Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.055883

ARTICLE

Path Planning of Multi-Axis Robotic Arm Based on Improved RRT∗

Juanling Liang[1], Wenguang Luo[1,2,*] and Yongxin Qin[1]

[1]School of Automation, Guangxi University of Science and Technology, Liuzhou, 545006, China

[2]Key Laboratory of AI and Information Processing of Education Department of Guangxi, Hechi University, Hechi, 546300, China

*Corresponding Author: Wenguang Luo. Email: wgluo@gxust.edu.cn

Received: 09 July 2024 Accepted: 02 September 2024 Published: 15 October 2024

ABSTRACT

An improved RRT∗ algorithm, referred to as the AGP-RRT∗ algorithm, is proposed to address the problems of poor directionality, long generated paths, and slow convergence speed in multi-axis robotic arm path planning. First, an adaptive biased probabilistic sampling strategy is adopted to dynamically adjust the target deviation threshold and optimize the selection of random sampling points and the direction of generating new nodes in order to reduce the search space and improve the search efficiency. Second, a gravitationally adjustable step size strategy is used to guide the search process and dynamically adjust the step-size to accelerate the search speed of the algorithm. Finally, the planning path is processed by pruning, removing redundant points and path smoothing fitting using cubic B-spline curves to improve the flexibility of the robotic arm. Through the six-axis robotic arm path planning simulation experiments on the MATLAB platform, the results show that the AGP-RRT∗ algorithm reduces 87.34% in terms of the average running time and 40.39% in terms of the average path cost; Meanwhile, under two sets of complex environments A and B, the average running time of the AGP-RRT∗ algorithm is shortened by 94.56% *vs*. 95.37%, and the average path cost is reduced by 55.28% *vs*. 47.82%, which proves the effectiveness of the AGP-RRT∗ algorithm in improving the efficiency of multi-axis robotic arm path planning.

KEYWORDS

Multi-axis robotic arm; path planning; improved RRT∗ algorithm; dynamic target deviation threshold; dynamic step size; path optimization

1 Introduction

Robotic arms play an important role in many industrial and service sectors, where they are capable of performing a variety of complex tasks such as assembly, welding, and material handling. Robotic arm path planning is a key problem in robotic arm control, which involves finding the optimal path of a robotic arm between a given start and target position for efficient, safe, and precise motion.

Path planning algorithms can be generally categorized into the following four groups: graph-based search algorithms [1,2], artificial potential field methods [3,4], intelligent algorithms [5–7], and sampling-based algorithms [8–11]. Graph-based search algorithms are prone to huge computational burdens in high-dimensional spaces, limiting their applications. Artificial potential field algorithms are often trapped in local optima and have difficulty avoiding interference from obstacles. The convergence

speed and stability of intelligent algorithms are affected by parameter settings and initial value selection. Sampling-based algorithms are suitable for path planning in high-dimensional space, which has the characteristics of convenience and efficiency, easy to constrain and probabilistic completeness. Sampling-based algorithms are mainly categorized into probabilistic roadmap (PRM) [12], and rapid search random tree (RRT) [13]. Among them, the RRT algorithm is favored for its fast search speed, powerful exploration ability, and probabilistic completeness. However, its common problems include long generation paths, inefficiency, and the possibility of failing to find feasible paths in complex environments.

To address the shortcomings of RRT algorithms, many scholars have made improvements to RRT algorithms. Adaptive RRT-Connect (ARRT-Connect) [14] effectively solves the narrow-channel planning problem by combining the advantages of the faster bidirectional RRT (RRT-Connect) and rapidly-exploring random vines (RRV). An elastic band-based rapidly-exploring random tree (EB-RRT) [15] realizes real-time optimal motion planning for mobile service robots in dynamic environments by introducing a hierarchical planning framework and dynamic optimization strategies. Rapid exploration of random tree stars (RRT∗) [16] effectively improves the quality and efficiency of path planning by separating the expansion and optimization processes using a dual-tree structure while combining the original RRT and modified RRT∗ strategies. The RRT-Connect algorithm combined with region partitioning (RRT-Connect-RP) [17] realizes the generation of welding paths for arc welding robots in ship subassembly welding by zone partitioning technique, which improves the efficiency of the robotic system and realizes intelligent production. Adaptive Stepsize RRT [18,19] improves the efficiency and accuracy of path planning through an adaptive stepsize strategy. A stream-based VF-RRT∗ (SVF-RRT∗) [20] effectively solves the problem of safe and energy efficient path planning for unmanned surface vehicles (USVs) in large-scale ocean environments with spatially variable currents by means of heuristic intervals, biased sampling, and tree pruning techniques. Gaussian mixture regression-rapid exploration of random tree stars (GMR-RRT∗) [21] achieves fast and efficient path planning for mobile robots in different environments by combining Gaussian mixed regression learning and the path planning capability of fast exploring random trees. Bidirectional assisting metric-rapid exploration of random tree stars (Bi-AM-RRT∗) [22] significantly improves the motion planning efficiency and path planning efficiency of mobile robots in dynamic environments. Reinforcement learning-rapid search random tree (RL-RRT) [23] provides an effective long-term planning solution for sampling-based motion planning for kinematic dynamics by combining a deep reinforcement learning strategy and a reachability estimator. Rapidly-exploring random trees based on heuristic probability bias-goal (PBG-RRT) [24] improves search efficiency and obstacle avoidance for robotic path planning in 3D space by combining heuristic probability and bias objective factors. Neural RRT∗ [25] effectively solves the problems of initial solution sensitivity and slow convergence of traditional RRT and its variants by using a non-uniform sampling distribution predicted by the convolutional neural network (CNN) model to guide the sampling process. A reliable and robust rapidly-exploring random tree (R2-RRT∗) [26] enhances the fast exploratory randomized tree algorithm for off-road automated ground vehicle task planning in uncertain terrain environments by introducing state and task mobility reliability metrics. vehicle task planning robustness and reliability.

This article explores further improvements to the existing RRT∗ algorithm. First, the adaptive bias probability sampling strategy is introduced to optimize the selection of random sampling points and dynamically adjust the target deviation threshold according to the environment, so as to accelerate the convergence speed of the algorithm and reduce the generation of redundant nodes. Secondly, the gravitationally adjustable step size strategy is adopted to dynamically adjust the step size according to the target gravitational force and obstacle information in order to improve the search efficiency

and enhance the adaptability to different environments. Finally, based on the node reconnection strategy, collision detection is performed after reconnecting the initial path nodes to remove redundant nodes, and three times B-spline curve is performed for path smoothing to optimize the obstacle avoidance path.

2 Robotic Arm Modeling and Collision Detection

2.1 Kinematic Modeling of a Multi-Axis Robotic Arm

The D-H (Denavit-Hartenberg) parametric method [27], which analyzes the motion of a robotic arm in space by establishing kinematic equations, is a commonly used approach for kinematic modeling. It is divided into two forms, Standard DH method (SDH) and Modified DH method (MDH). The MDH method assumes that the axes of all rotating joints are aligned with the Z-axis to simplify the calculation of the kinematic equations. Therefore, the MDH method is chosen to build the D-H parametric model of the robot.

Set T_i^{i-1} as the transformation matrix from coordinate system $\{i\}$ to coordinate system $\{i-1\}$. $c\theta_i$ denotes $\cos\theta_i$, $s\theta_i$ denotes $\sin\theta_i$, $c\alpha_{i-1}$ denotes $\cos\alpha_{i-1}$, $s\alpha_{i-1}$ denotes $sin\alpha_{i-1}$. Its general expression is shown in Eq. (1):

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

where θ_i: around the Z_i-axis, the angle of rotation from the X_{i-1}-axis to the X_i-axis;

d_i-around the Z_i-axis, the distance from the X_{i-1}-axis to the X_i-axis of rotation;

α_i-around the X_i-axis, the angle of rotation from the Z_i-axis to the Z_{i+1}-axis;

a_i-around the X_i-axis, the distance from the Z_i-axis to the Z_{i+1}-axis of rotation.

After determining the connection coordinate system and the corresponding connection parameters, the forward kinematics equations of the manipulator can be calculated as follows:

$$ {}^{0}_{S}T = \begin{bmatrix} \mathrm{n}_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

where S is the number of arm axes, n, o, a denote the spatial attitude, respectively; and the p denotes the spatial position.

2.2 Collision Detection

In the field of collision detection, there are several commonly used methods, including axis-aligned bounding box (AABB), oriented bounding box (OBB), cylinder enveloping box, and sphere enveloping box. For the robotic arm linkage, the model simplification using a cylindrical bounding box is an effective method because its shape is similar to that of a cylinder. The sphere-surrounding box is especially popular in application scenarios requiring fast response due to the simplicity of its computational process. The mathematical expression of the sphere-surrounding box can be simplified to the problem of calculating the center coordinates of the sphere and the radius of the sphere, as

shown in Eq. (3):

$$R = \left\{(x,\ y,\ z)\ \middle|\ (x - o_x)^2 + (z - o_x)^2 \le r^2\right\} \tag{3}$$

where the center coordinate of the enclosing sphere is $(o_x,\ o_y,\ o_z)$ and r denotes the radius of the sphere.

Take the cylinder enclosing the box approach to envelope the robotic arm linkage and the sphere enclosing the box approach to envelope the obstacle, the model is simplified as shown in Fig. 1. When computing collision detection, simplifying linkages into line segments and adding the radius of the cylindrical bounding box to that of the spherical bounding box transforms the robot-obstacle collision detection into a segment-sphere intersection problem. This reduces computational overhead, thereby enhancing solving efficiency.

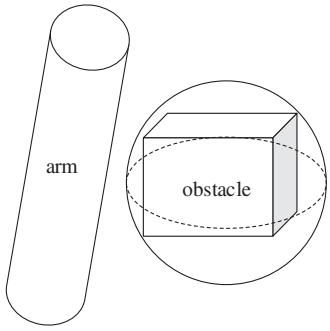


Figure 1: Simplified collision detection model

3 RRT∗ Algorithm

The main concept of the RRT algorithm is to randomly grow a tree from the initial point until it reaches the target point, as illustrated in Fig. 2. According to the randomly generated sampling state point q_{rand}, find the nearest node q_{near} to q_{rand} in the generated tree, and make q_{near} grow one step in the direction of q_{rand}, so as to generate a new node q_{new}, and if the path from q_{near} to the new node has no collision with the obstacles, add the new node to the tree, and vice versa, eliminate the q_{new} node. The above process is repeated until the tree expands to the goal point q_{goal}, then the algorithm ends.

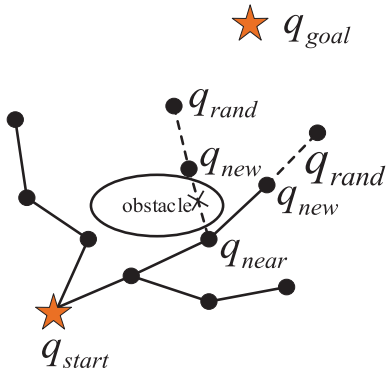
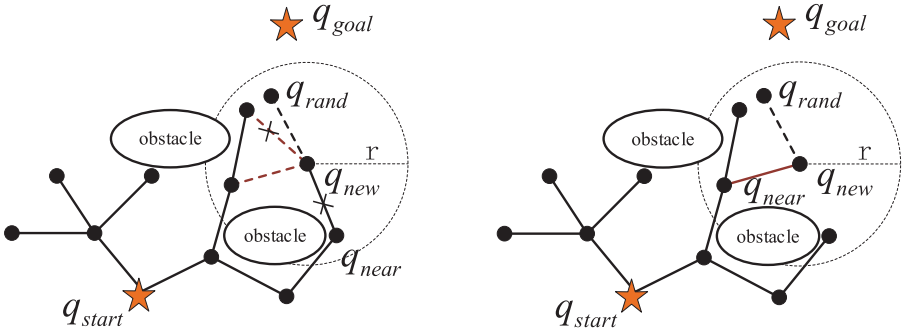


Figure 2: RRT expansion method

The RRT∗ algorithm improves the performance of the RRT algorithm by introducing a reselection and reconnection mechanism for the parent nodes, and Fig. 3 demonstrates the effect of this improvement. It is important to note that the RRT∗ algorithm enhances performance by optimizing the configuration space (the joint space of the robotic arm) instead of the task space (such as Cartesian space). The RRT∗ algorithm updates the parent node by reducing the path cost to the starting point and, after this adjustment, it replans the paths for all adjacent nodes. This algorithm increases the probability of obtaining an optimal or near-optimal path in the configuration space. Thus, the effect in Fig. 3 reflects the impact of mapping obstacles in the configuration space on path planning.



(a) Reselect the parent node (b) Rewire the node

Figure 3: RRT∗ expansion method

4 Improvement of the RRT∗ Algorithm

4.1 Adaptive Biased Probabilistic Sampling Strategy

The RRT∗ algorithm inherits the similar sampling method of RRT, but suffers from a lack of goal-orientation, leading to the generation of redundant nodes, which in turn reduces the overall sampling efficiency. To tackle this problem, a target bias strategy is introduced, directing the sampling toward the target to minimize the creation of unnecessary nodes and enhance the efficiency of path planning. The idea of this strategy is shown in Eq. (4). In this equation, a random distribution of the target bias is used, which is not uniformly random but enhances the probability of sampling near the target by directing the random samples towards the target region.

$$q_{rand} = \begin{cases} Random, & rand\,() > m \\ q_{goal}, & otherwise \end{cases} \tag{4}$$

where *Random* is a randomly generated sample point, q_{goal} is a target point, *rand* () is a random number between 0 and 1, m is the target deviation threshold.

When *rand* () $> m$, random sampling is performed; conversely, the sampling point will be directly equal to the target point. However, there are problems with this method: If there is an obstacle close to the target point, setting the sampling point to be identical to the target point could result in a collision; if there is no obstacle, random sampling will result in the generation of redundant nodes. To address this issue, dynamically adjust the value of m based on obstacle information.

Specifically, the initial value of m is first set to a preset m_0. When attempting to expand in the direction of the target, if the resulting new node does not come into contact with an obstacle, the target deviation threshold is made to gradually increase to $m_{\max}$. Where m_0 and $m_{\max}$ are obtained from

Eqs. (5) and (6). If a new node collides with an obstacle, m_0 and $m_{\max}$ can be adjusted using equations Eqs. (7) and (8), which can effectively reduce the value of m_0 and increase the value of $m_{\max}$ at the same time. Through the above methods, it ensures that the planning process effectively utilizes the goal guidance while also fully considering the uncertainty of the environment, and ultimately realizes the optimized path generation.

$$m_0 = kd \tag{5}$$

$$m_{\max} = nm_0 (n > 1) \tag{6}$$

where k is an empirical coefficient (usually between 0.1 and 0.5), d is the distance between the starting point and the target point, and n is the magnification factor.

$$m_0{}' = m_0 (1 - \alpha) \tag{7}$$

$$m_{\max}{}' = m_{\max} (1 + \beta) \tag{8}$$

where α and β are adjustment factors less than 1, respectively.

Fig. 4 shows a 2D simulation comparison between the RRT∗ algorithm and the RRT∗ algorithm that incorporates an adaptive biased probabilistic sampling strategy, referred to as the A-RRT∗ (Adaptive RRT∗) algorithm, which can be seen to generate fewer nodes than the RRT∗ algorithm.

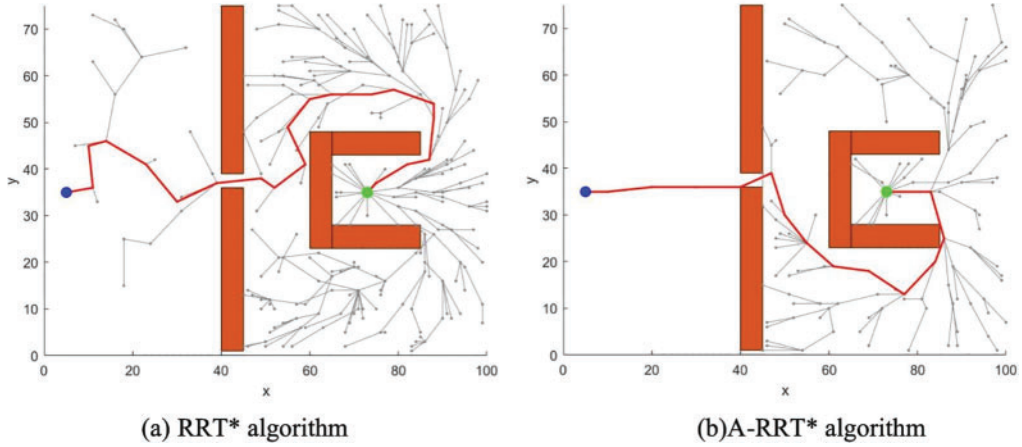


Figure 4: Comparison of RRT∗ algorithm before and after improvement

4.2 Gravitationally Adjustable Step Size Strategy

In the RRT∗ algorithm, the step size is a key factor that directly affects the efficiency of the pathfinding process. When the environment is free of obstacles, increasing the step size reduces the search time. However, in the presence of obstacles, too large a step size is prone to cause the algorithm to fall into a narrow region, leading to difficulties in finding the target point and feasible paths; while too small a step size produces redundant nodes and reduces the search efficiency. Therefore, the gravitationally adjustable step-size strategy takes the gravitational coefficient and obstacles into account to achieve dynamic adjustment of the step-size, which can optimize the performance of the algorithm, and its specific principles are as follows:

The new node formula for the RRT∗ algorithm is:

$$q_{new} = q_{near} + \rho \frac{(q_{rand} - q_{near})}{\|q_{rand} - q_{near}\|} \tag{9}$$

where ρ is the step size.

When the idea of target gravity is added, its new nodal formula is:

$$q_{new} = q_{near} + \rho_1 \frac{(q_{rand} - q_{near})}{\|q_{rand} - q_{near}\|} + \rho_2 \frac{(q_{goal} - q_{near})}{\|q_{goal} - q_{near}\|} \tag{10}$$

where ρ_1 is the step size of the extension in the direction of the random point and ρ_2 is the step size of the extension in the direction of the target point.

Order:

$$\begin{cases} \rho_1 = \rho \\ \rho_2 = k_p \rho \end{cases} \tag{11}$$

where k_p is the gravitational coefficient.

The expression for the gravitational coefficient is:

$$k_p = \bigcup_{i=1}^{N} \left(\sqrt{(q_{near,\,x} - x_i)^2 + (q_{near,\,y} - y_i)^2} - (R_{i,\,z} + r) \right) \tag{12}$$

where N is the total number of obstacles, $(x_i,\ y_i)$ are the coordinates of the center of the circle of obstacle i, $R_{i,\,z}$ is the radius of the closest obstacle to the node q_{near}, and r is the radius of the robot arm cylinder.

The initial step size is preset to be ρ_0, which is related to the step size ρ as in Eq. (13):

$$\begin{cases} \rho = \rho_0 + \varepsilon\rho_0, & \rho_{\max} = 2\rho_0, \rho_1 < \rho_2 \\ \rho = \dfrac{\rho_0}{2}, & otherwise \end{cases} \tag{13}$$

where ε is the incremental factor.

The selection of ε is typically based on specific application scenarios and environment characteristics. For example, in the case of a complex environment with many obstacles, a smaller ε may be needed to avoid collisions due to too large a step size; while in the case of a more open environment, the ε can be increased appropriately to accelerate the search. Setting the maximum step size to $2\rho_0$ is to ensure that the step size growth will not be too large, leading to instability or overextension during the search process. In many path planning algorithms, the step size needs to be set in a way that balances the speed of exploration with the safety of the path. The setting of the maximum step size can effectively limit the growth of the step size so that the path search is not too coarse or imprecise. The parameter $2\rho_0$ is typically chosen based on experience to ensure that the step size's maximum expansion remains within reasonable limits relative to the starting step size. This ensures that larger areas can be explored quickly during the search, while the step size can be adjusted to be smaller when approaching obstacles or when a precise path is required, thus improving the safety of the path.

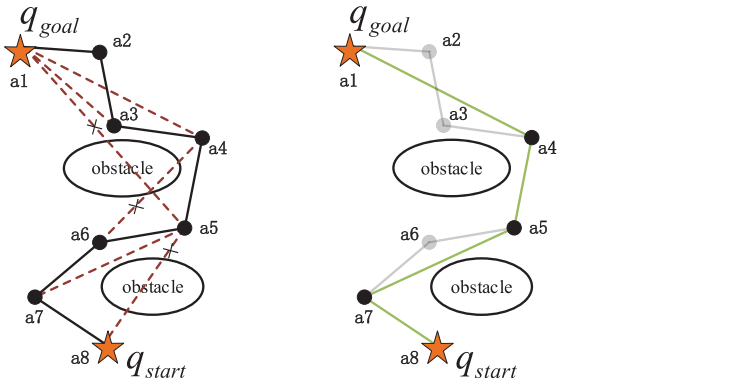
From Eq. (12), when node q_{near} is farther away from the obstacle, k_p is greater than 1, then $\rho_1 < \rho_2$, the step length grows according to $\rho = \rho_0 + \varepsilon\rho_0$, and the maximum step length is set to $2\rho_0$. The growth of the step length makes the search speed faster. Conversely, when node q_{near} is closer to the

obstacle, the gravitational coefficient decreases, and the step length is set to $\frac{\rho_0}{2}$, and the reduction of the step length helps to avoid the obstacle in a finer way and ensures the safety and effectiveness of path planning. With this strategy, the algorithm is able to integrate gravitational guidance and obstacle distribution so as to intelligently adjust the step size for more efficient path planning.

4.3 Path Optimization Strategy

The RRT∗ algorithm, when applied to 3D space for path exploration, yields a continuous line formed by a chain of discrete points. However, this path may be curved and unsmooth and contain many redundant nodes, which makes it difficult for the robotic arm to operate smoothly and thus reduces the service life of the robotic arm. Therefore, path optimization is needed, which is divided into the removal of redundant nodes and smoothing.

The basic principle of redundant node removal is that the nodes on the path are labeled in traversal order as a1, a2,..., an, where a1 is the target node and an is the start node. Starting from a1, it tries to establish a connection with a2, a3,...., an to establish a connection. During the connection attempt, collision detection is performed for each connection. If a collision occurs while trying to connect a1 with the gth node ($1 < g \leq n$), a1 establishes a connection directly with the g-1th node and ignores the gth node. The g-1th node is used as the new connection point, and collision detection and redundant node removal continue for subsequent nodes. After each redundant node removal, the path is updated and the connection verified by collision detection is retained. The algorithm ends when all nodes are collision detected and no collision occurs, or when the starting node is reached. The final optimized path obtained has less number of nodes while satisfying collision detection. As shown in Fig. 5a, the black path is the initial path, the red dashed line is the connection attempted to be established, and the green path in Fig. 5b is the path after removing the redundant nodes (a2, a3, a6), i.e., path a8-a7-a5-a4-a1.



(a) Attempting to connect (b) Successfully establishing connection

Figure 5: Schematic diagram of redundant point removal

After removing the redundant nodes, the path is shorter and has fewer turning points, but the path is zigzag, as shown in Fig. 5b, which is more zigzag and will lead to a serious jerky phenomenon of the robotic arm during operation. To solve this problem, the introduction of cubic uniform B-spline curves for path smoothing can effectively smooth the zigzag paths and improve the continuity and

maneuverability of the paths. k-order B-spline function expression is:

$$C(x) = \sum_{i=0}^{n} P_i N_{i,\,k}(x) \tag{14}$$

where P_i (i = 0, 1, 2, 3,..., n) denotes the control point, which can be moved to change the shape of the curve; $N_{i,\,k}(x)$ is the k-order B-spline basis function, which can be expressed by the deBoor-Cox recursive formula, as shown in Eq. (15):

$$N_{i,\,k}(x) = \begin{cases} \begin{cases} 1, & x_i \le x < x_{i+1},\ k = 1 \\ 0, & other \end{cases} \\ \dfrac{x - x_i}{x_{i+k-1} - x_i} N_{i,\,k-1}(x) + \dfrac{x_{i+k} - x}{x_{i+k} - x_{i+1}} N_{i+1,\,k-1}(x), \quad k \ge 2 \end{cases} \tag{15}$$

where k denotes the number of curves and the subscript i denotes the ordinal number.

The basis function of the cubic B-spline curve to optimize the smoothness of the path is given by:

$$\begin{cases} N_{0,\,3}(x) = \dfrac{1}{6}\left(-x^3 + 3x^2 - 3x + 1\right) \\ N_{1,\,3}(x) = \dfrac{1}{6}\left(3x^3 - 6x^2 + 4\right) \\ N_{2,\,3}(x) = \dfrac{1}{6}\left(-3x^3 + 3x^2 + 3x + 1\right) \\ N_{3,\,3}(x) = \dfrac{1}{6}x^3 \end{cases} \tag{16}$$

Then the 3 times B-spline segments are:

$$C_{0,\,3}(x) = C_0 N_{0,\,3}(x) + C_1 N_{1,\,3}(x) + C_2 N_{2,\,3}(x) + C_3 N_{3,\,3}(x) \tag{17}$$

However, the smoothing process is not without risk. The original path has been configured spatially mapped in joint space to ensure that the path does not collide with obstacles. However, the smoothed path may collide with the robotic arm due to the curve adjustment. Therefore, after applying B-splines to smooth the path, collision detection must be re-performed to ensure the safety and feasibility of the final path. Finally, a curved line segment was successfully smoothed on the MATLAB platform, as shown in Fig. 6.

In summary, an improved RRT∗ algorithm is proposed by fusing the adaptive biased probabilistic sampling strategy, the gravitationally adjustable step-size strategy, and the path optimization strategy, referred to as the AGP-RRT∗ algorithm.

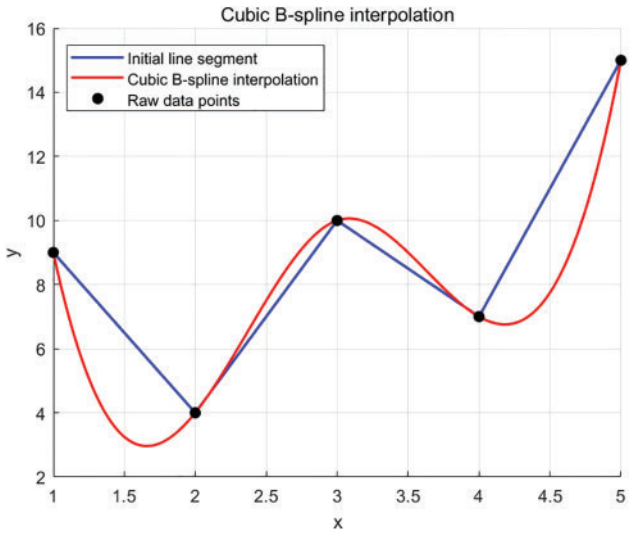


Figure 6: Curve smoothing processing map

5 Simulation Experiment and Result Analysis

To confirm the efficacy, dependability, and superiority of the proposed AGP-RRT∗ for multi-axis robotic arm path planning, the algorithm's performance is tested within a three-dimensional setting. The six-axis robotic arm was selected as the research object and simulation experiments were carried out based on its D-H parameters (see Table 1). Table 1 shows one of the customized robot parameters, where θ_i denotes the joint angle, d_i denotes the linkage deviation of the joint axis, a_{i-1} denotes the linkage length of the joint, and α_{i-1} denotes the linkage torsion angle. It is worth noting that different brands and models of robotic arms are provided with corresponding D-H parameters. During the experiments, the AGP-RRT∗ algorithm is compared and analyzed with the RRT∗ and A-RRT∗ algorithms. Additionally, the efficacy of the proposed algorithm is confirmed through testing in a sophisticated environment. The above simulation experiments are realized on the MATLAB 2023b platform.

Table 1: Six-axis robotic arm D-H parameters

Joint	α_{i-1}/°	a_{i-1}/mm	d_i/mm	θ_i/°
1	90	0	12.6	θ_1
2	0	0	13.9	θ_2
3	90	46.2	0	θ_3
4	90	38.7	0	θ_4
5	−90	0	9.8	θ_5
6	0	0	11.1	θ_6

5.1 Three-Dimensional Simulation

The task of 3D simulation is to ensure that the algorithm can find a path from the starting point to the target point, and at the same time realize the avoidance of obstacles in the path planning process. The experiment sets up a simulation environment of 100 × 100 × 85 size, and sets 9 spherical obstacles of different sizes in blue color. The start point is set as [5, 5, 5], the end point is set as [80, 70, 80], the step size of the algorithm is 5, and the target bias probability is 0.7. Considering that the algorithm has a certain stochastic nature, the experiments are conducted 50 times to guarantee the dependability of the outcomes. The results and key data of the experiments are shown in Fig. 7 and Table 2, respectively.

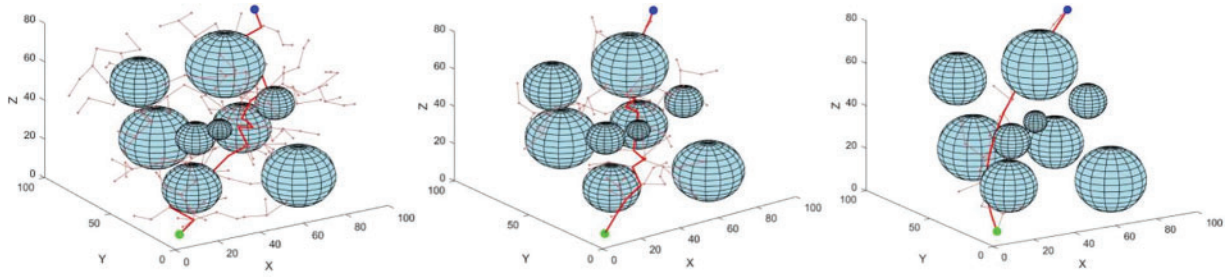


Figure 7: Simulation of 3D path planning for each algorithm

Table 2: Comparison of data for each algorithm in a three-dimensional environment

Algorithm	Average run time/s	Average path length/cm	Average number of sampling points
RRT∗	29.226	187.990	167
A-RRT∗	12.183	156.773	94
AGP-RRT∗	3.354	131.179	32

As shown in Fig. 7, the red lines indicate the final path of the algorithm, and the pink lines indicate the branches of the expanded tree. According to the data analysis in Table 2, it can be seen that the AGP-RRT∗ algorithm achieves 88.524% reduction in search time and 30.22% reduction in average path length compared to the RRT∗ algorithm; compared to the A-RRT∗ algorithm, the AGP-RRT∗ algorithm reduces the path cost by 16.326% and reduces the search time by 72.47%. The experimental results illustrate that the AGP-RRT∗ algorithm provides superior path planning performance in 3D environments.

5.2 Six-Axis Robotic Arm Obstacle Avoidance Path Planning Simulation

The six-axis robotic arm simulation experiment is performed on MATLAB software as shown in Fig. 8. There are four obstacles in the experimental environment, including two blue rectangular obstacles and two blue spherical obstacles. The start point is set as (−65, −25, 70) and the target point is set as (50, −30, −35). The robot arm is required to chart a course that bypasses any obstacles, extending from the origin to the endpoint.

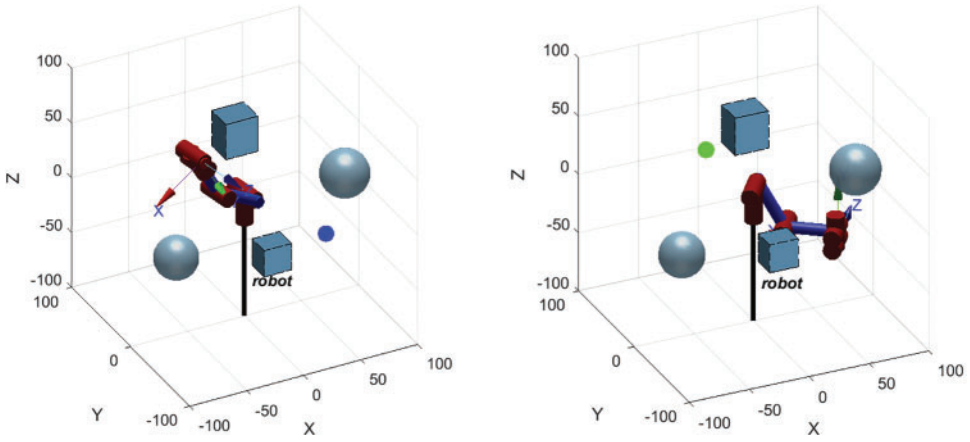


Figure 8: Simulation environment of robotic arm

Kinematic constraints are incorporated into the experiments to ensure that the movement of the robotic arm conforms to its physical characteristics and operational capabilities. The kinematic constraints are:

Positional constraints:

$$q_{j\min} \leq q_j(t) \leq q_{j\max} \tag{18}$$

Speed constraints:

$$|v_j| \leq v_{j\max} \tag{19}$$

Acceleration constraints:

$$|a_j| \leq a_{j\max} \tag{20}$$

where j represents the jth joint, $q_j(t)$ is the configuration of joint j at time t, $q_{j\min}$ is the minimum positional limit of joint j, $q_{j\max}$ is the maximum positional limit of joint j, $v_{j\max}$ is the maximum positional limit of joint j, and $a_{j\max}$ is the maximum acceleration limit of joint j.

The experiment is configured with a maximum of 10,000 iterations. The running results are shown in Figs. 9 and 10, where the red line represents the final route, the blue line denotes the initial, unrefined path, and the orange illustrates the algorithm's search progression. In Fig. 9a, the RRT∗ algorithm, due to its lack of goal orientation, results in a large sampling space and long and winding generated paths. In contrast, Fig. 9b illustrates the A-RRT∗ algorithm, which enhances target orientation by introducing adaptive biased probability sampling strategies and adjusting target deviation thresholds based on obstacle information. This approach generates shorter, less winding paths, but its sampling efficiency and path smoothness require improvement. The AGP-RRT∗ algorithm shown in Fig. 9c further improves the sampling speed by gravitationally dynamically adjusting the step size and generating smooth paths through path optimization processing. Further, it is observed that the joint position trajectory graph of the AGP-RRT∗ algorithm is smoother than the other two algorithms for robotic arm joint operation as shown in Fig. 10. The results show that the AGP-RRT∗ algorithm has the fastest search process and generates the best path performance.

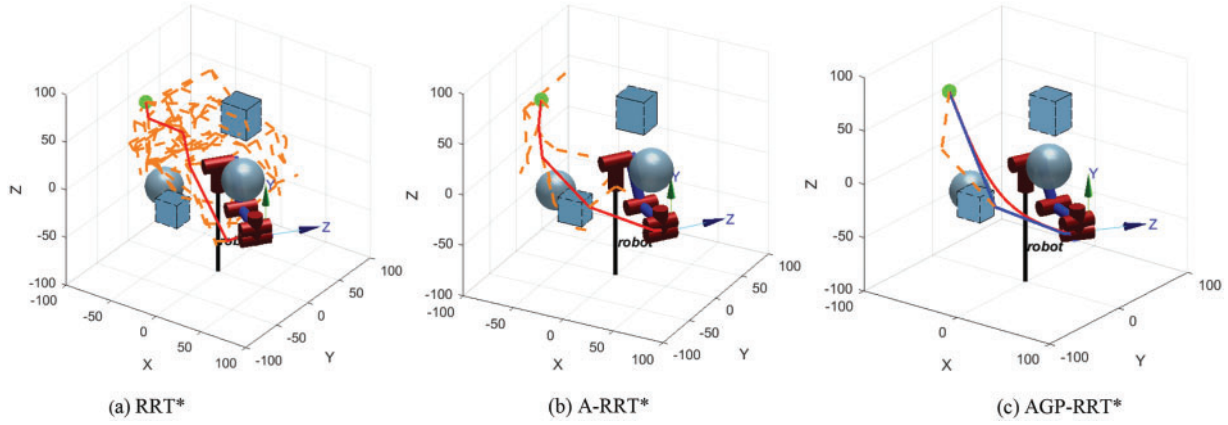


Figure 9: Simulation results of robotic arm path

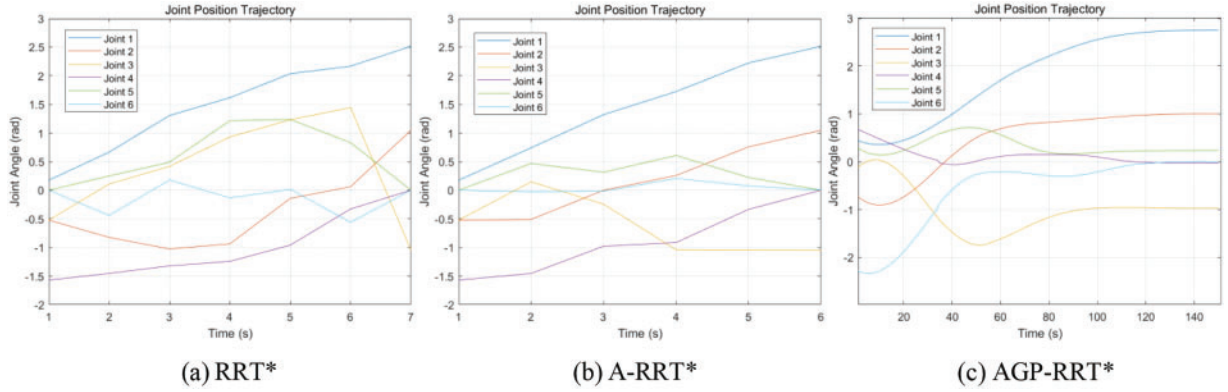


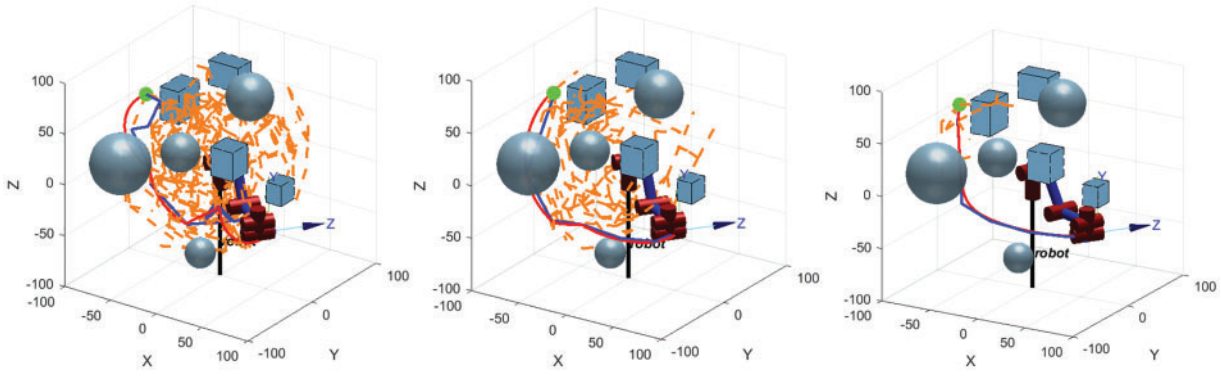
Figure 10: Joint position trajectory

According to the data in Table 3, the AGP-RRT∗ algorithm shows significant advantages in several indicators. Compared with the RRT∗ algorithm, the AGP-RRT∗ algorithm significantly reduces the average running time by 87.34%, and also reduces the average path cost by 40.39%; compared with the A-RRT∗ algorithm, the AGP-RRT∗ algorithm further optimizes the algorithmic performance by means of the gravitationally adjustable step-size strategy and the path optimization strategy. Specifically, the AGP-RRT∗ algorithm reduces 69.90% on the average running time and 16.86% on the average path cost. These results show that the AGP-RRT∗ algorithm not only improves the convergence speed, but also shortens the length of the generated path.

Table 3: Experimental data table for robotic arm simulation

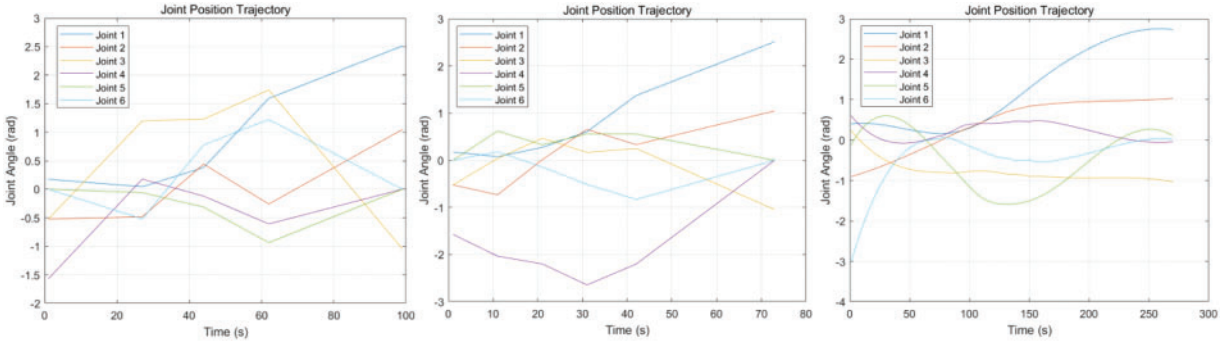
Algorithm	Average run time/s	Average path length/cm	Average number of sampling points
RRT∗	48.619	110.901	105
A-RRT∗	20.447	79.505	33
AGP-RRT∗	6.154	66.099	6

To ensure the adaptability of the proposed AGP-RRT∗ algorithm within various settings, it has been implemented in two distinct, more intricate scenarios, labeled Environment A and Environment B, each containing a total of eight obstacles. Due to the increase of obstacles, the paths generated by the RRT∗ and A-RRT∗ algorithms are more tortuous, especially the RRT∗ algorithm has more redundant nodes, which are difficult to run directly by the robotic arm, leading to the failure of the algorithm planning, and in order to carry out the comparison between the three, the paths are simplified by its two algorithms in order to ensure the smoothness of the paths. The simulation results are shown in Figs. 11–14.



(a) RRT* (b) A-RRT* (c) AGP-RRT*

Figure 11: Complex Environment A robotic arm path simulation results



(a) RRT* (b) A-RRT* (c) AGP-RRT*

Figure 12: Complex Environment A-joint position trajectory map

From Figs. 11 and 13, it can be seen that the traditional RRT∗ algorithm is difficult to adapt effectively, and the A-RRT∗ algorithm performs at a significant discount relative to simple environments. In contrast, the AGP-RRT∗ algorithm still shows good performance in complex environments. As shown in Figs. 12 and 14, the AGP-RRT∗ algorithm still outperforms the RRT∗ and A-RRT∗ algorithms for its joint position trajectories. Further analyzing the data in Tables 4 and 5, the AGP-RRT∗ algorithm reduces the average running time by 94.56% and 95.37%, and the average path cost by 55.28% and 47.82% compared to the RRT∗ algorithm; compared to the A-RRT∗ algorithm, the average running time is reduced by 87.88% and 89.24%, and the average path cost is reduced by 34.10% and 20.97%, respectively. Also, the AGP-RRT∗ algorithm generates the least number of nodes.

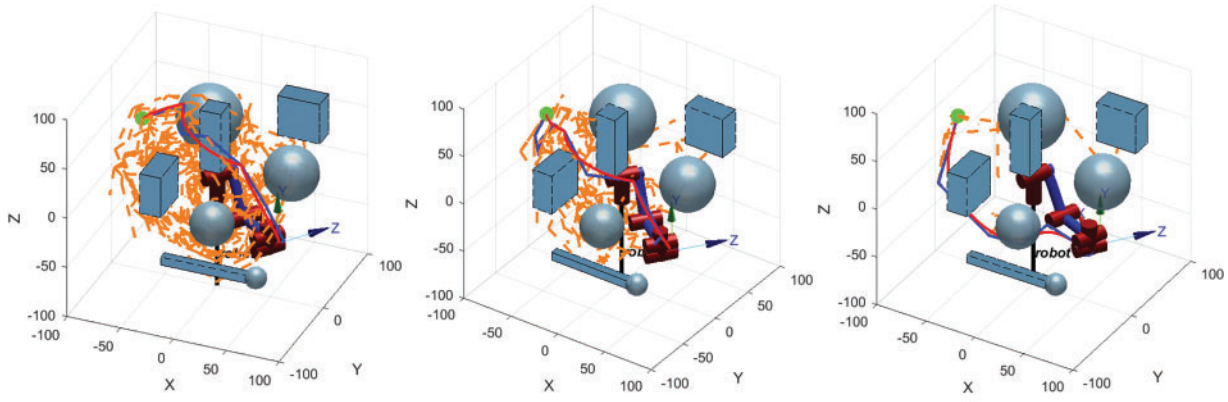


Figure 13: B.robotic arm path simulation results

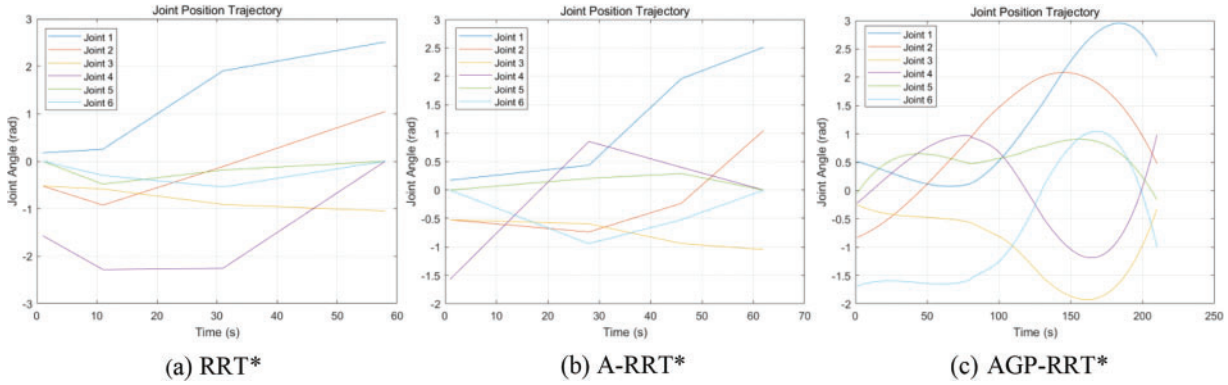


Figure 14: B.joint position trajectory map

Table 4: Complex Environment A robotic arm simulation experiment data sheet

Algorithm	Average run time/s	Average path length/cm	Average number of sampling points
RRT∗	162.085	243.695	336
A-RRT∗	72.735	165.366	155
AGP-RRT∗	8.816	108.972	18

Table 5: Complex Environment B robotic arm simulation experiment data sheet

Algorithm	Average run time/s	Average path length/cm	Average number of sampling points
RRT∗	203.087	238.947	388
A-RRT∗	87.329	157.784	195
AGP-RRT∗	9.396	124.689	20

The above results show that the AGP-RRT∗ algorithm exhibits significant advantages in several aspects. First, in terms of convergence speed, the significant reduction of the AGP-RRT∗ algorithm in the average running time (94.56% *vs*. 95.37% compared to the RRT∗ algorithm, and 87.88% *vs*. 89.24% compared to the A-RRT∗ algorithm) suggests that it is able to find the feasible paths faster. Second, in terms of adaptability, the AGP-RRT∗ algorithm performs particularly well in complex environments. Even if multiple obstacles are set up in the environment, the AGP-RRT∗ algorithm is still able to generate smoother paths, whereas traditional RRT∗ algorithms show obvious path zigzags and redundant nodes when faced with the same complex environments, making it difficult to be directly applied to robotic arm operation.

Finally, in terms of performance advantages, the AGP-RRT∗ algorithm shows a reduction in the average path cost (55.28% *vs*. 47.82% compared to the RRT∗ algorithm, and 34.10% *vs*. 20.97% compared to the A-RRT∗ algorithm), which means that the paths it generates are not only shorter, but also likely to lead to higher efficiency and economy in practical applications.

In summary, the AGP-RRT∗ algorithm demonstrates good performance advantages by significantly improving the convergence speed, enhancing the adaptability, and optimizing the path cost, especially the potential for application in complex environments.

In the simulation verification part, since the six-axis robotic arm is widely used in the industrial field, thus the six-axis robotic arm is selected as the experimental object to fully test the effectiveness and performance of the AGP-RRT∗ algorithm in path planning. Although the experiment is only a simulation test for the six-axis robotic arm, the AGP-RRT∗ algorithm has strong generality and adaptability, and can be effectively applied to other types of multi-axis robotic arms in theory. The design idea of the algorithm is compatible with the kinematic model and local constraints of the multi-axis robotic arm, which lays the foundation for its applicability under various motion laws. The adaptive biased probabilistic sampling strategy and the adjustable step size strategy enable the algorithm to flexibly cope with the demands of robotic arms with different numbers of axes.

Furthermore, the AGP-RRT∗ algorithm can handle various obstacle distributions and path planning complexities by combining environmental adaptation and path smoothing requirements on top of path search and node generation. Therefore, although the experiments only focus on a six-axis robotic arm, in theory, other types of multi-axis robotic arms with limited 3D space and complex joint constraints can also benefit from the algorithm.

In summary, the AGP-RRT∗ algorithm shows good applicability and potential application value for different types of multi-axis robotic arms in complex environments by virtue of its efficient path planning capability, good adaptability, and optimized path generation.

6 Summary

An AGP-RRT∗ algorithm is proposed, which aims to optimize the problems of large sampling area, long search time, tortuous paths, and too many redundant points in the traditional RRT∗ algorithm. The key improvement points of the algorithm include:

(1) Optimizing the sampling of the algorithm by dynamically adjusting the target deviation threshold m, which effectively avoids collisions and reduces the generation of redundant nodes, resulting in better performance of the generated paths.

(2) Dynamically adjusting the step size, combined with the gravitational guidance and obstacle distribution, to further improve the search efficiency of the algorithm.

(3) To improve path smoothness, redundant nodes are removed from the path and the path is optimized using third-order B-splines, enhancing path smoothness.

Through algorithmic 3D environment simulations and simulated experiments with six-axis robotic arms in varying levels of complexity, the AGP-RRT∗ algorithm has demonstrated significant reductions in path planning time and the generation of superior paths. Due to the generality and applicability of the algorithm, the algorithm can provide some reference value for the path planning of multi-axis robotic arms.

Acknowledgement: This paper grateful thanks to all reviewers for their selfless contributions to science, and to all editors for their outstanding work in improving the quality of the paper.

Funding Statement: The work is supported by Foundation of key Laboratory of AI and Information Processing of Education Department of Guangxi (No. 2022GXZDSY002) (Hechi University), Foundation of Guangxi Key Laboratory of Automobile Components and Vehicle Technology (Nos. 2022GKLACVTKF04, 2023GKLACVTZZ06).

Author Contributions: Study conception and design: Juanling Liang, Wenguang Luo; data collection: Juanling Liang; analysis and interpretation of results: Juanling Liang, Wenguang Luo, Yongxin Qin; draft manuscript preparation: Juanling Liang, Wenguang Luo. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The authors confirm that the data supporting the findings of this study are available within the paper.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] H. Xing, Y. Zhao, Y. Zhang, and Y. Chen, "3D trajectory planning of positioning error correction based on PSO-A* algorithm," *Comput. Mater. Contin.*, vol. 65, no. 3, pp. 2295–2308, 2020. doi: 10.32604/cmc.2020.011858.

[2] C. Lin, Y. He, X. Fang, and C. Wang, "An intelligent vehicle trajectory planning method for parking scenarios," (in Chinese), *J. Shanghai Jiao Tong Univ.*, vol. 57, no. 3, pp. 345–353, 2023.

[3] Y. Huang *et al.*, "A motion planning and tracking framework for autonomous vehicles based on artificial potential field elaborated resistance network approach," *IEEE Trans. Ind. Electron.*, vol. 67, no. 2, pp. 1376–1386, Feb. 2020. doi: 10.1109/TIE.2019.2898599.

[4] Z. Pan, C. Zhang, Y. Xia, H. Xiong, and X. Shao, "An improved artificial potential field method for path planning and formation control of the multi-UAV systems," *IEEE Trans. Circuits Syst. II, Exp. Briefs.*, vol. 69, no. 3, pp. 1129–1133, Mar. 2022. doi: 10.1109/TCSII.2021.3112787.

[5] M. Zhang, H. Ren, and Y. Zhou, "Research on global ship path planning method based on improved ant colony algorithm," *IEEE Open J. Intell. Transp. Syst.*, vol. 4, pp. 143–152, 2023. doi: 10.1109/OJITS.2023.3247377.

[6] Z. Lin, K. Wu, R. Shen, X. Yu, and S. Huang, "An efficient and accurate A-star algorithm for autonomous vehicle path planning," *IEEE Trans. Veh. Technol.*, vol. 73, no. 6, pp. 9003–9008, Jun. 2024. doi: 10.1109/TVT.2023.3348140.

[7] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016. doi: 10.1109/TITS.2015.2498841.

[8] Y. Zhang, Z. Ju, H. Zhang, and Z. Qi, "3-D path planning using improved RRT* algorithm for robot-assisted flexible needle insertion in multilayer tissues," *IEEE Can. J. Electr. Comput. Eng.*, vol. 45, no. 1, pp. 50–62, 2022. doi: 10.1109/ICJECE.2021.3120324.

[9] C. -B. Moon and W. Chung, "Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1080–1090, Feb. 2015. doi: 10.1109/TIE.2014.2345351.

[10] J. Dai, Y. Zhang, and H. Deng, "Novel potential guided bidirectional RRT* with direct connection strategy for path planning of redundant robot manipulators in joint space," *IEEE Trans. Ind. Electron.*, vol. 71, no. 3, pp. 2737–2747, Mar. 2024. doi: 10.1109/TIE.2023.3269462.

[11] J. Qi, H. Yang, and H. Sun, "MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment," *IEEE Trans. Ind. Electron.*, vol. 68, no. 8, pp. 7244–7251, Aug. 2021. doi: 10.1109/TIE.2020.2998740.

[12] M. F. Ozkan, L. R. G. Carrillo, and S. A. King, "Rescue boat path planning in flooded Urban environments," in *Proc. IEEE Int. Symp. Meas. Control Robot. (ISMCR)*, Houston, TX, USA, 2019, pp. B2-2-1–B2-2-9.

[13] H. -T. L. Chiang and L. Tapia, "COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2024–2031, Jul. 2018. doi: 10.1109/LRA.2018.2801881.

[14] B. Li and B. Chen, "An adaptive rapidly-exploring random tree," *IEEE/CAA J. Autom. Sinica.*, vol. 9, no. 2, pp. 283–294, Feb. 2022. doi: 10.1109/JAS.2021.1004252.

[15] J. Wang, M. Q. H. Meng, and O. Khatib, "EB-RRT: Optimal motion planning for mobile robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 2063–2073, Oct. 2020. doi: 10.1109/TASE.2020.2987397.

[16] L. Chen, Y. Shan, W. Tian, B. Li, and D. Cao, "A fast and efficient double-tree RRT*-like sampling-based planner applying on mobile robotic systems," *IEEE/ASME Trans. Mechatron.*, vol. 23, no. 6, pp. 2568–2578, Dec. 2018. doi: 10.1109/TMECH.2018.2821767.

[17] X. Wang, Y. Hua, J. Gao, Z. Lin, and R. Yu, "Digital twin implementation of autonomous planning arc welding robot system," *Complex Syst. Model. Simul.*, vol. 3, no. 3, pp. 236–251, Sep. 2023. doi: 10.23919/CSMS.2023.0013.

[18] B. An, J. Kim, and F. C. Park, "An adaptive stepsize RRT planning algorithm for open-chain robots," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 312–319, Jan. 2018. doi: 10.1109/LRA.2017.2745542.

[19] H. Shen, W. -F. Xie, J. Tang, and T. Zhou, "Adaptive manipulability-based path planning strategy for industrial robot manipulators," *IEEE/ASME Trans. Mechatron.*, vol. 28, no. 3, pp. 1742–1753, Jun. 2023. doi: 10.1109/TMECH.2022.3231467.

[20] W. Zhang, L. Shan, L. Chang, and Y. Dai, "SVF-RRT*: A stream-based VF-RRT* for USVs path planning considering ocean currents," *IEEE Robot. Autom. Lett.*, vol. 8, no. 4, pp. 2413–2420, Apr. 2023. doi: 10.1109/LRA.2023.3245409.

[21] J. Wang, T. Li, B. Li, and M. Q. -H. Meng, "GMR-RRT*: Sampling-based path planning using Gaussian mixture regression," *IEEE Trans. Intell. Veh.*, vol. 7, no. 3, pp. 690–700, Sep. 2022. doi: 10.1109/TIV.2022.3150748.

[22] Y. Zhang, H. Wang, M. Yin, J. Wang, and C. Hua, "Bi-AM-RRT*: A fast and efficient sampling-based motion planning algorithm in dynamic environments," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 1282–1293, Jan. 2024. doi: 10.1109/TIV.2023.3307283.

[23] H. -T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "RL-RRT: Kinodynamic motion planning via learning reachability estimators from RL policies," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4298–4305, Oct. 2019. doi: 10.1109/LRA.2019.2931199.

[24] C. Yuan, W. Zhang, G. Liu, X. Pan, and X. Liu, "A heuristic rapidly-exploring random trees method for manipulator motion planning," *IEEE Access*, vol. 8, pp. 900–910, 2020. doi: 10.1109/ACCESS.2019.2958876.

[25] J. Wang, W. Chi, C. Li, C. Wang, and M. Q. -H. Meng, "Neural RRT*: Learning-based optimal path planning," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 1748–1758, Oct. 2020. doi: 10.1109/TASE.2020.2976560.

[26] C. Jiang *et al.*, "R2-RRT*: Reliability-based robust mission planning of off-road autonomous ground vehicle under uncertain terrain environment," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 2, pp. 1030–1046, Apr. 2022. doi: 10.1109/TASE.2021.3050762.

[27] L. Wu, R. Crawford, and J. Roberts, "An analytic approach to converting POE parameters into D-H parameters for serial-link robots," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2174–2179, Oct. 2017. doi: 10.1109/LRA.2017.2723470.