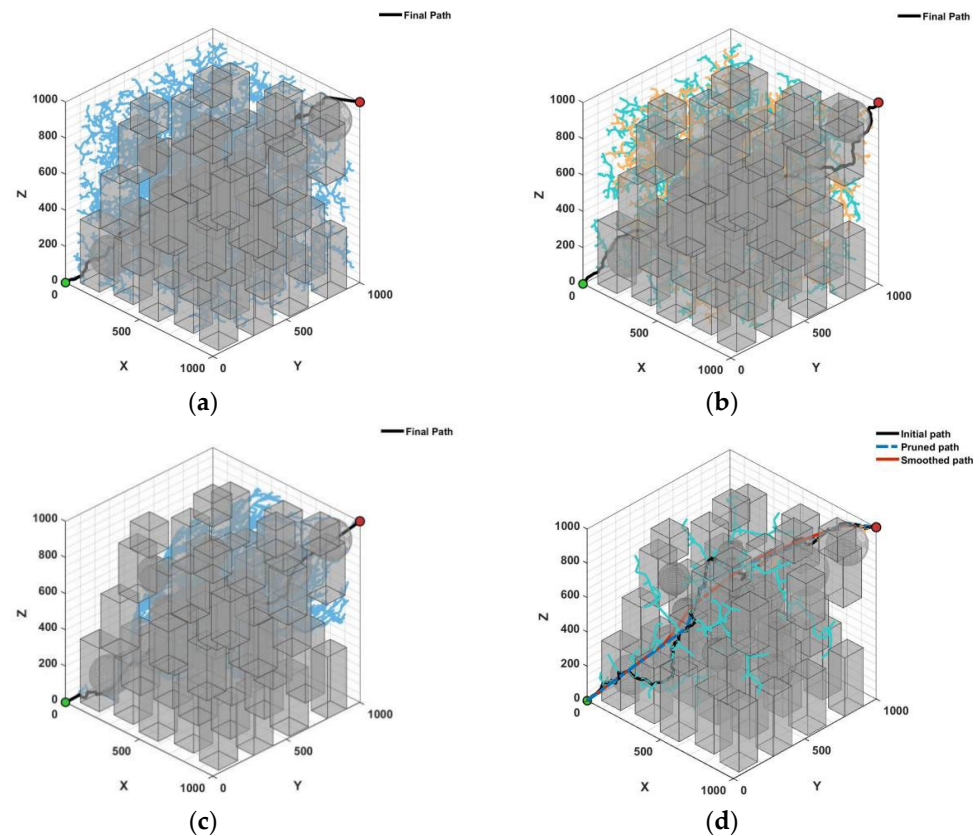


Figure 8. Environment 4. Subfigures (a–d) correspond to Algorithms a, b, c, and d, respectively.

Table 3. Comparison of path planning results among algorithms.

Algorithm	Environment	Planning Time/s	Path Length	Success Rate/%
<i>a</i>	1	2.27	2997.31	53
<i>b</i>		1.74	2742.11	100
<i>c</i>		3.55	2149.49	100
<i>d</i>		0.62	2044.69	100
<i>a</i>	2	5.55	2247.64	100
<i>b</i>		4.04	2236.98	100
<i>c</i>		5.74	1897.61	100
<i>d</i>		0.67	1763.10	100
<i>a</i>	3	3.78	2529.05	100
<i>b</i>		2.61	2652.13	100
<i>c</i>		3.27	2283.87	100
<i>d</i>		0.52	1977.53	100
<i>a</i>	4	4.94	2497.81	46
<i>b</i>		2.61	2442.35	73
<i>c</i>		5.75	2117.21	84
<i>d</i>		1.58	1902.97	97

The success rate in Environment 4 is lower than 100% because this environment contains dense obstacles and narrow passages. In such spaces, sampling-based planners can still fail within a finite iteration budget even when they are probabilistically complete in theory, because valid samples near narrow passages occupy a small proportion of the total configuration space and repeated expansions may be blocked by obstacle boundaries. The proposed bridge sampling, boundary sampling, local density-aware step

reduction, and collision repair improve this probability but cannot eliminate finite budget failures completely.

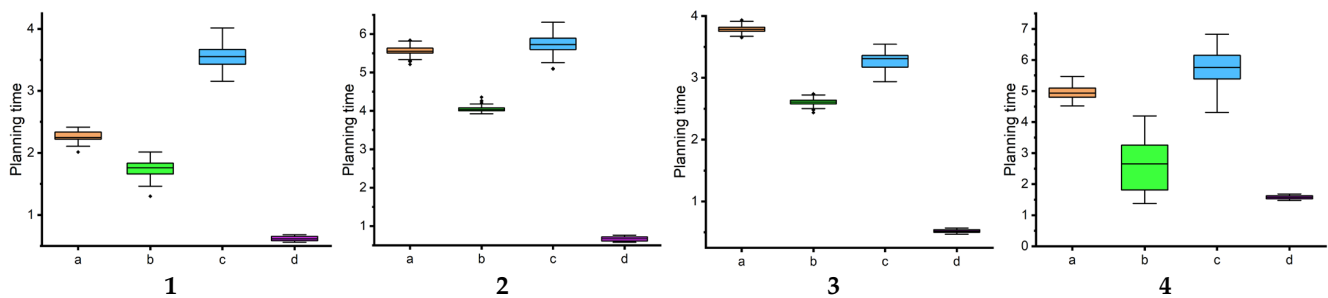


Figure 9. Box plot of planning time. Subfigures (1–4) correspond to Environment 1, 2, 3, and 4, respectively.

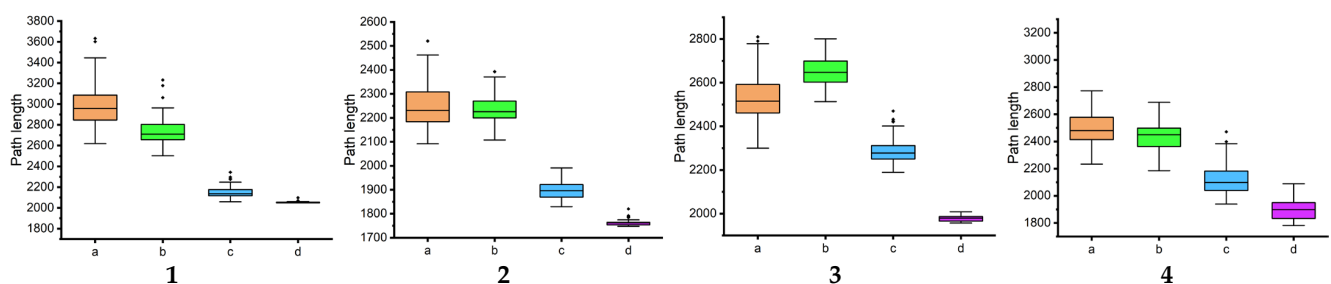


Figure 10. Box plot of path length. Subfigures (1–4) correspond to Environment 1, 2, 3, and 4, respectively.

To further validate the performance of each module, using Environment 1 as an example, the ablation experiments were conducted in two parts, with each algorithm running independently 50 times under identical conditions. First, the PAWOA component was removed for comparative validation. The results are shown in Figure 11.

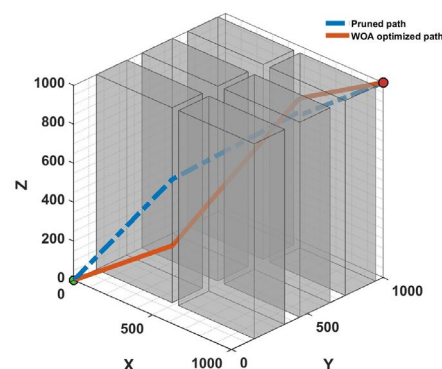


Figure 11. Collision map.

Compared to the pruning path, adding the PAWOA component reduced the average path length from 2202.59 to 2054.53, with significant improvements in route optimization and substantial reduction in path length. Meanwhile, the average execution time of PAWOA remained at 0.079 s, showing minimal increase within acceptable limits. A stepwise ablation experiment was then conducted: using the optimized RRT algorithm d_1 as baseline, the collision-rate-driven hybrid sampling strategy, local obstacle-density-aware expansion, and backtracking safe step size were sequentially removed to develop algorithms d_2 , d_3 , and d_4 without additional post-processing. Changes in planning time, path length, success rate, and iteration count were analyzed, as shown in Table 4. Results demonstrate

that performance progressively declines with each removed RRT enhancement module. Removal of environmental feedback sampling increased both planning time and iteration count, indicating its role in improving effective sampling efficiency and accelerating search convergence. Further removal of density-aware expansion raised initial path length to 2479.63, confirming its effectiveness in enhancing exploration performance in densely obstructed areas. Subsequent elimination of collision avoidance mechanisms and local reconnection improved path length and iteration count further, demonstrating its capacity to enhance expansion success rates while reducing path redundancy. Overall, the three RRT improvement modules significantly enhance search efficiency, initial path quality, and planning robustness.

Table 4. Comparison of path planning results.

Algorithm	Planning Time/s	Path Length	Iterations
d_1	0.52	2116.82	796
d_2	0.81	2268.47	934
d_3	1.12	2479.63	1086
d_4	1.37	2724.58	1224

5. Conclusions

This study addresses the blind sampling, frequent expansion blocking, high initial path redundancy, and collision-prone post-processing of traditional RRT algorithms in complex three-dimensional obstacle environments. An environment feedback hybrid sampling bidirectional RRT path planning method integrated with a path-aware repair whale optimization algorithm is proposed. The method improves both the initial path search stage and the post-processing optimization stage, enhancing search efficiency, path quality, and environmental adaptability in complex three-dimensional spaces. The main conclusions are as follows:

First, in the initial path search stage, a collision-rate-driven environment feedback hybrid sampling strategy and a local obstacle-density-aware expansion mechanism are designed. The algorithm adaptively switches the sampling mode according to the cumulative expansion obstruction of the random tree, improving the match between sampled points and environmental structure. It also fuses the sampling direction, goal direction, opposite tree attraction direction, and obstacle repulsion direction and combines adaptive dual step sizes with a backtracking safe step size. As a result, the random tree maintains fast expansion in open regions while reducing the step size and strengthening obstacle avoidance in dense obstacle regions, thereby reducing invalid expansions and improving the efficiency of initial feasible path search.

Second, in the path post-processing stage, a path-aware repair whale optimization algorithm is proposed. Through key node selection, only nodes that substantially affect path length, smoothness, and safety margin are optimized. Path corridor constraints and elite opposition initialization improve the quality of the initial population. A dynamically weighted fitness function, collision-aware repair, variable neighborhood local search, and elastic band local refinement improve the feasible solution ratio and optimization capability under limited population size and limited iterations. Combined with local B-spline smoothing, the algorithm further improves path continuity and smoothness while satisfying collision-free constraints. The proposed algorithm was implemented and evaluated in simulation experiments. Results demonstrated that across all four environments, it achieved the shortest average planning time and path length among all compared algorithms, with an average planning time of 0.85 s and an average path length of 1922.07 units. Ablation experiments further validated the effectiveness of key implementation modules:

the PAWOA module reduced the average path length from 2202.59 to 2054.53. When the RRT improvement module was sequentially removed, planning times increased from 0.52 s to 0.81 s, 1.12 s, and 1.37 s respectively; path lengths grew from 2116.82 to 2268.47, 2479.63, and 2724.58 units; and iteration counts rose from 796 to 934, 1086, and 1224. These findings confirm that the combined use of these modules significantly enhances search efficiency, initial path quality, and planning stability. The experimental environments in this study are static and contain only spherical and axis-aligned cuboid obstacles; dynamic obstacles, perception uncertainties, and non-axis-aligned objects are not considered. Future work will focus on extending the proposed method to dynamic environments and validating the algorithm's performance on real UAV platforms.

Author Contributions: Methodology, Z.W. supervision, D.X. and Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets presented in this article are not readily available because the data are part of an ongoing study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yang, X.; Fang, H.; Li, J.; Xue, Q. Human factors engineering for human-cyber-physical systems collaboration in intelligent manufacturing. *China Mech. Eng.* **2023**, *34*, 1710–1722+1740. (In Chinese)
2. Liu, S.; Zhang, D.; Ming, Z.; Liu, M.; Liu, Z.; Chen, Q.; Zhang, J.; Wu, J.; Yan, T. Remote brain-controlled UAV system based on brain-computer interface and human-machine closed loop. *Acta Armamentarii* **2024**, *45*, 3191–3203. (In Chinese)
3. Niu, Y.X.; Bai, Z.F.; Wang, H. Design and kinematic analysis of cable-driven underactuated manipulator on spacecraft. *Mech. Based Des. Struct. Mach.* **2024**, *53*, 3887–3903. [[CrossRef](#)]
4. Ma, C.L.; Provost, J. Design-to-test approach for programmable controllers in safety-critical automation systems. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6499–6508.
5. Gao, W.; Han, C.; Chen, Z. Altitude-adaptive modulation for covert and anti-jamming in terahertz UAV communications. In Proceedings of the 2022 IEEE Globecom Workshops (GC Wkshps), Rio de Janeiro, Brazil, 4–8 December 2022; pp. 1586–1591.
6. Sandino, J.; Hawary, A.F.; Gonzalez, F. A review of UAV path-planning algorithms and obstacle avoidance methods for remote sensing applications. *Remote Sens.* **2024**, *16*, 4019. [[CrossRef](#)]
7. Wang, H.; Yu, Y.; Yuan, Q. Application of Dijkstra algorithm in robot path-planning. In Proceedings of the 2011 Second International Conference on Mechanic Automation and Control Engineering, Hohhot, China, 15–17 July 2011; pp. 1067–1069.
8. Chen, J.Q.; Li, M.Y.; Su, Y.S.; Li, W.; Lin, Y. Direction constraints adaptive extended bidirectional A* algorithm based on random two-dimensional map environments. *Robot. Auton. Syst.* **2023**, *165*, 104430. [[CrossRef](#)]
9. Chang, J.; Dong, N.; Li, D.; Ip, W.H.; Yung, K.L. Skeleton extraction and greedy algorithm-based path planning and its application in UAV trajectory tracking. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 4953–4964.
10. Zhang, Y.; Chen, J. Three-dimensional UAV omnidirectional path planning algorithm based on urban obstacle environment. *Algorithms* **2026**, *19*, 393. [[CrossRef](#)]
11. Fan, L.; Zhang, H.; Xu, Z.; Lv, M.; Hu, J.; Zhao, C.; Liu, X. UAV dense-obstacle avoidance algorithm based on safe flight corridors. *J. Northwest. Polytech. Univ.* **2022**, *40*, 1288–1296. (In Chinese) [[CrossRef](#)]
12. Ye, L.; Duan, J.; Yang, Z.; Zou, X.; Chen, M.; Zhang, S. Collision-free motion planning for the litchi-picking robot. *Comput. Electron. Agric.* **2021**, *185*, 106151.
13. Pei, Y.; Zhai, S. Active obstacle-avoidance path planning for autonomous vehicles in multiple scenarios under icy and snowy road conditions. *J. Northeast. Univ. (Nat. Sci.)* **2025**, *46*, 1–11. (In Chinese)
14. Liu, X.; Kang, L.; Shan, Z.; Su, C.; Liu, Y. Manipulator path planning based on the RTSR-RRT* algorithm. *Chin. J. Sci. Instrum.* **2025**, *46*, 65–73. (In Chinese) [[CrossRef](#)]
15. Tu, H.; Deng, Y.; Li, Q.; Song, M.; Zheng, X. Improved RRT global path planning algorithm based on bridge test. *Robot. Auton. Syst.* **2024**, *171*, 104570.
16. Feng, Z.; Zhou, L.; Qi, J.; Hong, S. DBVS-APF-RRT*: A global path planning algorithm with ultra-high speed generation of initial paths and high optimal path quality. *Expert Syst. Appl.* **2024**, *249*, 123571. [[CrossRef](#)]

17. Ganesan, S.; Ramalingam, B.; Mohan, R.E. A hybrid sampling-based RRT* path planning algorithm for autonomous mobile robot navigation. *Expert Syst. Appl.* **2024**, *258*, 125206. [[CrossRef](#)]
18. Xiong, Z.; Hou, W.; Wang, C.; Chen, H. Classified-RRT*: Node classification RRT*-based algorithm with improved solution performance and convergence rate. *Expert Syst. Appl.* **2025**, *296*, 129118.
19. Uwacu, D.; Yammanuru, A.; Nallamotu, K.; Chalasani, V.; Morales, M.; Amato, N.M. HAS-RRT: RRT-based motion planning using topological guidance. *IEEE Robot. Autom. Lett.* **2025**, *10*, 6223–6230. [[CrossRef](#)]
20. Wei, W.; Wei, K.; Zhang, J. A novel method of path planning for an intelligent agent based on an improved RRT* called KDB-RRT. *Sensors* **2025**, *25*, 7545. [[CrossRef](#)] [[PubMed](#)]
21. Liu, N.; Hu, Z.; Wei, M.; Guo, P.; Zhang, S.; Zhang, A. Improved A* algorithm incorporating RRT* thought: A path planning algorithm for AGV in digitalised workshops. *Comput. Oper. Res.* **2025**, *177*, 106993. [[CrossRef](#)]
22. Zhu, W.; Qiu, G. Path planning of intelligent mobile robots with an improved RRT algorithm. *Appl. Sci.* **2025**, *15*, 3370. [[CrossRef](#)]
23. Jung, Y.; Oh, B.S. DC-RRT*: Dubins-guided curvature RRT* for 3D path planning of unmanned aerial vehicles. *IEEE Access* **2025**, *3*, 148936–148948.
24. Guan, T.; Han, Y.; Kong, M.; Wang, S.; Feng, D.; Yang, W. An improved artificial potential field with RRT star algorithm for autonomous vehicle path planning. *Sci. Rep.* **2025**, *15*, 16982. [[CrossRef](#)] [[PubMed](#)]
25. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Technical Report; Iowa State University: Ames, IA, USA, 1998.
26. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2997–3004.
27. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [[CrossRef](#)]
28. Yang, G.; Sun, Z. A lightweight and efficient improved RRT* algorithm for global path planning in complex environments. *Appl. Sci.* **2026**, *16*, 5002. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.