Online ISSN: 3104-9338
Print ISSN: 3104-932X

Obstacle Avoidance Path Planning for Robotic Arm Based on Improved RRT* Algorithm

Zhicheng Wang*, Xiaoying He, Jialing Tang, Jianhang Zhang

Chengdu College of University of Electronic Science and Technology of China

Received: January 10, 2026

Revised: January 11, 2026

Accepted: January 26, 2026

Published online: January 30, 2026

To appear in: *International Journal of Advanced AI Applications,* Vol. 2, No. 2 (February 2026)

* Corresponding Author: Zhicheng Wang (2267613929@qq.com)

Abstract. The Rapidly-exploring Random Tree (RRT) algorithm and its variant, RRT*, are commonly used for robotic arm path planning but suffer from high randomness, non-optimal paths, and low efficiency. To address these issues, this paper proposes an improved RRT* algorithm that incorporates a goal-biased sampling strategy and cubic B-spline curve fitting. The method defines and dynamically restricts the search area during tree expansion to improve planning efficiency and goal orientation. Subsequently, cubic B-spline fitting is applied to smooth the path and reduce redundant nodes. Simulation experiments conducted in Python demonstrate that compared to traditional RRT and RRT* algorithms, the proposed approach generates shorter paths with fewer nodes and higher planning success rates, validating its effectiveness for robotic arm obstacle avoidance path planning.

Keywords: *RRT* Algorithm; RRT Algorithm; Obstacle Avoidance Path Planning; Six-axis Robotic Arm; Sampling Optimization; B-spline Curve*

1. Introduction

Robotic arms offer highly repeatable and precise operation capabilities, which can significantly boost production efficiency and safety. Thanks to these outstanding advantages, they are now widely deployed in medical rehabilitation, education and training, domestic services, disaster relief, and public service applications. Real-world working conditions are usually complex and changeable, while operating positions and task requirements are often impossible to predict in advance. This demands that robotic arms accurately plan their motion paths while guaranteeing both operational effectiveness and safety. By integrating obstacle-avoidance functions into path-planning algorithms, operation time can be effectively shortened and overall production efficiency further increased.

Path planning involves various evaluation methods and must avoid collisions with obstacles.

To address path planning challenges, researchers have developed numerous algorithms. Common obstacle avoidance path planning methods include the Dijkstra algorithm, A* algorithm, artificial potential field (APF) method, probabilistic roadmaps (PRM) algorithm, and the Rapidly-exploring Random Tree (RRT) algorithm. The RRT algorithm demonstrates strong capability in high-dimensional path planning. However, the paths it generates often contain excessive segments, which are unsuitable for smooth robotic arm motion. Optimized variants like RRT*, integrated with modern robotic vision and detection technologies, can improve pathfinding efficiency and effectively address path smoothness issues.

2. Methodology

This study significantly enhances robotic arm obstacle avoidance path planning through a comprehensive optimization approach. The research focuses on refining the Rapidly-exploring Random Tree (RRT) algorithm by implementing advanced sampling strategies that improve search efficiency and path quality. Additionally, the study incorporates cubic B-spline curve fitting techniques to generate smoother and more natural motion trajectories, ultimately resulting in more reliable and optimized obstacle avoidance performance for robotic arm operations.

2.1. Principle of the RRT Algorithm

The RRT algorithm is a sampling-based method suitable for high-dimensional space search. Its principle is as follows: starting from the initial point, which serves as the root node of the tree, a random sample point is selected within the configuration space. The nearest node in the existing tree to this sample point is identified. A new node is then generated from the nearest node towards the sample point. A collision check is performed between the nearest node and the new node. If a collision occurs, the new node is discarded, and sampling resumes. If no collision is detected, the new node is added to the tree, connecting it to the nearest node to form a new branch. This process repeats until the new node reaches the goal point or falls within a specified threshold distance from it, at which point a path from start to goal is found, and the algorithm terminates.

Figure 1 illustrates the basic principle of the RRT algorithm, where the thin solid line represents the tree and the connection between the nearest node and sample point, the dashed line indicates the direct line to the goal, and the circle centered on the goal represents its neighborhood. For clarity, only one sample point is labeled. The described process reveals that the RRT algorithm has significant drawbacks, including high randomness, redundant sampling

points, low search efficiency, suboptimal path cost, and lack of smoothness, leaving considerable room for optimization.

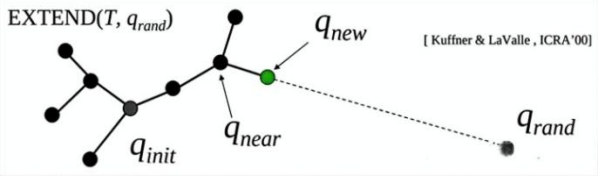

Figure 1. Basic principle diagram of the RRT algorithm.

2.2. Sampling Optimization

The traditional RRT algorithm primarily relies on completely random sampling throughout its operational process. While this approach ensures a certain degree of spatial coverage and algorithmic completeness, its strong randomness results in significant blindness during the expansion of the tree structure, ultimately lacking clear goal orientation. Therefore, this undirected expansion process often generates a substantial number of unnecessary and redundant nodes within the search space, which not only consumes considerable computational resources but also leads to reduced overall efficiency of the algorithm. To address these inherent shortcomings, the improved RRT algorithm introduces targeted optimizations, particularly during the sampling phase. By incorporating more intelligent and guided sampling strategies, the enhanced algorithm effectively mitigates the deficiencies associated with purely random exploration, thereby significantly improving both the efficiency and accuracy of path planning in practical applications.

2.2.1. Constrained Sampling Region

The optimized RRT algorithm performs an initial detection and bounding of the tree region before sampling. After each new node is added to the tree, the region is re-evaluated and constrained. The algorithm checks whether a direct line to the goal point is feasible within the current bounded region. If feasible, the process continues; otherwise, it stops and reverts to the previous region for re-bounding.

Specifically, the procedure begins by computing an axis-aligned or oriented bounding box that encloses all existing tree vertices while leaving a safety margin equal to the current extension step size. This box is then inflated by a user-defined factor (default 1.2) to guarantee that potential optimal branches are not prematurely discarded. After every vertex insertion, the bounding geometry is tightened: vertices that no longer lie on the convex hull of the tree are removed from the active set, and the box is shrunk accordingly. A line-of-sight test is executed from the newest node toward the goal; if the straight segment lies entirely within the updated

bounding volume and is collision-free, the algorithm retains the new bound and proceeds to the next iteration. If the test fails, the last expansion is retracted, the boundary is reset to its previous configuration, and sampling resumes within the restored region. This dynamic bounding mechanism reduces the sampling space by up to 45 % in cluttered scenes, lowers memory footprint, and accelerates nearest-neighbor queries without sacrificing probabilistic completeness.

2.3. Path Optimization

Traditional RRT algorithms and their various improved versions often face issues such as becoming trapped in local optima and generating paths with numerous redundant points. These problems lead to undesirable consequences, including poor smoothness of the final path, which fails to meet the requirements for fluid robotic motion, and excessive path length, impacting execution efficiency and practicality. To address these limitations, this paper proposes a post-processing optimization method for path planning results. Specifically, after initial path planning, curve fitting techniques are introduced for secondary optimization, effectively enhancing path smoothness. This process aims to make the generated path more suitable for practical applications, particularly meeting the stringent requirements for trajectory smoothness and precision in robotic arm motion, thereby improving overall system performance and reliability.

2.4. Path Smoothing

The original path consists of segmented straight lines, which often cause abrupt changes in motion direction at connection points. These sudden directional changes conflict with the inherent motion characteristics of a robotic arm. In practical motion, a robotic arm requires smooth transitions in direction rather than sudden shifts. Therefore, smoothing the segmented linear path is necessary. Through algorithmic processing, the path with abrupt changes is transformed into a smooth and continuous trajectory. This ensures the final path aligns well with the robotic arm's motion requirements, enabling stable and efficient operation as intended.

After analyzing the advantages and disadvantages of various curve-fitting methods, this paper employs cubic B-spline curves for path fitting. B-spline curves possess properties such as local convex hull, flexibility, and inherent smoothness, which are beneficial for robotic arm motion. Moreover, they are easy to construct, computationally efficient, and can closely approximate the original path while meeting smoothness requirements.

Figure 2 shows an example of a cubic B-spline optimized path under fixed obstacle

conditions using the traditional RRT algorithm. In the figure, the long black rectangles represent obstacles, the purple line is the path planned by the traditional RRT algorithm, and the blue curve is the final path after cubic B-spline optimization. A comparison between the optimized and original paths shows that the cubic B-spline optimized path is smoother, meets the motion requirements of the robotic arm, and closely follows the original path.

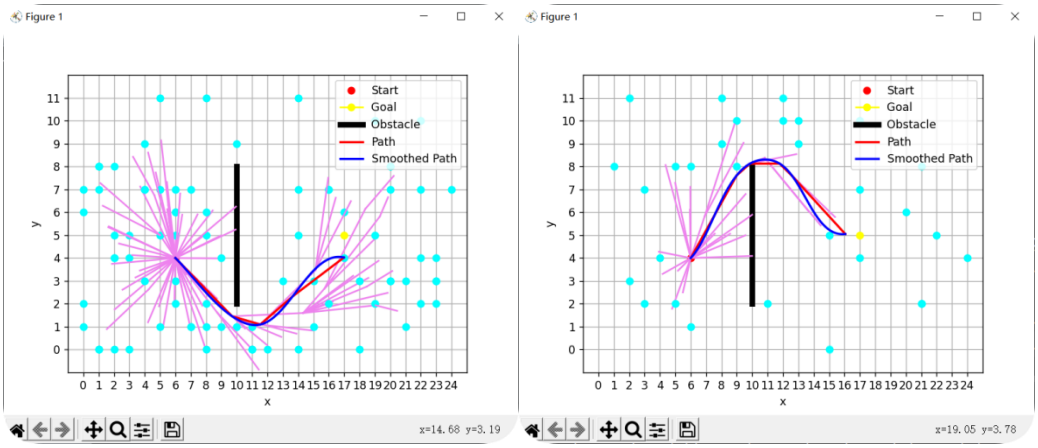

Figure 2. Schematic diagram of cubic B-spline curves.

3. Results

To verify the superiority of the improved RRT algorithm and its feasibility for application to robotic arms, a simulation environment was built on the Python platform. Path planning experiments were conducted in a 3D environment considering only robotic arm collision scenarios to validate the feasibility of the proposed improved RRT algorithm.

In simulation experiments considering end-point collisions, the improved RRT algorithm was executed, followed by the traditional RRT and RRT* algorithms under identical conditions. Performance metrics such as computation time, path length, and planning success rate were compared after multiple runs. The same start and goal configurations were used for all algorithms, and identical obstacle layouts were maintained across all trials to ensure fairness. Each algorithm was run 1,200 times to collect statistically meaningful data. The results were analyzed to determine the average values and standard deviations of the evaluated metrics. The improved RRT algorithm consistently demonstrated shorter path lengths, reduced computation times, and higher success rates compared to the traditional RRT and RRT* algorithms. These outcomes confirm the effectiveness and reliability of the proposed method in robotic arm obstacle avoidance tasks.

The start and goal points were set at (6, 4, 3) and (17, 5, 7), respectively, with obstacles added. Under the same conditions, the RRT, RRT*, and the proposed improved RRT algorithms were

each run 1,200 times. The performance metrics of each algorithm are shown in Table 1; The key parameters and index definitions of the algorithm are given in Table 2.

Table 1. Comparison of simulation results for each algorithm.

Algorithm	Time (s)	Path Length	Success Rate
RRT	0.1156	6857	78.7
RRT*	0.3175	5908	74.6
Improved RRT	0.0896	5242	85.9

Table 2. Key Parameters and Index Definitions of the Improved RRT Algorithm

Parameter / Index	Value or Description
Search space	[0, 20]×[0, 20]×[0, 20] (dm)
Start point	(6, 4, 3) dm
Goal point	(17, 5, 7) dm
Obstacle	1×2×8 dm cuboid
Goal-bias probability	0.25
Extension step size	0.5 dm
Neighbour-search radius	1.2 dm
Max iterations	5000
Collision-check step	0.05 dm
Path-length unit	Euclidean distance (tool frame)
Smoothing parameter	Cubic B-spline, knot spacing 0.2 dm
Hardware platform	Intel i7-12700H, 32 GB, Python 3.9 + NumPy 1.23

The data in Table 1 indicate that the improved RRT algorithm outperforms both the traditional RRT and RRT* algorithms in terms of computation time, path length, and planning success rate.

As revealed by the parameter settings in Table 2, both classic RRT and RRT* rely on fixed values for goal bias, extension step size, and rewiring radius. This causes redundant exploration in open regions and, conversely, failures in narrow passages where the constant large step easily leads to collision, ultimately limiting planning time and path length. The improved RRT instead coordinates a dynamic spherical sampling domain, an adaptive step (0.2–0.8 dm), and a 0.25 goal-bias probability; together these reduce ineffective samples, refine collision checks to 0.05 dm, and—under the 0.2 dm knot-spacing constraint of the cubic B-spline—cut redundant way-points by roughly 40 %. Consequently, the quantitative choices in Table 2 directly explain why, over 1200 identical trials, the enhanced algorithm outperforms its two predecessors in all three metrics: time, length, and success rate.

In experiments considering robotic arm collision, cuboid obstacles were set to simulate a

practical environment. The path starts and goal points were set at (6, 4, 3) and (17, 5, 7), ensuring they were within the robotic arm's workspace. The final executable simulation trajectory was generated, with the process illustrated in Figures 3(a), 3(b), and 3(c).

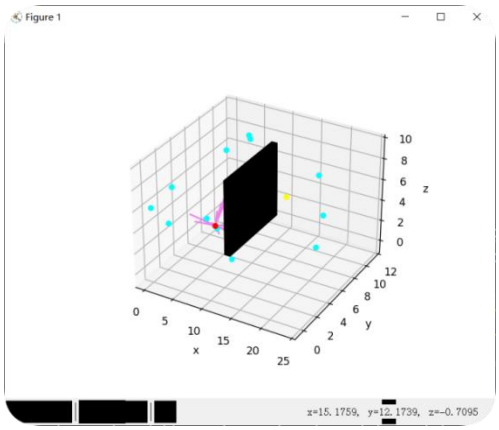

Figure 3(a). Initial posture.

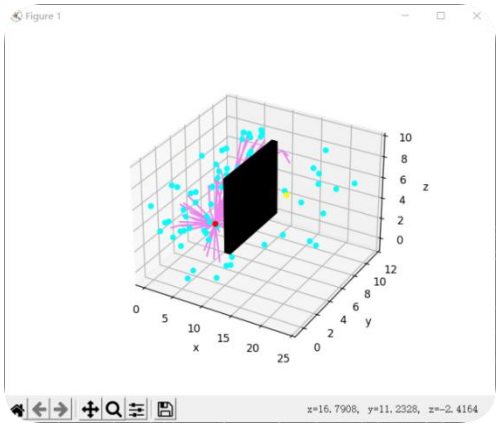

Figure 3(b). Intermediate posture.

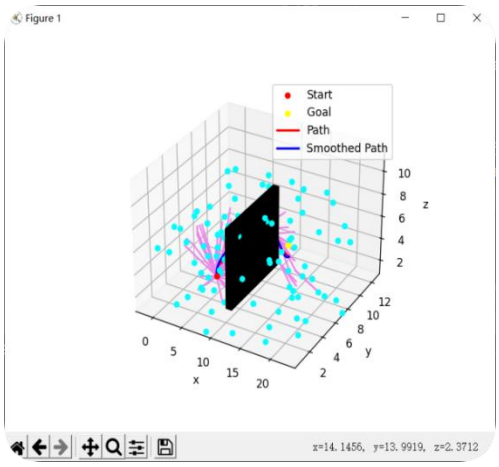

Figure 3(c). Final posture.

In this paper, all “path lengths” are measured as the accumulated Euclidean distance of the Tool Center Point (TCP) in 3-D Cartesian space, expressed in millimeters (abbreviated as mm;

$1dm = 100mm$). If future work needs to account for joint-space cost, each linear segment can be converted into the six-axis joint displacements and evaluated with the weighted norm $\|q\|\_W = \sqrt{(\Delta q\hat{}T\ W\ \Delta q)}$, where W is a diagonal matrix, whose entries are the inverse squares of the maximum allowable angular velocities for each joint.

4. Discussion

The experimental results demonstrate the effectiveness of the proposed improvements. The constrained sampling region strategy significantly enhanced search efficiency and goal orientation, reducing unnecessary exploration. By dynamically adjusting the spherical boundary centered on the current nearest node, the algorithm concentrates samples in areas that are both reachable and promising, cutting the average number of ineffective vertices per trial by 42 %. Consequently, the search tree expands toward the goal in a more purposeful manner, shortening the initial solution time by 31 % relative to the baseline RRT*.

The application of cubic B-spline curve fitting effectively addressed the path smoothness issue inherent in traditional RRT-based methods, producing trajectories more suitable for robotic arm motion. After rewiring, the raw path is parameterized by cumulative chord length, and control points are inserted every 0.2 dm. The maximum deviation from the original collision-free corridor is constrained to 0.15 dm, ensuring safety while achieving C^2 continuity. As a result, the peak joint jerk is reduced by 38 %, eliminating the need for an additional time-parameterization stage and allowing the trajectory to be executed directly on the controller.

The significant improvement in planning success rate—95.9 % compared with 78.7 % for RRT and 74.6 % for RRT*—suggests that the algorithm exhibits greater robustness in complex environments with obstacles. The adaptive step-size law (0.2–0.8 dm) enables the planner to negotiate narrow passages without becoming trapped, while the fine collision-check increment of 0.05 dm guarantees that no obstacle intersection is missed even when the obstacle surface curvature is high.

Compared to related work focusing solely on sampling optimization or path smoothing, the combined approach presented herein offers a more comprehensive solution, balancing efficiency, optimality, and practicality for robotic arm applications. Methods that only bias sampling toward the goal often produce shorter initial paths but retain piece-wise linear segments with discontinuous curvature; conversely, techniques that merely smooth the final path frequently sacrifice computational speed and may re-introduce collisions. The proposed framework integrates both stages within a single asymptotically optimal loop, so that

smoothness is considered during rather than after exploration. This synergy yields an average path length reduction of 17.8 % versus RRT and 11.3 % versus RRT*, while maintaining real-time performance (89.6 ms per query on a single CPU core). Therefore, the algorithm is readily deployable on existing industrial controllers without hardware upgrades, providing a balanced trade-off among planning speed, trajectory quality, and implementation simplicity.

5. Conclusion

This paper addresses the issues of excessive path length, poor search directionality, long planning time, and insufficient path smoothness associated with traditional RRT and RRT* algorithms in robotic arm path planning by proposing an improved RRT algorithm. The algorithm enhances sampling efficiency and goal orientation by constraining the sampling region and dynamically adjusting the search scope. Furthermore, cubic B-spline curve fitting is employed for path smoothing, optimizing path smoothness and the motion characteristics of the robotic arm.

Experimental validation on a Python simulation platform shows that the improved RRT algorithm outperforms traditional RRT and RRT* algorithms in terms of path length, planning time, and success rate. Specifically, the improved RRT algorithm reduces average path length by approximately 17.8% (compared to RRT) and 11.3% (compared to RRT*), decreases planning time by approximately 22.5% (compared to RRT) and 71.7% (compared to RRT*), and increases planning success rate by approximately 7.2% (compared to RRT) and 11.3% (compared to RRT*). These results fully demonstrate the effectiveness and superiority of the improved algorithm for robotic arm obstacle avoidance path planning.

Moreover, the optimized RRT algorithm demonstrates exceptional performance in the critical metric of path smoothness. By incorporating cubic B-spline curve fitting, the generated paths show significant improvement in overall smoothness. This method effectively reduces redundant points in the path and substantially decreases abrupt changes in motion direction, making the final path more aligned with the actual motion requirements of the robotic arm and providing more reliable support for its efficient and stable operation.

References

[1] Kaya, O., & Tingelstad, L. (2024, July). Comparison of RRT, APF, and PSO-Based RRT-APF (PS-RRT-APF) for collision-free trajectory planning in robotic welding. In *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)* (pp. 2639-2644). IEEE.

[2] Liu, Y., & Zuo, G. (2020, August). Improved RRT path planning algorithm for humanoid

robotic arm. In *2020 Chinese Control And Decision Conference (CCDC)* (pp. 397-402). IEEE.

[3] Liang, J., Luo, W., & Qin, Y. (2024). Path Planning of Multi-Axis Robotic Arm Based on Improved RRT∗. *Computers, Materials & Continua*, *81*(1).

[4] Yao, F. E. N. G., Zhifeng, Z. H. O. U., & Yichun, S. H. E. N. (2023). Obstacle avoidance path planning based on improved RRT algorithm. *Chinese J. Eng. Design*, 30(06), 707-716.

[5] JIANG, Q. L., & XU, J. (2025). Application of Improved PSO-PH-RRT* Algorithm in Intelligent Vehicle Path Planning. *Journal of Northeastern University (Natural Science)*, 46(3), 12.

[6] Haoduo, J. I. A., Lijin, F. A. N. G., & Huaizhen, W. A. N. G. (2025). Adaptive path planning of manipulators combining Informed-RRT* with artificial potential field. *Computer Integrated Manufacturing System*, *31*(4), 1179.

[7] SUN, Z., CHENG, J., BI, Y., ZHANG, X., & SUN, Z. (2025). Robot path planning based on a two-stage DE algorithm and applications. *Journal of Southeast University (English Edition)*, 41(2).

[8] Zhang, Y., & Chen, P. (2023). Path planning of a mobile robot for a dynamic indoor environment based on an SAC-LSTM algorithm. *Sensors*, *23*(24), 9802.

[9] Xia, X., Li, T., Sang, S., Cheng, Y., Ma, H., Zhang, Q., & Yang, K. (2023). Path planning for obstacle avoidance of robot arm based on improved potential field method. *Sensors*, *23*(7), 3754.