Pose-Smoothing A* Path Planning Method for Three-Dimensional Environments

Yihao Lu*, Chenhao Zhu

Nanjing Tech University, Nanjing, China.

3247582732@qq.com

Abstract. The traditional A* algorithm ensures global optimality under admissible heuristics but ignores pose continuity in 3D, causing abrupt orientation changes. We extend A* with quaternion-based pose modeling, dynamic weighting, and Slerp refinement. The node state combines position and pose, while the heuristic adaptively balances distance and orientation. A coarse-to-fine strategy rapidly generates an approximate path and then refines poses using Slerp. MATLAB and ROS validations demonstrate smoother trajectories and near-zero terminal error at moderate computational cost. Although the algorithm no longer strictly satisfies admissibility, it achieves a practical balance between efficiency and pose smoothness, providing a promising solution for UAV and robot navigation.

Keywords: A* Algorithm; Three-Dimensional Path Planning; Quaternion; Slerp; Dynamic Weighting; Pose Smoothing; UAV Navigation.

1. Introduction

In complex three-dimensional environments, autonomous navigation and path planning are critical challenges for unmanned aerial vehicles (UAVs) and mobile robots [1][2]. Applications such as disaster rescue and urban logistics require rapid generation of collision-free and executable trajectories. However, neglecting pose continuity often leads to abrupt angular velocity changes that reduce control stability.

Since its introduction in 1968, the A* algorithm has been widely applied in robotic path planning [5]. While its heuristic ensures optimality and efficiency, in 3D spaces the cubic growth of grid cells and branching neighbors raises computational cost. More importantly, standard A* considers only positional information and ignores pose continuity, producing abrupt pose changes. Existing smoothing or curve-fitting methods alleviate this but increase complexity and do not explicitly incorporate pose constraints.

To address these issues, this paper proposes an improved A* algorithm with quaternion-based pose modeling and Spherical Linear Interpolation (Slerp). Node states combine position and pose, and the heuristic integrates distance and angular difference with a distance-based dynamic weighting strategy. A coarse-to-fine approach first generates a rapid approximate path, then applies Slerp to ensure smooth pose transitions.

The proposed method is validated through MATLAB simulations and ROS experiments. Results show that, compared with standard A*, it reduces pose discontinuities while maintaining efficiency, yielding smoother and more executable trajectories [10][11][12]. This provides a practical approach to UAV and robot path planning in complex 3D environments, balancing efficiency and executability.

2. Theoretical Foundations

2.1 Standard A Algorithm*

The A* algorithm is one of the most classic heuristic search methods, introduced by Hart et al. in 1968[5]. Its core lies in the introduction of a cost function:

$$f(n) = g(n) + h(n) \tag{1}$$

where $g(n)$ denotes the accumulated cost from the start to node n, and $h(n)$ is the heuristic estimate from n to the goal. The node with the smallest $f(n)$ is expanded until the goal is reached. In a three-dimensional grid, the functions are typically defined as:

$$g(n) = \sqrt{(x - x_s)^2 + (y - y_s)^2 + (z - z_s)^2} \quad (2)$$

$$h(n) = \sqrt{(x - x_g)^2 + (y - y_g)^2 + (z - z_g)^2} \quad (3)$$

Where (x_s, y_s, z_s) represents the start point and (x_g, y_g, z_g) represents the goal point coordinates. The optimality of **A*** relies on the **admissibility** （$\forall n,\ 0 \le h(n) \le h*(n)$）and **consistency/monotonicity**（$\forall n, m,\ h(n) \le c(n,m) + h(m)$） of the heuristic function. When these two conditions are met, A* guarantees the minimum-cost path from the start to the goal[5]. **However, when the heuristic function is weighted or when non-positional terms (such as pose angles) are added, these conditions are typically violated, and optimality can no longer be ensured.** When the heuristic is weighted or includes non-positional terms such as pose angles, these conditions are violated and optimality is lost.

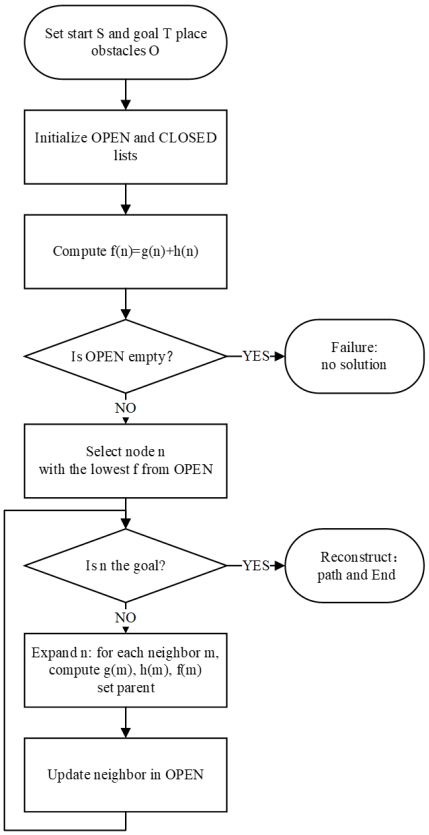

Figure 2.1 Standard A* Algorithm Flowchart

Figure 2.1 shows the flowchart of the standard A* algorithm. Starting from the initial conditions, the OPEN and CLOSED lists are initialized and $f(n)$=$g(n)$+$h(n)$ is computed. The node with the smallest $f(n)$ is expanded until the goal is reached; otherwise, its neighbors are updated and added to OPEN. If OPEN becomes empty, no feasible path exists. This process illustrates how A* progressively approaches the optimal path based on the cost function.

2.2 Definition of Quaternions and Pose Representation

Quaternions are a mathematical tool proposed by Hamilton[6], and are generally written as

$$q = w + xi + yj + zk, w, x, y, z \in R \quad (4)$$

When the normalization condition $\| q \| = 1$ is satisfied, a quaternion can represent a three-dimensional rotation. Suppose the rotation angle is θ and the rotation axis is the unit vector $u = (ux, uy, uz)$; the corresponding quaternion representation is given by

$$q = cos\ \frac{\theta}{2} + (u_x i + u_y j + u_z k) sin\ \frac{\theta}{2} \quad (5)$$

For a three-dimensional vector p, its rotation can be achieved by

$$p' = q\,p\,q^{-1} \tag{6}$$

where p is expressed as a pure quaternion $(0, \mathrm{p})$, and q^{-1} denotes the conjugate of quaternion. Quaternions avoid the gimbal lock problem, provide numerical stability, and are well suited for pose representation.

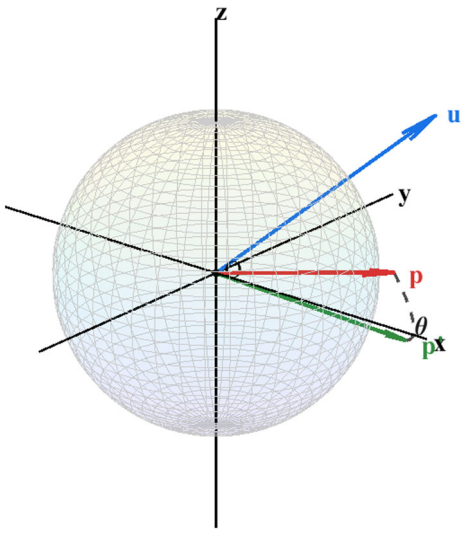

Figure 2.2 Geometric Illustration of Rotation Represented by a Quaternion

In Figure 2.2, the red arrow denotes the original vector **p**, the green arrow denotes the rotated vector **p′**, and the blue arrow denotes the rotation axis **u**. The dashed arc represents the rotation trajectory with the rotation angle **θ** indicated. The black dot denotes the origin **O**, and the gray spherical surface illustrates that the rotation occurs on the unit sphere, where the vector moves along the spherical trajectory around the rotation axis while preserving its magnitude.

2.3 Spherical Linear Interpolation (Slerp)

In pose interpolation problems, linear interpolation (LERP) often leads to non-uniform angular velocity, resulting in unnatural trajectory execution. To address this issue, Shoemake proposed the spherical linear interpolation (Slerp) method[7]. Let two unit quaternions be $q_0, q_1 \in S^3$, with the angle between them defined as

$$\theta = arccos\,(q_0 \cdot q_1) \qquad q_0 \cdot q_1 = w_0w_1 + x_0x_1 + y_0y_1 + z_0z_1 \tag{7}$$

then Slerp is defined as

$$Slerp(q_0, q_1; t) = \frac{sin\,((1-t)\theta)}{sin\;\theta} q_0 + \frac{sin\,(t\theta)}{sin\;\theta} q_1, t \in [0,1] \tag{8}$$

Slerp possesses several advantages, including endpoint consistency, interpolation points lying on the unit sphere, constant angular velocity, and approximation to LERP under small-angle conditions. These properties ensure smooth transitions of pose[7].

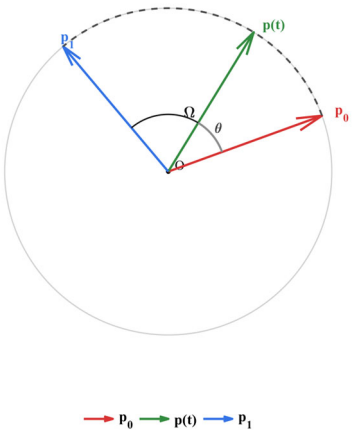

Figure 2.3 Geometric Illustration of Slerp Interpolation

In Figure 2.3, the red vector represents the starting point p_0, and the blue vector represents the ending point p_1, with the circle center **O** as the origin. The interpolation point $p(t)$ (green) lies on the great-circle arc from p_0 to p_1, forming an angle $\theta = t\Omega$ with p_0, where Ω denotes the total angle. This geometric relationship illustrates that Slerp achieves smooth pose transitions and uniform angular velocity variation by interpolating along the great-circle arc on the unit sphere.

2.4 Comparison with LERP

LERP (Linear Interpolation) refers to **linear interpolation**, which essentially performs a straight-line interpolation between two points or vectors according to a given ratio. When a quaternion q_0,q_1 are regarded as four-dimensional vectors, LERP is defined as

$$Lerp(q_0, q_1; t) = (1-t)q_0 + tq_1, t \in [0,1] \quad (9)$$

LERP is simple but results in non-uniform angular velocity, and its interpolation points deviate from the unit sphere; in contrast, Slerp remains on the sphere, ensuring smooth and natural transitions[7].

3. Improved A* Method Based on Slerp

Standard A* ensures global optimality with admissible heuristics but neglects pose continuity, leading to abrupt angular changes. Incorporating pose terms and dynamic weighting breaks these conditions, so the method is regarded as a weighted A* variant prioritizing efficiency–smoothness balance over strict optimality.

3.1 State Modeling

In the traditional A* algorithm, the node state is typically represented by the positional coordinates $p = (x, y, z)$. To simultaneously account for pose information, we extend the state of each node to a combination of position and pose. Specifically, the state of a node is expressed as:

$$n = (p, q), p \in R^3, q \in S^3 \quad (10)$$

where p denotes the position vector and q denotes the pose represented by a quaternion. Quaternion q is a commonly used representation of three-dimensional rotation, capable of avoiding the gimbal lock problem inherent in Euler angles, while also providing higher numerical stability in computation. Thus, each node not only contains its current position but also its associated pose information, enabling the search process to ensure both spatial feasibility of the path and continuity of the pose.

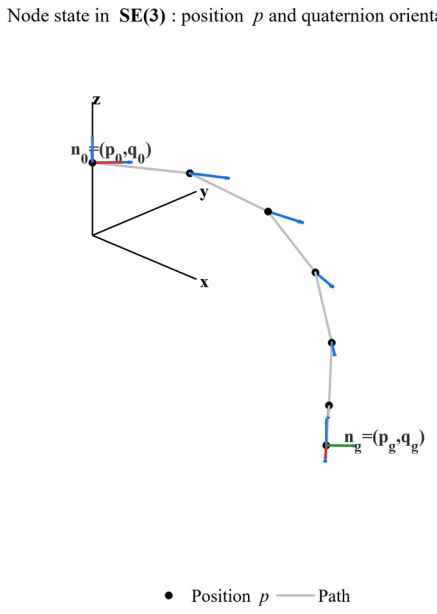

Figure 3.1 Illustration of Node State Modeling

Figure 3.1 shows the state modeling of a node in three-dimensional space, where the position is represented by coordinates and the pose is represented by a directional arrow corresponding to the quaternion. This illustrates the concept of extending the state from the Euclidean space $R3$ to the special Euclidean group $SE(3)$.

3.2 Incorporating Pose Distance into the Heuristic Function

In the standard A* algorithm, the heuristic function $h(n)$ typically considers only the positional distance, which is calculated using the Euclidean distance:

$$h(n) = \sqrt{(x - x_g)^2 + (y - y_g)^2 + (z - z_g)^2} \tag{11}$$

However, in path planning that considers pose continuity, a heuristic function based solely on position is insufficient. To ensure that the planning process optimizes not only the position but also the pose transition, we incorporate the pose distance $d_R(q, q_g)$ into the heuristic function, yielding the new form:

$$h(n) = \alpha \cdot d_T(p, p_g) + \beta \cdot d_R(q, q_g) \tag{12}$$

where $d_T(p, p_g)$ is the Euclidean distance between positions, $d_R(q, q_g) = arccos\,(|\, q \cdot q_g \,|)$ is the angular difference between quaternions[8], and α and β are dynamically adjusted weighting coefficients used to balance the influence of position and pose. Through this formulation, the heuristic function not only accounts for positional optimization but also ensures smooth pose transitions along the path.

With this improvement, the A* algorithm is able to perform optimization simultaneously in both the positional space and the pose space, thereby avoiding abrupt pose changes.

3.3 Dynamic Weighting Switching Mechanism

To prioritize “reaching the target position as quickly as possible” in the early stages of the search, and “pose alignment and smoothness” in the final stages, this paper adjusts the weighting based on the normalized distance dd to the target[1]:

$$\alpha(d) = \alpha_0 \cdot \frac{d}{d_{max}}, \beta(d) = \beta_0 \cdot \left(1 - \frac{d}{d_{max}}\right) \tag{13}$$

(13)

Here, d_{max} denotes the initial maximum distance, while α_0 and β_0 are the initial weighting coefficients that control the relative influence of position and pose in the heuristic function. In the early stage of the search, the positional influence dominates; as the search progresses, the influence of pose gradually increases, thereby guiding the A* algorithm to place greater emphasis on pose alignment and smoothness as it approaches the target.

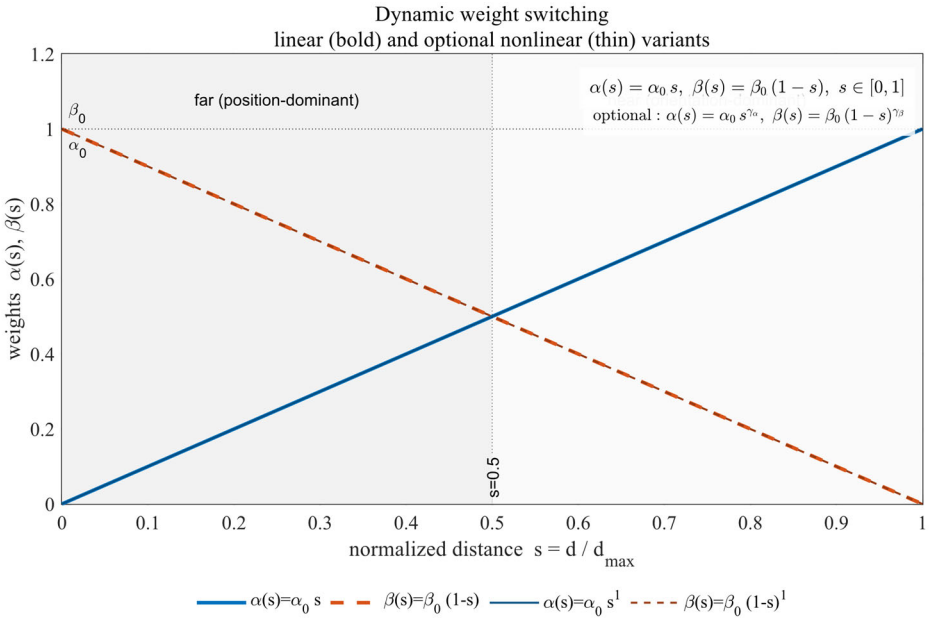

Figure 3.3 Dynamic Weighting Curve

In Figure 3.3, the horizontal axis shows the normalized distance $s=d/d_{max}$, and the vertical axis the heuristic weights $\alpha(s)$, $\beta(s)$. Thick lines denote linear forms $\alpha(s)=\alpha_0 s$, $\beta(s)=\beta_0(1-s)$, and thin lines indicate optional nonlinear variants. When $s<0.5$, position weight dominates and the search favors path length; when $s>0.5$, pose weight increases, emphasizing terminal alignment. This design maintains efficiency in distant regions and smooth pose transitions near the goal.

3.4 Coarse-to-Fine Two-Stage Strategy

To balance efficiency and smoothness, a "two-stage（$coarse \rightarrow fine$）":

Coarse stage: A larger step size and weaker pose constraints are used for rapid search, resulting in an approximate optimal path skeleton;

Fine stage: Slerp pose interpolation is applied between adjacent key points (q_i, q_{i+1}) of the coarse path, followed by uniform resampling based on arc length or time, yielding a final trajectory with uniform angular velocity and continuous pose [7][9]. If necessary, short-range local smoothing is performed on the spatial positions, ensuring feasibility is maintained.

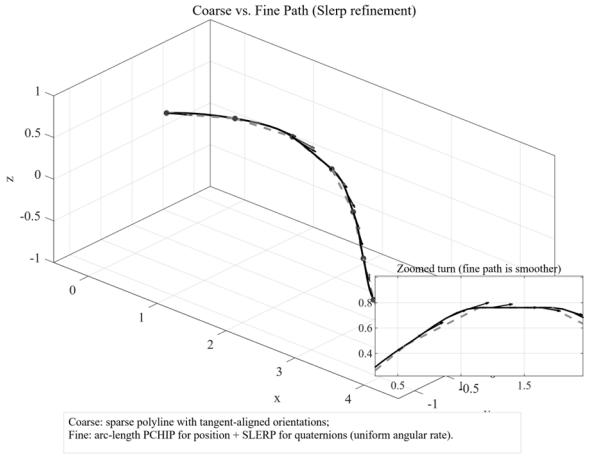

Figure 3.4 Coarse/Fine Path Comparison

In Figure 3.4, the gray dashed line represents the sparse coarse path, while the black solid line represents the smoothed path after refinement using Slerp. The enlarged view shows that at the corners, the coarse path exhibits noticeable sharp turns, whereas the fine path achieves a continuous and natural transition, demonstrating the smoothness advantage of the coarse-to-fine strategy.

3.5 Improved Algorithm Design

By incorporating pose modeling and Slerp, the improved A* accounts for both distance and pose continuity. The maximum start–goal distance is used to dynamically adjust weights $\alpha(d)$ and $\beta(d)$, with α decreasing and β increasing near the target to enhance alignment. RefineWithSlerp then interpolates between key nodes to ensure smooth transitions and uniform angular velocity [7].

Improved A* Algorithm Pseudo-code

```
Improved_AStar_SE3(start=(p_s,q_s), goal=(p_g,q_g)):
  OPEN ← {start}; CLOSED ← ∅
  g[start] ← 0; d_max ← d_T(p_s,p_g)
  h[start] ← α *d_T(p_s,p_g)+ β *d_R(q_s,q_g)
  while OPEN ≠ ∅:
    n ← argmin f(x), x∈OPEN; move n to CLOSED
    if ( ‖ p_n−p_g ‖ ≤ ε_p) and (d_R(q_n,q_g) ≤ ε_q):
      return RefineWithSlerp(ReconstructPath(parent,n))
    for each neighbor m of n:
      if m ∈ CLOSED: continue
      tentative_g ← g[n]+step_cost(n,m)
      update α,β by normalized distance d
      h[m] ← α *d_T(p_m,p_g)+ β *d_R(q_m,q_g)
      if (m ∉ OPEN) or (tentative_g < g[m]):
        parent[m] ← n; g[m] ← tentative_g
        insert_or_update(OPEN,m)
  return failure

RefineWithSlerp(path):
  for each segment (p_i,q_i)→(p_{i+1},q_{i+1}):
    interpolate p(t), q(t) by linear/Slerp
    append to refined path
  return refined
```

RefineWithSlerp(path) denotes performing Slerp-based refinement interpolation on the final path to realize the coarse-to-fine optimization.

3.6 Improved Algorithm Flowchart

The flowchart of the improved A* algorithm is shown in Figure 3.6. Compared to the standard A* algorithm, this method introduces pose distance and dynamic weighting switching in the cost calculation step, and adds Slerp-based interpolation optimization in the path output stage.

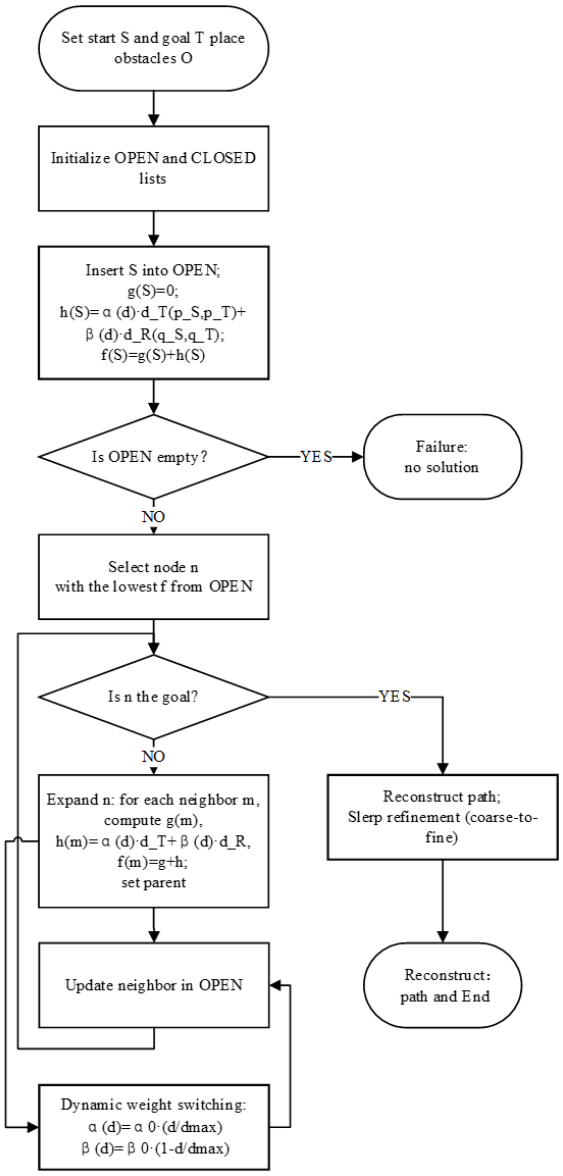

Figure 3.6 Flowchart of the Improved A* Algorithm

Figure 3.6 shows the flowchart of the improved A* algorithm. After defining start, goal, and obstacles, the OPEN and CLOSED sets are initialized and the cost function is computed. The node with the smallest $f(n)$ is iteratively expanded until the goal condition is met, after which Slerp interpolation generates the final smooth path. If not, neighbors are expanded with dynamic weighting updates until a feasible path is found or OPEN is empty. This highlights the design concept of dynamic weighting, pose-guided search, and Slerp refinement for improved continuity and executability.

4. Experiments and Results Analysis

In this chapter, the Slerp-based improved A* algorithm is validated in a 3D voxel environment in MATLAB, with standard A* as the baseline due to its representativeness in highlighting pose improvements. This choice limits the evaluation scope, as methods like D*, Theta*, or RRT are not included, which will be explored in future work. Experiments focus on path length, efficiency, pose smoothness, and terminal alignment, with local enlarged views illustrating differences in high-curvature segments.

4.1 Experimental Environment and Parameter Settings

The experiments were conducted in MATLAB R2023b on an Intel i7/16 GB Ubuntu 20.04 system. The 50×50×20 voxel map (Figure 4.1) contains floating obstacles, narrow passages, tunnels, and

clutter. The start and goal are set at (3,3,2) and (47,46,18) with a 35° yaw constraint. An **18-connected** neighborhood allows axial, face-diagonal, and skew moves. Both algorithms use the same cost function: standard A* considers only positional distance, while the improved version adds pose angular difference with dynamic weighting.The two algorithms share the same cost function, with their main differences summarized as follows:

Standard A*: The heuristic function considers only positional distance;

Improved A*: The heuristic function incorporates both positional distance and pose angular difference, with a dynamic weighting mechanism based on normalized distance. The final path is further refined by Slerp resampling with equal arc length to guarantee pose continuity.

The evaluation metrics include path length L, expanded nodes $Nexp$, search time T, pose smoothness $\sigma \Delta \theta$, and terminal pose error e.

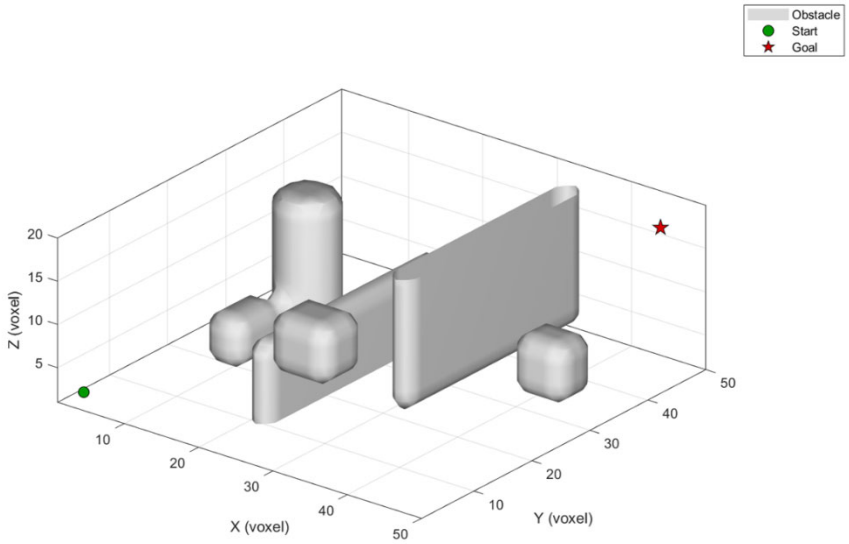

Figure 4.1 Three-Dimensional Grid Environment

As shown in Figure 4.1, the map is composed of multiple cylinders, several cubic blocks, and a vertical wall. Obstacles are mainly distributed in the central and right regions of the environment, forming a typical three-dimensional test scenario with both narrow passages and detour requirements.

4.1 Path Planning Comparison

As shown in Figure 4.2, this experiment compares the performance of the standard A* and the improved A* in the same three-dimensional grid environment. The quantitative results of the two methods are listed in Table 4.1. It can be observed that the improved A* produces a shorter path (reduced by approximately 6.4%), with pose error reduced to nearly zero, but incurs a computational cost about 2.4 times that of the standard A*.

Table 4.1 Performance Comparison between Standard A* and Improved A*

Method	Path Length L	Nodes Expanded N_{exp}	Search Time T/ ms	Standard Deviation of Pose Increments $\sigma_{\Delta\theta}$/ (°)	Terminal Pose Error e_q / (°)
Standard A*	74.88	18,679	2830.2	23.7	56.4
Improved A* (Slerp)	70.10	44,040	6921.4	4.1	<0.1

Figure 4.2 compares path and pose of the two methods. The improved A* shows smoother trajectories with continuous pose alignment, while the standard A* produces sharp peaks and abrupt turns, especially in narrow passages. Although it incurs about 2.4× higher cost, the improved A* yields shorter paths, smoother transitions, and nearly zero terminal error, greatly enhancing executability.

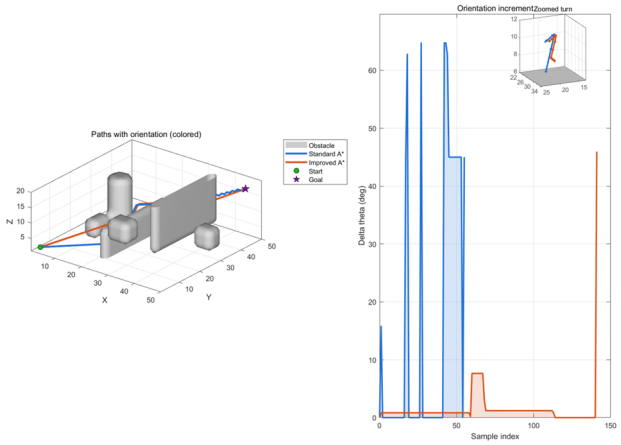

Figure 4.2 Comparison of Path Planning Results between Standard A* and Improved A*

4.2 Analysis of Dynamic Weighting Switching Effects

To evaluate the effectiveness of the dynamic weighting mechanism, four strategies are compared: **S0** (position term only), **S1** (fixed weight), **S2** (linear switching), and **S3** (power-law switching). The evaluation metrics include path length L, number of expanded nodes N_{exp}, search time T, standard deviation of pose angle increments $\sigma\Delta\theta$, and terminal pose error e_{q}.

Table 4.2 presents the median results. It can be seen that S0/S1 achieve high efficiency but suffer from large pose errors; S2 improves smoothness but at excessively high cost; S3 provides a more balanced trade-off between efficiency and accuracy.

Further sensitivity analysis (Figure 4.3) shows that when $\gamma, \delta \in [1,2]$, **S3** can maintain a relatively small terminal pose error under different parameters, with most cases achieving an error below $90°$. The recommended parameters are $\gamma \approx \delta = 1.5$, which strike a good balance between convergence speed and pose executability.

Table 4.2 Ablation Study Results of Dynamic Weighting Strategies

Strategy	Path Length L	Nodes Expanded N_{exp}	Search Time T / ms	Standard Deviation of Pose Increments $\sigma_{\Delta\theta}$ / (°)	Terminal Pose Error e_{q} / (°)
S0 pos-only	54.01	4,123	628.4	11.0	117.1
S1 fixed	54.80	1,253	195.8	11.6	127.4
S2 linear	54.06	26,890	4382.3	7.6	119.4
S3 power (γ=1.5, δ=1.5)	54.05	33,380	5088.0	9.0	117.1

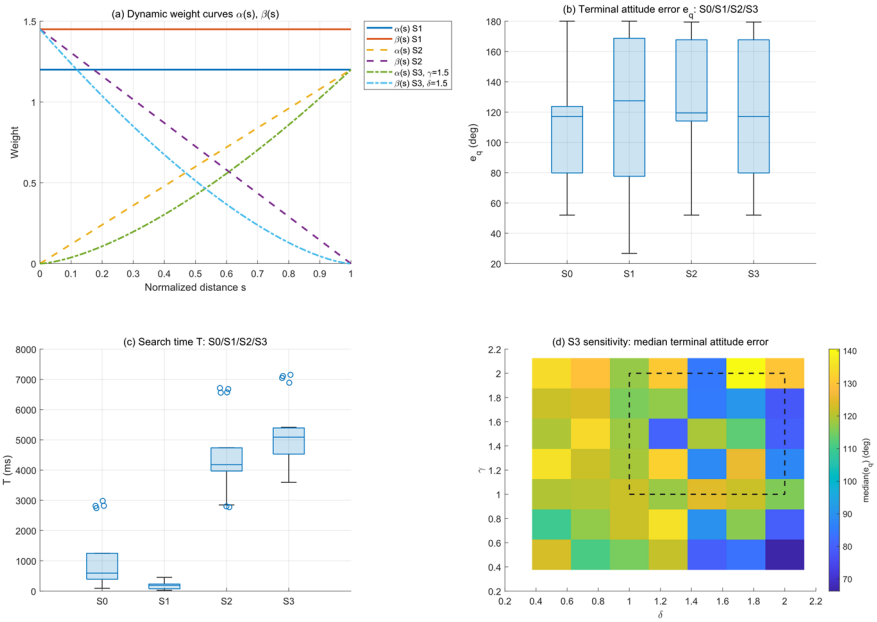

Figure 4.3 Comparison and Sensitivity Analysis of Dynamic Weighting Switching

Figure 4.3 presents the comparison and sensitivity analysis of the dynamic weighting strategies:(a) variation curves of $\alpha(s), \beta(s)$ under different strategies; (b) boxplot of terminal pose error e_q; (c) boxplot of search time T; (d) heatmap of terminal error for S3 power-law switching under the (γ, δ) grid (the dashed box indicates the recommended parameter region).

4.3 Effect of the Coarse-to-Fine Strategy (Equal Budget and Controlled Variables)

To ensure fairness, an equal-budget and controlled-variable design was adopted. At the coarse level, a low-resolution global path is generated, and at the fine level, refinement is performed within the corridor, with only the downsampling ratio and corridor radius adjusted.

Under **equal-time conditions** (Figure 4.4(a)), the Pareto frontier of the coarse-to-fine (C2F) strategy is overall superior to that of the improved A* algorithm, achieving lower terminal pose error $e_q$and smaller pose $\sigma\Delta\theta$at the same computational cost.

Under **equal-resolution conditions** (Figure 4.4(b)), the (T, eq) frontier of C2F shifts noticeably to the lower-left, indicating that it can achieve lower error within the same computation time, or significantly shorten planning time at the same error level. The locally enlarged view further shows that this advantage is primarily concentrated in the low-error region, demonstrating better practical applicability.

In summary, the C2F strategy simultaneously improves efficiency, accuracy, and pose smoothness without increasing computational budget, offering a superior solution for 3D path planning.

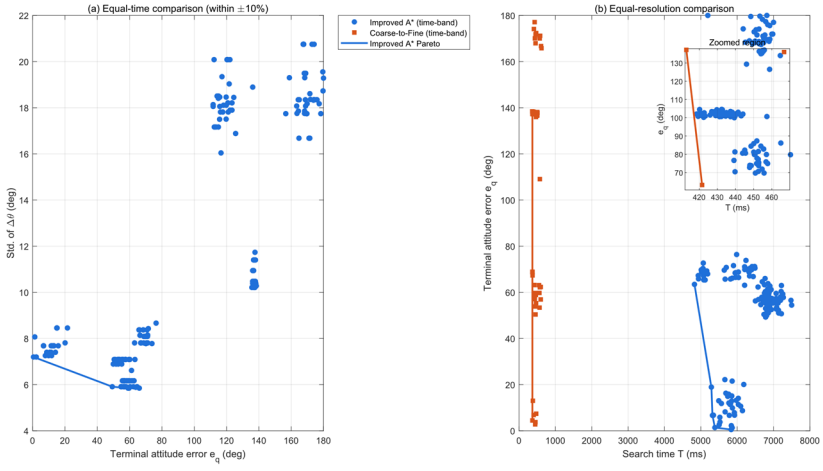

Figure 4.4 Pareto Comparison of Coarse-to-Fine and Improved A* under Equal Budget

4.4 ROS Validation

In the ROS 2 environment, the SE(3) trajectory generated by the improved A* algorithm was validated using RViz. The voxel map and trajectory data exported from MATLAB were loaded, with obstacles, start, and goal points marked. As shown in Figure 4.5, the planned trajectory successfully navigated the complex environment while maintaining safe clearance from voxel obstacles and avoiding collisions. The path exhibited natural turns and smooth transitions without abrupt pose jumps. Comparison with MATLAB results demonstrated strong consistency, confirming the executability and reliability of the proposed method for UAV control experiments.

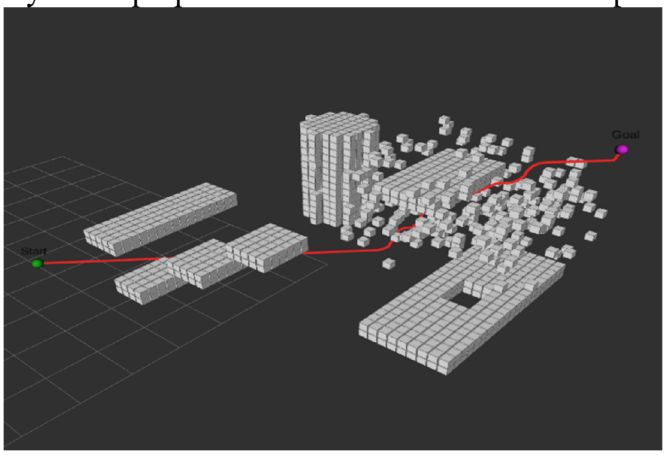

Figure 4.5 RViz Validation

5. Conclusions and Future Work

This paper presents an improved 3D A* algorithm with quaternion pose modeling and Slerp interpolation. By incorporating pose into the heuristic and adopting dynamic weighting, the method balances efficiency and smoothness but no longer strictly satisfies admissibility, making it a weighted A* variant emphasizing executability. Experiments show that terminal pose error decreases from about 56° to below 0.1°, with smoother transitions despite higher node expansion. ROS/RViz validation confirmed reliable navigation in complex environments. Although only standard A* was used as baseline, this setting highlights improvements in pose smoothness, while future work will extend comparisons to more advanced planners.

References

[1] LaValle S M. Planning algorithms[M]. Cambridge university press, 2006.

[2] Choset H, Lynch K M, Hutchinson S, et al. Principles of robot motion: theory, algorithms, and implementations[M]. MIT press, 2005.

[3] Koenig S, Tovey C, Smirnov Y. Performance bounds for planning in unknown terrain[J]. Artificial Intelligence, 2003, 147(1-2): 253-279.

[4] Dellin C, Srinivasa S. A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors[C]//Proceedings of the international conference on automated planning and scheduling. 2016, 26: 459-467.

[5] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths[J]. IEEE transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.

[6] Hamilton W R. Ii. on quaternions; or on a new system of imaginaries in algebra[J]. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1844, 25(163): 10-13.

[7] Shoemake K. Animating rotation with quaternion curves[C]//Proceedings of the 12th annual conference on Computer graphics and interactive techniques. 1985: 245-254.

[8] Huynh D Q. Metrics for 3D rotations: Comparison and analysis[J]. Journal of Mathematical Imaging and Vision, 2009, 35(2): 155-164.

[9] Park F C, Ravani B. Smooth invariant interpolation of rotations[J]. ACM Transactions on Graphics (TOG), 1997, 16(3): 277-295.

[10] Gao F, Lin Y, Shen S. Gradient-based online safe trajectory generation for quadrotor flight in complex environments[C]//2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2017: 3681-3688.

[11] Zhou B, Gao F, Pan J, et al. Robust real-time uav replanning using guided gradient-based optimization and topological paths[C]//2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020: 1208-1214.

[12] Zhang Y, Hu Y, Lu J, et al. Research on path planning of mobile robot based on improved theta* algorithm[J]. Algorithms, 2022, 15(12): 477.