WebLog Journal of Robotics and Applications

Research Article
Published: 12 Jun, 2026

An Obstacle Avoidance Trajectory Planning Algorithm for Robotic Arm Based on Two-Stage Grid Method

Guanglei Wu and Shuai Zhao*

Dalian University of Technology, Dalian City, Liaoning Province, China

Abstract

This work presents an obstacle avoidance path search algorithm based on a two-stage grid method, which is illustrated with a five-degree-of-freedom (5-dof) robot arm, with the first three joints grouped for obstacle space detection and the remaining two joints for extending obstacle space. Under the working space of the robotic arm, the corresponding discrete point sets of obstacle avoidance paths are obtained based on arbitrary feasible start and end positions. The discrete points free of obstacle collision are fitted with a non-uniform quintic B-spline curve, and the node interval of the B-spline curve is optimized with the NSGA-II algorithm to obtain a better trajectory in terms of impact and energy consumption. The obstacle avoidance and trajectory planning algorithm is illustrated with the 5-dof robotic arm, and the results show that the presented obstacle avoidance trajectory planning algorithm can effectively achieve obstacle avoidance and trajectory performance optimization.

Keywords: Collision Detection; Joint Space; Grid Method; Trajectory Optimization

Introduction

Robotic arms are widely used in agriculture, industry, medical care, military and other fields, bringing much convenience to human life [1]. However, the practical application of the robotic arm needs to take into account the influence of the surrounding obstacles, and the obstacle avoidance path planning technology is also a major hotspot in the field of robotics research at present. According to the difference of path planning solution space, trajectory planning can be divided into Cartesian space and joint space [2]. The planning in Cartesian space, whose object is the actuator at the end of the robotic arm, has the advantage of facilitating the intuitive demonstration of the posture of the end actuator in the obstacle avoidance process, whereas, when mapped to the relevant joint motor control, it needs to carry out the relevant inverse solution computation, which consumes a large amount of time and cost [3]. The trajectory planning in the joint space, whose objects of operation are the kinematics parameters such as the joint angle of the robotic arm, the joint velocity, the joint acceleration and other kinematics parameters, can save the computational amount of the inverse kinematics solving part, and at the same time, the trajectory planning in joint space can effectively avoid the singularity problem of the robotic arm that occurs in the Cartesian space trajectory planning [4].Therefore, the obstacle avoidance motion planning is carried out in the joint space.

Common path planning algorithms include A* algorithm [5], Dijkstra's algorithm [6], artificial potential field method [7], PRM (Probabilistic Roadmaps) algorithm [8], RRT (Rapidly-Exploring Random Tree) algorithm [9], etc. They prove their timeliness for path searching in two-dimensional planes, however, their performance in three-dimensional and high-dimensional spaces is lacking, and they need to be improved accordingly for specific applications [10]. Wu et al. [11] address the problem of difficult passage in narrow spaces by dividing the extended space into numbers and proposing a random sampling strategy, which increases the possibility of breaking through the narrow barriers and improves the stability of the algorithm in general. Kang et al. [12] propose a rewiring method based on triangular inequalities based on the RRT-connect algorithm, which shortens the path planning time and makes the path length shorter. Yang et al. [13] proposed an EPF-RRT algorithm by combining the environmental potential field and RRT, and through UAV experiments, the method was proved to have better convergence and stronger planning capability than the traditional RRT. Cheng et al. [14] proposed a GMM/GMR-MPRM algorithm by integrating GMM/GMR algorithms on the basis of PRM algorithm, which can increase the free space sampling density at narrow passages and reduce the redundancy of sampling points in free space. Guo et al. [15] constrain the sampling region on the basis of the RRT-Connect algorithm

OPEN ACCESS

Correspondence:
Guanglei Wu, Dalian University of Technology, Dalian City, Liaoning Province, China, Tel: 15940994026; E-mail: gwu@dlut.edu.cn

Received Date: *07 May 2026*
Accepted Date: *11 Jun 2026*
Published Date: *12 Jun 2026*

Citation:
Guanglei Wu, Shuai Zhao. An Obstacle Avoidance Trajectory Planning Algorithm for Robotic Arm Based on Two-Stage Grid Method. WebLog J Robot Appl. wjra.2026.f1202. https://doi.org/10.5281/zenodo.20963146

to ensure the directionality of each search, and at the same time smooth the searched paths by using gradient descent to remove large-angle turns and make the search paths shorter and smoother. Klemm [16] combines the asymptotic optimization idea of the RRT* algorithm with RRT-Connect, and conducts experimental validation on the relevant robotic arm, proving that this algorithm is more efficient in searching and obtains smoother paths compared to the RRT algorithm. In recent years, with the development of artificial intelligence and deep learning, some intelligent algorithms are also gradually applied to path planning. Ma et al. [17] proposed a path planning algorithm based on sixth degree polynomials, which defines the to-be-determined coefficients as chromosomes, and combines the genetic algorithm for solving and obtains collision-free paths. Zhang et al. [18] proposed an improved potential field ant colony algorithm based on the artificial potential field method, effectively overcoming the problem of local optimality of ant colony algorithm and achieving a balance of search capability and convergence speed. algorithm, effectively overcoming the problem of local optimality of the ant colony algorithm, and realizing the balance of search ability and convergence speed. Zhang et al. [19] proposed an improved particle swarm optimization algorithm, which optimizes the search ability by dynamically adjusting the inertia weights and the learning factor, and introduces multiple populations for planning to improve the robustness and the quality of solution of the algorithm. Arup et al. [20] proposed a hybrid algorithm which combines the firefly algorithm and Q-learning, and the firefly algorithm can be used for solving the collision-free paths by Q-learning adaptive Adjusting the relevant parameters in the firefly algorithm improves the adaptive ability and convergence speed of the algorithm, and experiments in complex environments prove the effectiveness and practicability of this algorithm. Gao et al. [21] proposed an improved obstacle avoidance path planning algorithm combining RRT* and Back Propagation Neural Network (BPNN), which reduces the unnecessary searches by dividing the space and discretizing obstacles, and avoids blind sampling in the obstacle-intensive area by utilizing BPNN neural network prediction. region blind sampling, which significantly improves the efficiency of path planning.

The actual operation process of the robotic arm needs to consider the smoothing treatment of the relevant paths, and a reasonable planning algorithm can get a trajectory with better smoothness, shorter movement time, and more stable movement process [22]. The trajectory planning algorithms in joint space mainly include linear interpolation, polynomial interpolation [23] and spline curve interpolation [24]. Wu et al. [25] proposed a trajectory planning method based on fourth-order S-curve, which divides each acceleration segment into three independent phases, so that the whole trajectory curve has fifteen segments, and compared with the traditional third-order S-curve, the fourth-order S-curve is able to successfully realize the continuity of the additive acceleration, and to improve the accuracy of the robot motion. Xiao et al. [26] proposed a kind of S-shaped fitting velocity curve using two hyperbolic tangent functions to construct the S-shaped fitting velocity curve at the articulation point of S-shaped curve in different stages of motion, which effectively overcomes the disadvantages of the high-order non-trivial and system shock at the articulation point of acceleration, uniform speed, and deceleration. Liu et al. [27] used fifth degree polynomials to plan the trajectory of joint space, and obtained a continuous and smooth acceleration trajectory, but there is an obvious abrupt change in the leap of the generated trajectory. Marek [28] proposed a method to plan smooth trajectories using asymmetric polynomials, taking into account constraints on velocity, acceleration, and leapfrog, and compared it to S-curves, proving its advantages. Dong et al. [29] proposed an improved B-spline curve construction method, which makes the generated curve pass through all the original type-valued points by increasing the control vertices, avoiding solving the multivariate system of equations and simplifying the computation, and the circular arc experiments verified the validity of the method. Moreover, in order to meet the demand of energy efficient production, it is also necessary to consider the problem of trajectory optimization, that is, to meet the requirements of high efficiency, low impact, and low energy consumption of the robotic arm, etc. Zhang et al. [30] propose an improved particle swarm trajectory optimization algorithm under multiple constraints, introducing an inertia weight update strategy based on chaotic mapping, which enables particles to jump out of the local optimum in the late iteration, balancing the global exploration ability and local exploitation ability of particles. Wu et al. [31] proposed an improved multi-objective optimization algorithm for the NSGA-II algorithm, which is easy to fall into the defects of local optimum, and improved the generation of new population by setting the population optimization strategy, and the experimental results proved that the improved algorithm solves the optimal solution of Pareto with a more uniform distribution and shorter iteration time. Kucuk [32] chose the combination of seventh polynomials and cubic B-splines, and used particle swarm algorithms to obtain the optimal time of the trajectory. spline combination and used particle swarm algorithm for optimization to construct the minimum time smooth motion trajectory with zero impact at the beginning. Huang et al. [33] optimized trajectories with kinematic constraints based on an ant colony algorithm and demonstrated the reduction of energy consumption after trajectory optimization on a hair transplantation robotic arm.

Based on the above analysis, the main content of this paper is to study the five-degree-of-freedom robotic arm obstacle avoidance motion planning in joint space, where a two-stage obstacle avoidance path search algorithm is proposed based on the degree-of-freedom splitting of the grid method 3-2. Afterwards, the discrete points are fitted using B spline curves, and the B spline node intervals are optimized using the NSGA-II algorithm, in order to generate the obstacle avoidance trajectories with lower impact and better energy consumption.

The remaining of the paper is structured as follows. section 2 introduces the principles of the proposed algorithm, including obstacle avoidance path planning, trajectory planning and optimization. In Section 3, the implementation of the proposed algorithm is illustrated with the 5-dof robot arm as a specific case study. In Section 4, the effectiveness of the proposed algorithm is verified by means of four sets of physical simulation experiments with CoppeliaSim. Finally, conclusions are drawn in the last section.

Two-stage Obstacle-avoidance Path Search Algorithm

This section firstly introduces the path planning algorithm for obstacle avoidance, which is illustrated along with a five-degree-of-freedom (5-dof) serial robotic arm, as shown in Figure 1. The five active joints of the robot arm are placed at points O_0-O_4, while, the end-effector attached at point O_5 is not involved in the planning procedure. The motion planning procedure consists of the establishment of a collision database, the obstacle avoidance search algorithm, trajectory

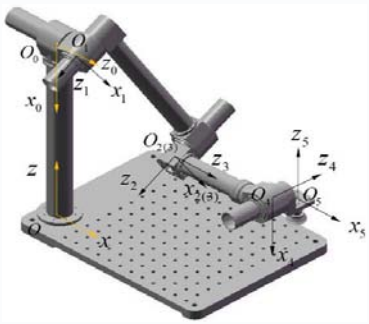

Figure 1: CAD model of a 5-dof robotic arm with coordinate systems.

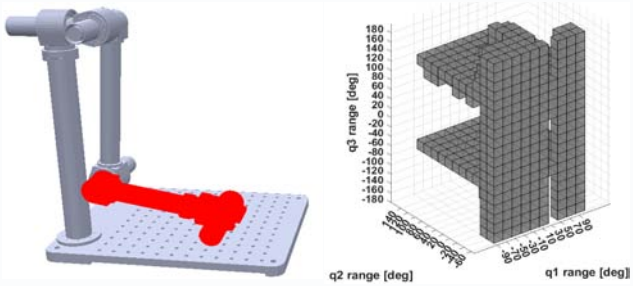

Figure 2: Self-collision space construction of the first three joints (left: robot end-effector collision with base; right: the C_{obs} space of the first three joints in a three-dimensional space).

planning and optimization, which will be introduced as follows.

Establishment of collision database

In this work, the obstacle avoidance planning is carried out in joint configuration space (*C*-space), wherein the joint displacements can be written in a vector (q_1, q_2, q_3, q_4, q_5) to represent the generalized coordinates. Accordingly, the regions in the *C*-space that collide with obstacles are named as obstacle space C_{obs}, while, the remaining reachable space are called as free-region space C_{free}. The task of obstacle avoidance motion planning in *C*-space means to planning a path to connect the start and end positions in the free-region space C_{free} [34].

Figure 2 depicts the self-collision configurations of the first three joints of the robot arm in Figure 1, e.g., the configuration when the robot end-effector has a collision with the base, which can be represented in a three-dimensional space, with the gray areas representing the obstacle space C_{obs}. It is known that the dimensions of the *C* space are equal to the dofs of the robotic manipulator, on the other hand, with the increasing dofs of the robotic arm, the computational burden of collision detection will increase accordingly, together with the increased difficulties in the graphical representation, which cannot meet the real-time planning requirements in turn.

Based upon the previous analysis, this work adopts a combined offline-online approach to build the joint space in obstacle avoidance, of which the flowchart is shown in Figure 3. In the offline stage, the Cartesian space of the robot arm is discretized, in order to obtain the collision configurations, i.e., the establishment of the total collision database (OCQD, namely, Obstacle Collision Query Database). In the online stage, the discrete workspace is used to identify the collision configurations for the identified obstacles, where the self-collision query database (SCQD) of the robotic arm is found, to serve as the collision database for searching space of obstacle avoidance.

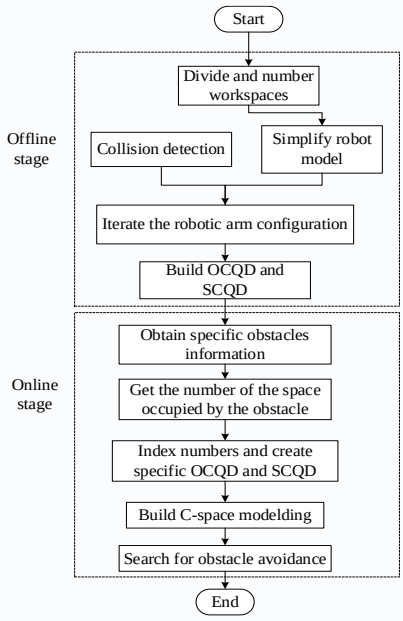

Figure 3: Flowchart of the offline-online approach for obstacle avoidance.

The foregoing approach is applicable to different serial robotic manipulators, and high computational efficiency of the obstacle collision database construction can be realized by tuning the parameters. By contrast, the SCQD can take less time to be built by checking all the joints in a predefined sequence.

Obstacle avoidance search algorithm

Considering that it is difficult to visualize the *C* space when the dof of the robot is more than three, this section presents proposes a two-stage obstacle avoidance search algorithm, namely, two *C* subspaces that includes the first three and the remaining two joints, respectively. In this approach, the collisions encountered by the first three joints are used as the main basis discriminative, and collision ratio of remaining two joints is used as the secondary search basis. The related search principle can be expressed in the following steps:

Step 1: Based on the given obstacle information, the corresponding collision database is established. With the given start and end position coordinates, the inverse solution to the robot arm is solved to select the appropriate start q_s and end q_g configurations, which ensures that the robot links are as far as possible away from the obstacle at these two positions. The distance discrimination formula is shown in Eq. (1), where $dis(link_i,o_j)$ denotes the distance between the axis of the link i and the obstacle *J*.

$$Dis = \sum_{i=1}^{m}\sum_{j=1}^{n} dis(link_i, o_j) \quad (1)$$

Step 2: The first three joints without duplications in the collision database are extracted, and the number of all configurations of the remaining last two joints is used as the surrogate value of the current point location. For the points that are not in the collision database, their surrogate value is 0. After data preprocessing, the first three joint configurations correspond to a point in the 3D joint space, and each point carries a surrogate value. The improved A* algorithm is adopted to search for the first three joints, if a series of extended points search in the same direction, and the intermediate points of the value of the generation of 0, the path can be appropriately processed. The search process takes the length of the search path and the cost into

Algorithm 1: Obstacle Avoidance Search for The First Three Joints.

Input: q_{init}: Initial joint configuration; q_{goal}: Final joint configuration; *qnum_std*: Initially given cost value discrimination conditions; $OCQD_s$: Specific obstacle collision database extraction; *SCQD*: Self-collision Query Database;	
Output: Obstacle avoidance path scatter points(q_{path});	
1:	**Build** the cost of all combinations of the first three joints(q_{123});
2:	**if** the *cost* in q_{123} > *qnum_std*
3:	**Set** *cost=Nan*;
4:	**endif**
5:	**Search path:** using the improved *A* algorithm*
6:	**Record** *visited_nodes* and their information of the last two joints in q_{path}
7:	**Finish**

consideration, which is written as follows:

$$F = e_1 \times Cost + e_2 \times Steps \tag{2}$$

where *Cost* is total generation value of the search path, *Steps* is the length of the search path, and e_1, e_2 are the proportion indicators of the two factors respectively.

The cost formula for the A* search algorithm can be expressed as follows:

$$F = G + H \tag{3}$$

where *G* represents the cost of moving from the starting point to the current point, *H* represents the cost of moving from the current point to the end point, *F* represents the total cost of moving the selected path from the starting point to the end point, and A* search algorithm is usually selected according to the minimum *F* [35].

After combining Eqs. (2) and (3), the cost function used for the first three joint searches can be expressed as:

$$F = e_1 \times (G + H) + e_2 \times Steps \tag{4}$$

Moreover, in order to ensure the matching efficiency of the subsequent two joints during the search process, the reference value *qnum_std* is set in advance, and if the generation value at any point is more than *qnum_std*, it is considered that the generation value here is infinite, in which the current point is automatically excluded from the path search process.

The search process for the first three joints using the improved A* algorithm can be represented in the following pseudo-code of algorithm 1.

Step 3: After finding the path scatter values of the first three joints, the next step is to perform the matching operation on the last two joints. The collision discrete points of the last two joints are converted into square region information according to the joint sampling distance, and the collision information of the converted two joints can be displayed using the black and white square occupancy map under the 2D plane (e.g., similar to the representation in Figure 2). The detailed process will be given in Section 3, for which the formula for converting the collision discrete points into collision square information is as follows:

$$square\_collision = \begin{cases} 1, & seria\ umber \geq 2 \\ 0, & else \end{cases} \quad \max\_number = 4 \tag{5}$$

In Eq. (5), according to the joint sampling, distance division of the two-dimensional plane is used for the corresponding number of small square regions. When this square perimeter of at least two of the four points in the collision, the expansion of the whole square region is recognized as the collision region.

In order to ensure that the latter two joints in the whole obstacle avoidance path search process changes in the small fluctuations and the overall feasibility of the robotic arm obstacle avoidance, in addition to the start and end points, the remaining points at the joints of the matching, are used in the former and latter points and the current points of the collision area information map superposition operation. This procedure is similar to the "Boolean sum" operation, which can effectively ensure the success of obstacle avoidance of each rod of the robot arm. After establishing the collision area map corresponding to each point, the eight points around the current point are used as the pre-expansion direction, and the distance from each obstacle square is used as the final judgment basis to select the final value of joints 4 and 5.

Trajectory planning and optimization

After obstacle avoidance path search, a discrete sequence of path points is found, and the actual operation of the robotic arm requires smoothing the trajectory passing the discrete points. Considering that the number of discrete path points searched by the obstacle avoidance algorithm is uncertain for different obstacles, it is difficult to use the conventional fixed-order polynomials, such as the 3-4-3, etc., for fitting, and moreover, lower degree polynomials cannot satisfy the requirements of acceleration smoothing. While high degree polynomials or spline curves will encounter the difficulties in solving and local distortion problems. Therefore, the discrete-point fitting in this work is carried out by choosing the quintic non-uniform B-splines with better fitting effect and applicable to uncertain points.

A B-spline curve [36] can be viewed in the form of a multi-segment polynomial accumulation. A k-times B-spline curve passing through *f* key points can be converted into:

$$C(u) = \sum_{i=0}^{n} Q_i N_{i,k}(u) \tag{6}$$

In Eq. (6), *n* is the number of control vertices, *n=f+k+1*, *k* is the order of the B-spline curve, we set it to 5 in this paper, Q_i*(i=0, 1 ,…, n)* is the control vertex of the spline curve, $N_{i,k}(u)$ is the sub-normed B-spline basis function, which can be obtained recursively by the de Boer-Cox formula, as shown in Eq. (7).

$$\begin{cases} N_{i,k}(u) = \dfrac{u - u_i}{u_{i+k} - u_i} N_{i,k-1}(u) + \dfrac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \\ N_{i,0}(u) = \begin{cases} 1 & u_i \leq u \leq u_{i+1} \\ 0 & \text{else} \end{cases} \end{cases} \tag{7}$$

In Eq. (7), u_i *(i=0, 1, …, m=n+k+1)* is the curve normalized node vector parameter, for a non-uniform B-spline curve can be expressed as:

$$\{\underbrace{0, 0, \ldots, 0}_{k+1}, u_{k+1}, u_{k+2}, \ldots, u_n, \underbrace{1, 1, \ldots, 1}_{k+1}\} \tag{8}$$

The process of solving the B spline curve can be shown by the following procedure:

$$Q = A^{-1}K \tag{9}$$

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, K = \begin{bmatrix} K_1 \\ K_2 \end{bmatrix} \tag{10}$$

$$K_1 = [P_0, P_1, \ldots, P_{f-1}]^T, K_2 = [0, 0, 0, 0, 0, 0]^T, K_1 \in R^{\wedge}(f, 1) \tag{11}$$

$$P_i = q_{m,i}, i \in (0, f-1), m \in (1, 5) \tag{12}$$

$$A_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ N_{1,k}(u_{k+2}) & N_{2,k}(u_{k+2}) & \cdots & N_{n,k}(u_{k+2}) \\ N_{1,k}(u_{k+3}) & N_{2,k}(u_{k+3}) & \cdots & N_{n,k}(u_{k+3}) \\ \vdots & \vdots & \cdots & \vdots \\ N_{1,k}(u_{k+f-1}) & N_{2,k}(u_{k+f-1}) & \cdots & N_{n,k}(u_{k+f-1}) \\ 0 & 0 & \cdots & 1 \end{bmatrix} \in \mathrm{R}^{f\times n} \tag{13}$$

$$A_2 = \begin{bmatrix} Coe_{1,1} & Coe_{2,1} & & & \cdots & & & & \\ Coe_{1,2} & Coe_{2,2} & Coe_{3,2} & & \cdots & & & & \\ Coe_{1,3} & Coe_{2,3} & Coe_{3,3} & Coe_{4,3} & \cdots & & & & \\ & & & & \cdots & & & Coe_{n-1,1} & Coe_{n,1} \\ & & & & \cdots & & Coe_{n-2,2} & Coe_{n-1,2} & Coe_{n,2} \\ & & & & \cdots & Coe_{n-3,3} & Coe_{n-2,3} & Coe_{n-1,3} & Coe_{n,3} \end{bmatrix} \in \mathrm{R}^{6\times n} \tag{14}$$

For each of the five joints discrete path points are fitted and $q_{m,i}$ denotes the discrete point i of the joint m.

The impact and energy consumption are used as optimization indexes to optimize the fitted trajectory. During the motion process, the impact of the robotic arm is mainly related to the leap degree of each joint, and the energy consumption of the robotic arm is mainly related to the motor output power and the angular acceleration of each joint. The impact and energy consumption during the movement of the robotic arm can be approximately characterized by Eqs. (15) and (16).

$$f_1(\mathbf{x}) = \sum_{i=1}^{5}\sqrt{\frac{1}{T}\int_0^T (j_i)^2\,dt} \tag{15}$$

$$f_2(\mathbf{x}) = \sum_{i=1}^{5}\sqrt{\frac{1}{T}\int_0^T (\alpha_i)^2\,dt} \tag{16}$$

The mathematical model for multi-objective optimization based on the above analysis is as follows:

$$\begin{aligned} &\min\ \mathbf{f}(x) = \min[f_1(\mathbf{x}), f_2(\mathbf{x})] \\ &over\ \ \mathbf{x} = \{\underbrace{0,0,...,0}_{k+1}, u_{k+1}, u_{k+2}, ..., u_n, \underbrace{1,1,...,1}_{k+1}\} \\ &s.t.\begin{cases} |\omega_i(t)| \le \omega_{i\max} \\ |\alpha_i(t)| \le \alpha_{i\max} \\ |J_i(t)| \le J_{i\max} \\ |\tau_i(t)| \le \tau_{i\max} \\ \theta_{i\min} \le \theta_i \le \theta_{i\max} \\ c'(u)|_{u=u_0} = v_0, c'(u)|_{u=u_t} = v_1 \\ c''(u)|_{u=u_0} = a_0, c''(u)|_{u=u_t} = a_1 \\ c'''(u)|_{u=u_0} = j_0, c'''(u)|_{u=u1} = j_1 \end{cases} \end{aligned} \tag{17}$$

The optimization process changes the acceleration and jerk of the trajectory by changing the length distribution of the B-spline node interval. Moreover, ω_i, α_i, j_i and τ_i are the instantaneous angular velocity, angular acceleration, angular jerk, and instantaneous output torque of the joint i, and ω_{imax}, α_{imax}, j_{imax} and τ_{imax} are the maximum angular velocity, angular acceleration, angular jerk, and maximum torque of the joint, respectively. For instance, the previous constraints of the robot arm shown in Figure 1 are listed in Table 1.

Table 1: Motion constraints for each joint.

Constraints	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5
$\omega_{max}(rad/S)$	1.7π	1.7π	1.7π	1.7π	1.7π
$\alpha_{max}(rad/S^2)$	2.3π	2.3π	2.3π	2.3π	2.3π
$J_{max}(rad/S^3)$	8	8	8	8	8
$\tau_{max}(Nm)$	45.65	45.65	45.65	20.42	20.42

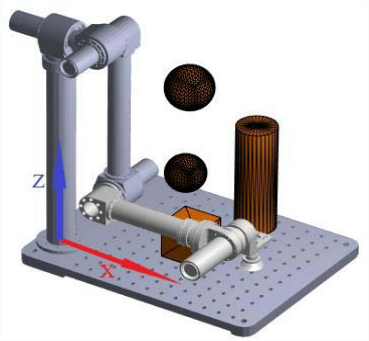

Figure 4: Schematic diagram of robotic arm obstacle environment.

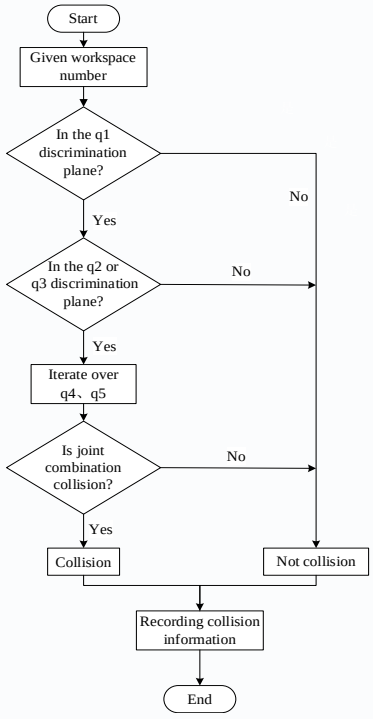

Figure 5: Joint 1-23-45 collision discrimination process.

Case Study

Discrete points-based path searching

This section presents the numerical example of the previous obstacle avoidance search algorithm, which is applied to the 5-dof robot arm., The obstacle environment is established with the robot arm base as the world coordinate system as shown in Figure 4, and the obstacle avoidance information is shown in Table 2. Considering the modeling time and search efficiency of the obstacle database, the discrete distance of the working space of the robotic arm is set to 20 mm, and the discrete angle of each joint is set to 20°.

In the first step, the establishment of obstacle collision database is carried out. Considering the special characteristics of the structure of the in-group verification robotic arm, this work presents an obstacle collision discrimination algorithm with the hierarchical enclosing box as the subjective idea, and plane distance discrimination as the specific implementation means in the process of the establishment of obstacle total collision database (OCQD). Compared to direct collision discrimination, this method can realize the establishment of

Table 2: Specific obstacle information.

Body-center coordinate	Cylinder height(m)	Cylinder radius(m)	Cube Length(m)	Sphere radius(m)	Start point(m)	End point(m)
(0.25,0.15,0.25)	—	—	—	0.05	$\begin{bmatrix}0.460\\0.088\\0.076\end{bmatrix}$	$\begin{bmatrix}0.200\\0.400\\0.100\end{bmatrix}$
(0.16,0.3,0.4)	—	—	—	0.06		
(0.3,0.1,0.1)	—	—	0.1	—		
(0.35,0.3,0.2)	0.15	0.05	—	—		

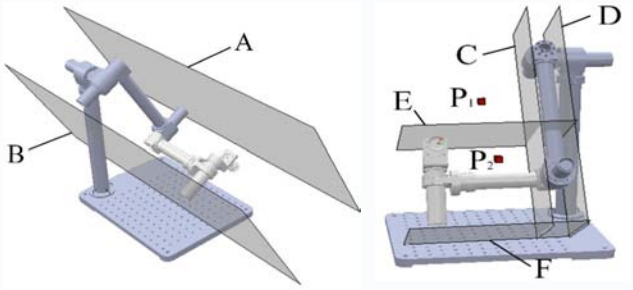

Figure 6: Collision area discrimination: (a) Joint 1 discrimination (b) Joint 2 and Joint 3 discrimination.

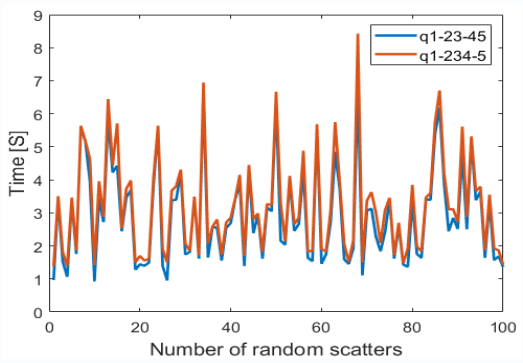

Figure 7: Comparison of time-consuming database setup.

collision database faster.

Figure 5 shows the flow chart of joint 1-23-45 layer-by-layer collision discrimination, because the movement of the joints and the movement of the numbered workspace is relative, here, the numbered workspace is taken as the object to illustrate the principle of layer-by-layer collision discrimination. Figure 6 shows the schematic diagram of the establishment of the relevant discrimination plane under the configuration of a robotic arm, based on the center of the joints, the maximum length of the remaining joints as the expansion distance to both sides to expand the establishment of the regional plane.

Considering the collision discrimination way of joint 1-234-5 at the same time, 100 random spatial numbering units are randomly selected to compare the time of the two ways to establish the corresponding collision database. The results are shown in Figure 7, and after data analysis, it is known that the first method can be faster than the second method by 0.8s on average. In this light, the first method, i.e., the method of joints 1-23-45 for the establishment of the whole obstacle collision database, is selected.

Based on the inverse solution of the robotic arm, according to the distance from each obstacle, the beginning and end configurations are determined as q_{s1} =[-5,0,0,0,0] and q_{g1} =[30,-30,10,-15,-20], and then

Table 3: First three joints search path.

Serial number	Joint key points	Serial number	Joint key points
1	(-5 0 0)	11	(50 40 -20)
2	(-10 0 0)	12	(90 40 -20)
3	(-30 0 0)	13	(90 -20 -20)
4	(-30 20 0)	14	(50 -20 -20)
5	(-30 40 0)	15	(90 -40 -20)
6	(-30 60 0)	16	(50 -40 0)
7	(-10 60 0)	17	(30 -40 0)
8	(10 60 0)	18	(30 -40 20)
9	(10 60 -20)	19	(30 -30 10)
10	(50 60 -20)		

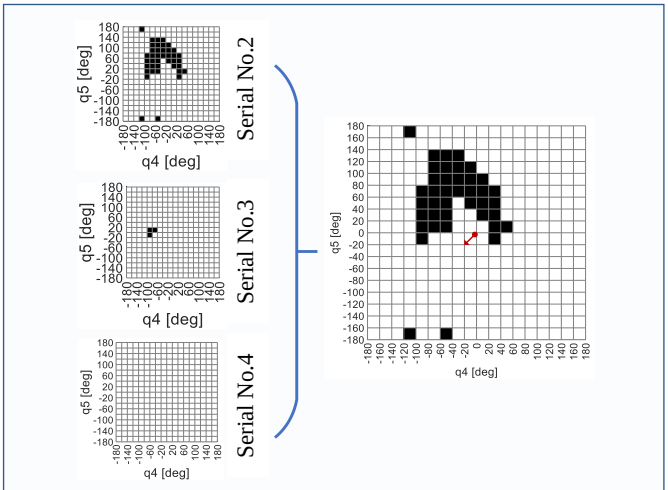

Figure 8: Joint 4 and joint 5 finding process at serial No.3.

based on the established obstacle collision database, the beginning and end points are converted to the standard point positions q_s =[-10,0,0,0,0] and q_g =[30,-40,20,-20,-20]. Searching according to the above 3-2 degree of freedom segmentation principle, the first three joints are obtained with the scatter values of the obstacle avoidance paths, as shown in Table 3.

Figure 8 shows the search process for joints 4 and 5 at sequence numbers. First, the point information is expanded into square occupancy information as shown in Eq. (5), and then the collision information at sequence numbers 2 and 4 is fused to select the point based on the distance to the obstacle.

The full path found after the previous method is as follows. When the RRT algorithm under five-dimensional space for obstacle avoidance path searching is used, the values of discrete point sequences found are listed in Table 5, and the corresponding pseudo code of RRT algorithm is depicted as Algorithm 2.

Table 4: 3-2 Degree of freedom split finding complete path points.

Serial number	Joint key points	Serial number	Joint key points
1	(-5 0 0 0 0)	11	(50 40 -20 -20 -20)
2	(-10 0 0 0 0)	12	(90 40 -20 -20 -20)
3	(-30 0 0 -20 -20)	13	(90 -20 -20 -20 -20)
4	(-30 20 0 -20 -20)	14	(50 -20 -20 -20 -20)
5	(-30 40 0 -20 -20)	15	(90 -40 -20 -20 -20)
6	(-30 60 0 -20 -20)	16	(50 -40 0 -20 -20)
7	(-10 60 0 -20 -20)	17	(30 -40 0 -20 -20)
8	(10 60 0 -20 -20)	18	(30 -40 20 -20 -20)
9	(10 60 -20 -20 -20)	19	(30 -30 10 -15 -20)
10	(50 60 -20 -20 -20)		

Algorithm 2: Improved RRT algorithm for pathfinding.

Input: q_{init}: Initial joint configuration; q_{goal}: Final joint configuration; $OCQD_s$: Specific obstacle collision database extraction; $SCQD$: Self-collision Query Database;

Output: Obstacle avoidance path scatter points (RRT_{path});

1: *Tree.add* (q_{init}); q_{near}=q_{init};
2: **While** $i<N$ **do**
3: q_{expand} = **Expandpoint** (); i++
4: **if** q_{expand} ≠ collision
5: *Tree.add* (q_{expand});
6: **Update** q_{near};
7: **endif**
8: **if** q_{expand} = q_{goal}
9: **return GetPath** (Tree);
10: **return** Advanced;
11: **endif**
12: **End While**
13: **if** not find RRT_{path}
14: **return** Failure;
15: **endif**

Table 5: Paths found by the RRT algorithm.

Serial number	Joint key points	Serial number	Joint key points
1	(-5 0 0 0 0)	13	(90 60 20 -20 -20)
2	(-10 0 0 0 0)	14	(90 -20 20 -20 -20)
3	(-10 0 0 -20 0)	15	(90 -20 0 -20 -20)
4	(-10 0 0 -20 -20)	16	(70 -20 0 -20 -20)
5	(-10 0 20 -20 -20)	17	(70 -20 -20 -20 -20)
6	(-30 0 20 -20 -20)	18	(50 -20 -20 -20 -20)
7	(-30 20 20 -20 -20)	19	(50 -40 -20 -20 -20)
8	(-30 40 20 -20 -20)	20	(50 -40 0 -20 -20)
9	(-10 40 20 -20 -20)	21	(30 -40 0 -20 -20)
10	(-10 60 20 -20 -20)	22	(30 -40 20 -20 -20)
11	(10 60 20 -20 -20)	23	(30 -30 10 -15 -20)
12	(30 60 20 -20 -20)		

Using RRT algorithm, the path is found as follows. The motion of the end point of the path searched by comparing the foregoing two methods is shown in Figure 9. By analyzing the data, It is found

Table 6: Paths found by the RRT algorithm.

	Average time spent(s)	Average of path length ratios (RRT/3-2 splitting)
RRT algorithm	3.26	1.16
3-2 splitting algorithm	1.07	

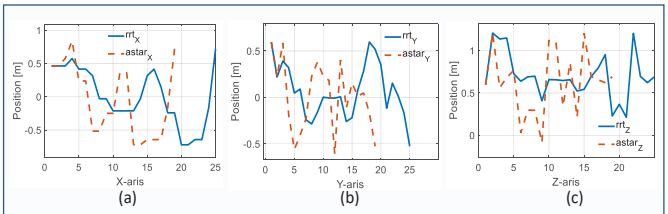

Figure 9: Motion of endpoints: (a)X-axis direction (b)Y-axis direction (c) Z-axis direction.

that the proposed algorithm is 0.32m shorter than the RRT algorithm, which is 3.5% shorter than that of the RRT algorithm.

The simulation is carried out 100 times using a cube and a sphere as obstacles with randomized obstacle sizes and start and end point locations, from which the time consuming and end point path lengths of the two methods are compared as shown in Table 6.

Trajectory optimization

The previous comparative analysis shows that the proposed algorithm has higher search efficiency than that of RRT algorithm in path search. After optimizing the node intervals of the B spline curve using the NSGA-II algorithm, the fitting of each discrete point is shown in Figure 10, and analysis of the data in the figure shows that the optimized B spline curve can successfully fit the scattered points.

The superiority of a multi-objective optimization problem cannot be judged only by the size of the objective function, and the Pareto dominance is also an important evaluation factor. In the process of multi-objective trajectory optimization using the NSGA-II algorithm,

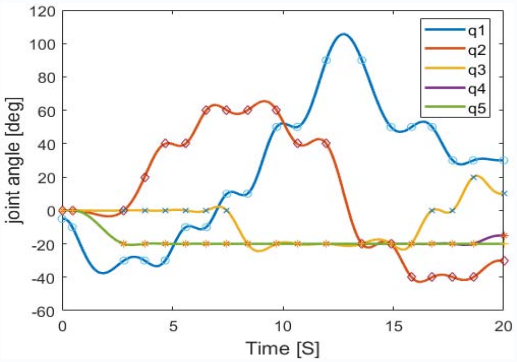

Figure 10: Scatterplot of five non-uniform B-spline fits.

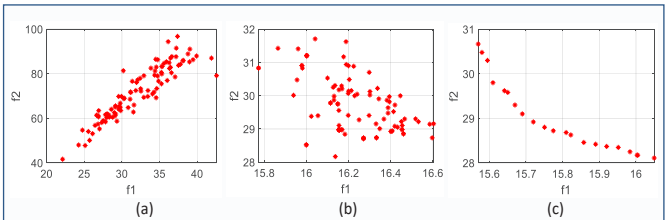

Figure 11: The process of optimizing the change in the value of the objective function.

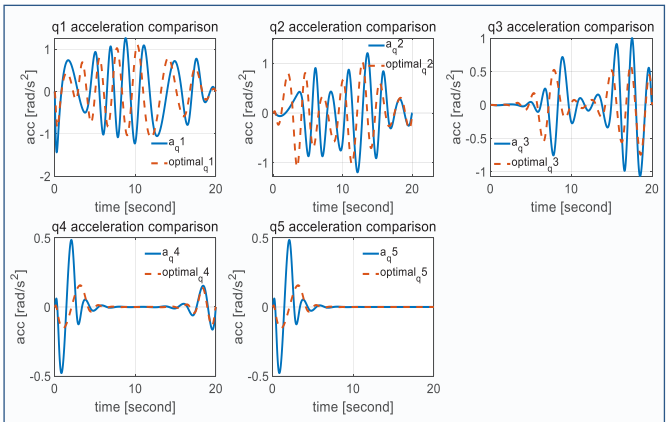

Figure 12: Acceleration comparison before and after optimization.

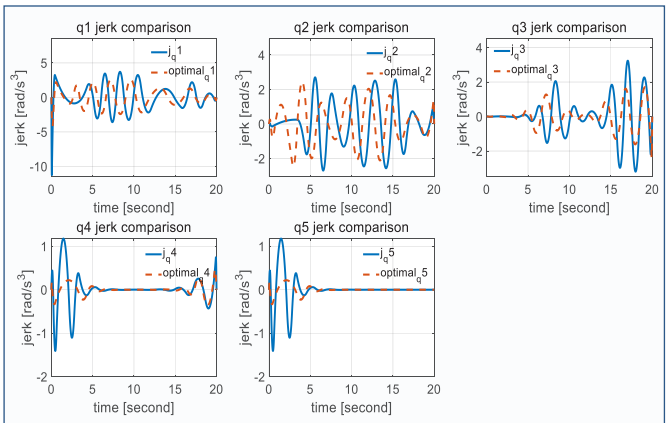

Figure 13: Jerk comparison before and after optimization.

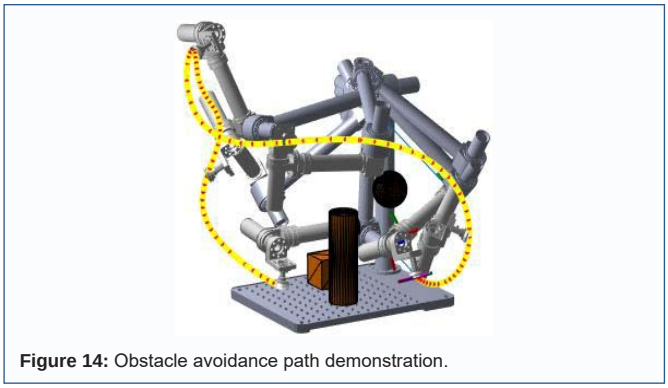

Figure 14: Obstacle avoidance path demonstration.

the trend of the relevant objective function values is shown in Figure 11, and the numerical solution of the Pareto frontier is obtained as shown in Figure 11(c).

The NSGA-II algorithm was applied to optimize the motion trajectory, comparing the acceleration and leap degree of each joint before and after the optimization as shown in Figure 12 and Figure 13.

As can be seen from the comparison of the acceleration and jerk curves in Figure 12 and Figure 13, there is a significant degree of reduction in amplitude and volatility after optimization. For acceleration, the maximum acceleration peaks of each axis are reduced by 5.1%, 52.5%, 36.3%, 20.2%, and 76.6%, respectively. For leaps, the maximum peak leaps of each axis are reduced by 18.2%, 55.6%, 25.9%, 37.4%, and 75.3%, respectively. The results show that the optimized trajectory shows a large improvement in both shock and energy consumption.

software [37]. CoppeliaSim can be used for the validation of robotic arm obstacle avoidance path planning. As shown in Figure 15, the software will give an error alert when the robotic arm bars collide with each other or with an obstacle.

Next, four sets of simulation experiments are conducted for different obstacle environment information to verify the accuracy of the proposed algorithm, and the relevant experimental information is shown in Table 7.

Figures. 16-19 show the results of four sets of simulation experiments with CoppeliaSim software, which record the position, velocity, moment, and change of the end trajectory point of each joint during the simulation, respectively.

From the simulation of the foregoing figures, it can be seen that the proposed algorithm can successfully realize the obstacle avoidance and satisfy the relevant constraints. The data from analytical and CoppeliaSim simulations are used to analyze the variation of each joint torque in detail. Figure 20 shows the variation of each joint torque in experiment 4, from which it can be seen that the torque changes of the two simulations follow the same trend, indicating the correctness of the CoppeliaSim simulation operations.

Figure 21 shows the root-mean-squared (RMS) errors for each joint over the four experiments and the trend of change for each joint with the experiment progresses. It can be seen that the error of each joint meets the requirements in general, which indicates that CoppeliaSim simulation can be used instead of Matlab to verify the obstacle avoidance motion and validates the proposed obstacle

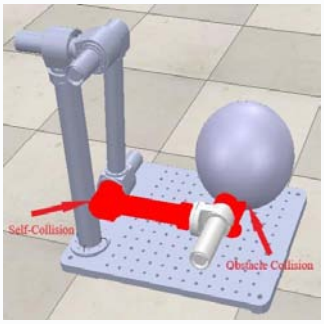

Figure 15: Collision schematic display.

CoppeliaSim Based Simulation Verification

In this section, a couple of sets of simulations are carried out with Matlab-CoppeliaSim co-simulation for different obstacle environments to verify the effectiveness of the algorithm proposed previously, and the changes in the driving moments of the joints under the two simulations are compared synchronously to prove the accuracy of the simulation. The obstacle avoidance trajectories used in this section are all the trajectories with NSGA-II optimization.

CoppeliaSim (formerly known as V-REP, namely, virtual robot expeimentation platform), developed by CoppeliaRobotics, Switzerland, is a feature-rich robot simulation software that allows you to set up different simulation environments to analyze robots according to your needs, making it an ideal robot simulation

Table 7: Environmental information on obstacles.

Experiment No.	Obstacle information					Start point(m)	End point(m)
	Body-center coordinate	Cylinder height(m)	Cylinder radius(m)	Cube Length(m)	Sphere radius(m)		
1	(0.25,0,0.4)	—	—	—	0.1	$\begin{bmatrix}0.461\\0.114\\0.064\end{bmatrix}$	$\begin{bmatrix}0.130\\0.373\\0.139\end{bmatrix}$
	(0.4,0.3,0.13)	0.2	0.1	—	0.06		
2	(0.47,0.2,0.13)	—	—	0.2	—	$\begin{bmatrix}0.461\\0.010\\0.053\end{bmatrix}$	$\begin{bmatrix}0.140\\0.440\\0.060\end{bmatrix}$
	(0.3,0.15,0.48)	—	—	0.1	—		
	(0.24,0.02,0.45)	—	—	0.1	—		
3	(0.25,0,0.08)	0.1	0.08	—	—	$\begin{bmatrix}0.461\\0.114\\0.064\end{bmatrix}$	$\begin{bmatrix}0.127\\0.314\\0.061\end{bmatrix}$
	(0.25,0.18,0.18)	0.3	0.1	—	—		
	(0.47,0.3,0.105)	—	—	0.15	—		
	(0.1,0.25,0.3)	—	—	—	0.075		
4	(0.47,0.2,0.13)	—	—	0.1	—	$\begin{bmatrix}0.461\\0.516\\0.296\end{bmatrix}$	$\begin{bmatrix}0.461\\0.114\\0.064\end{bmatrix}$
	(0.3,0.1,0.08)	—	—	0.1	—		
	(0.05,0.15,0.13)	0.2	0.08	—	—		
	(0.4,0.15,0.4)	—	—	—	0.1		
	(0.25,0.2,0.4)	—	—	—	0.1		

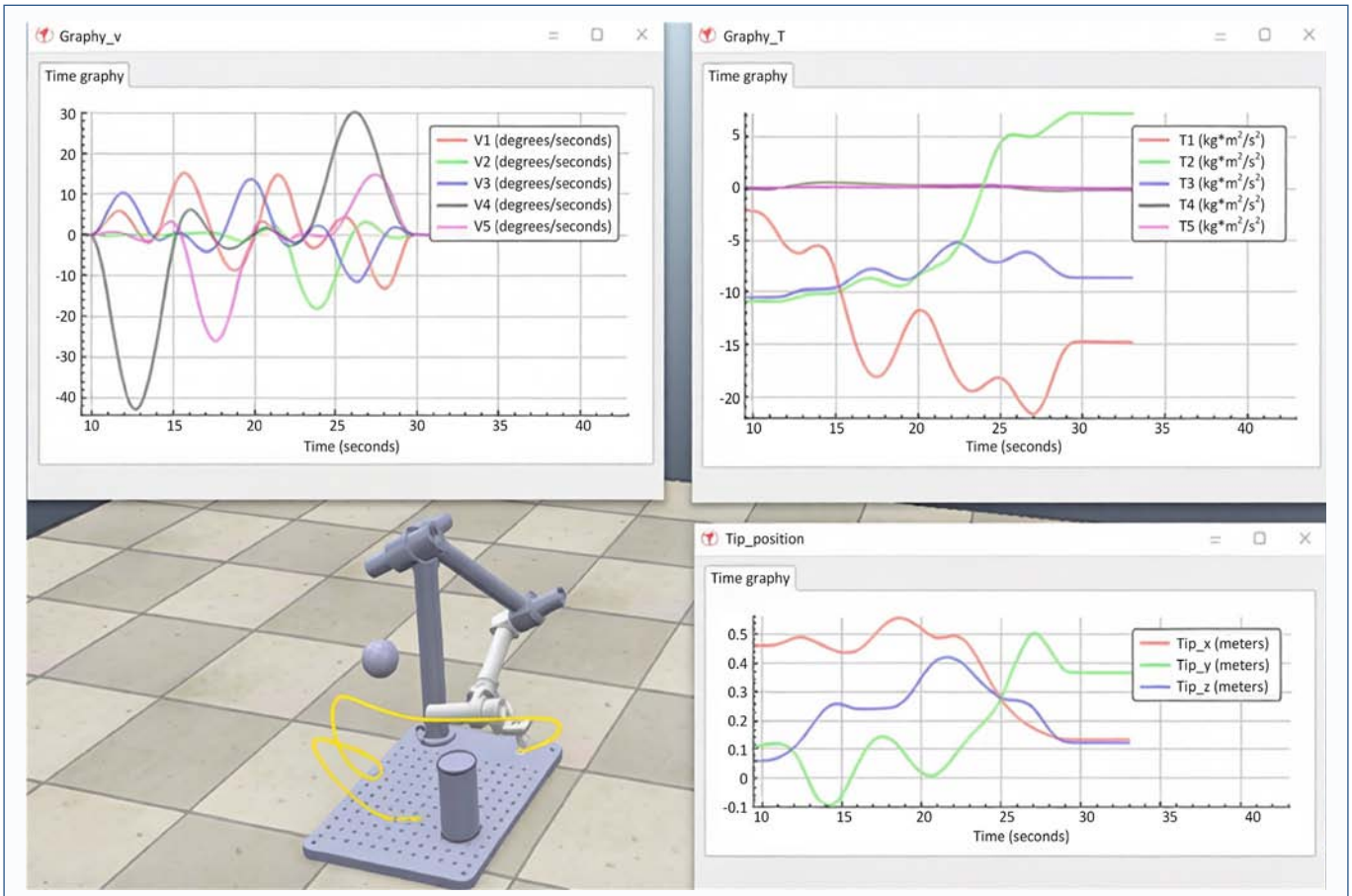

Figure 16: Experiment 1 simulation.

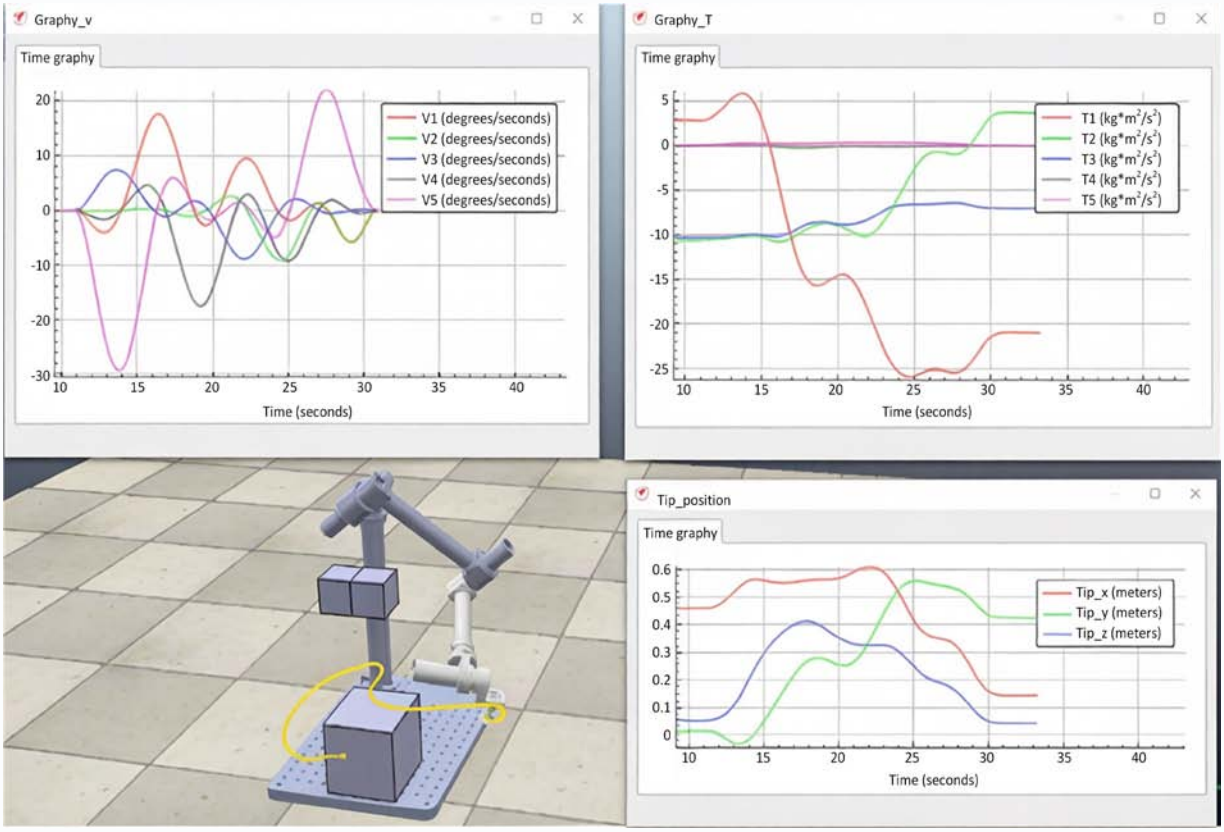

Figure 17: Experiment 2 simulation.

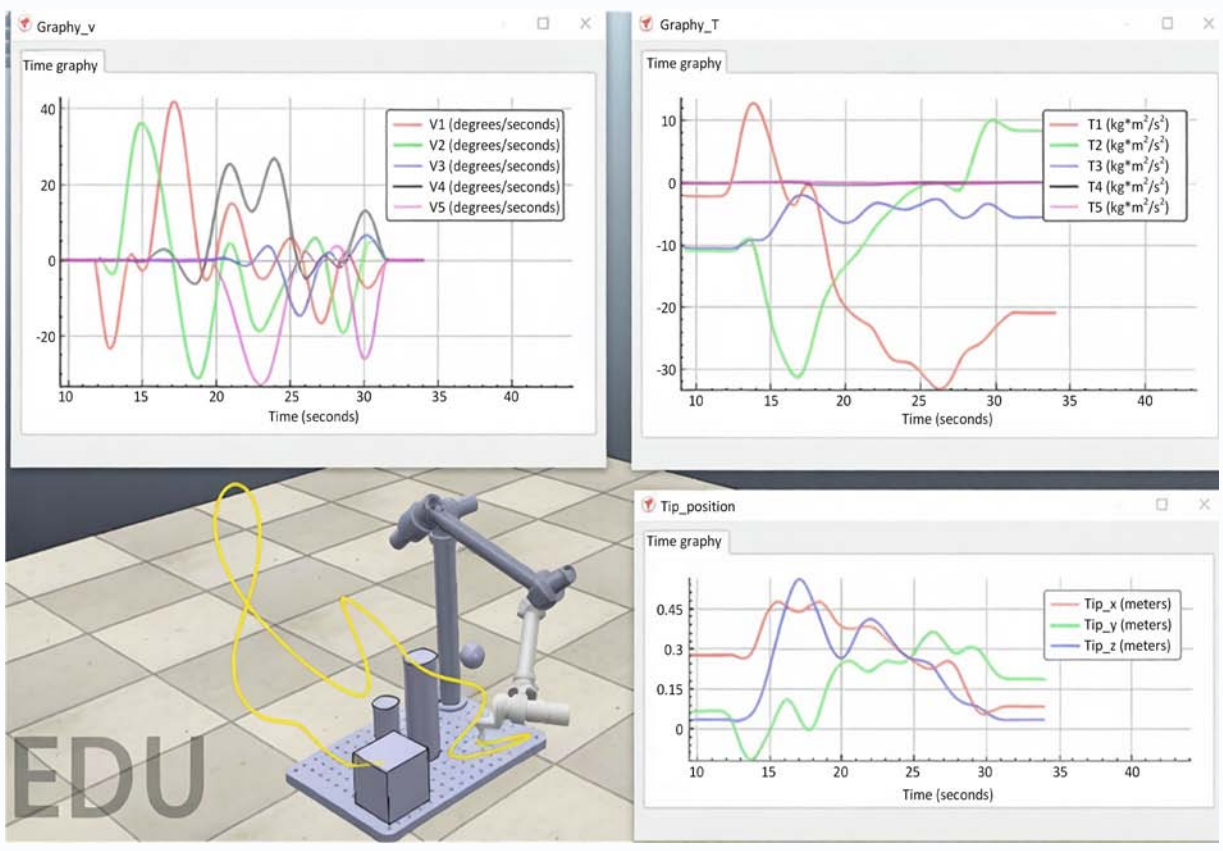

Figure 18: Experiment 3 simulation.

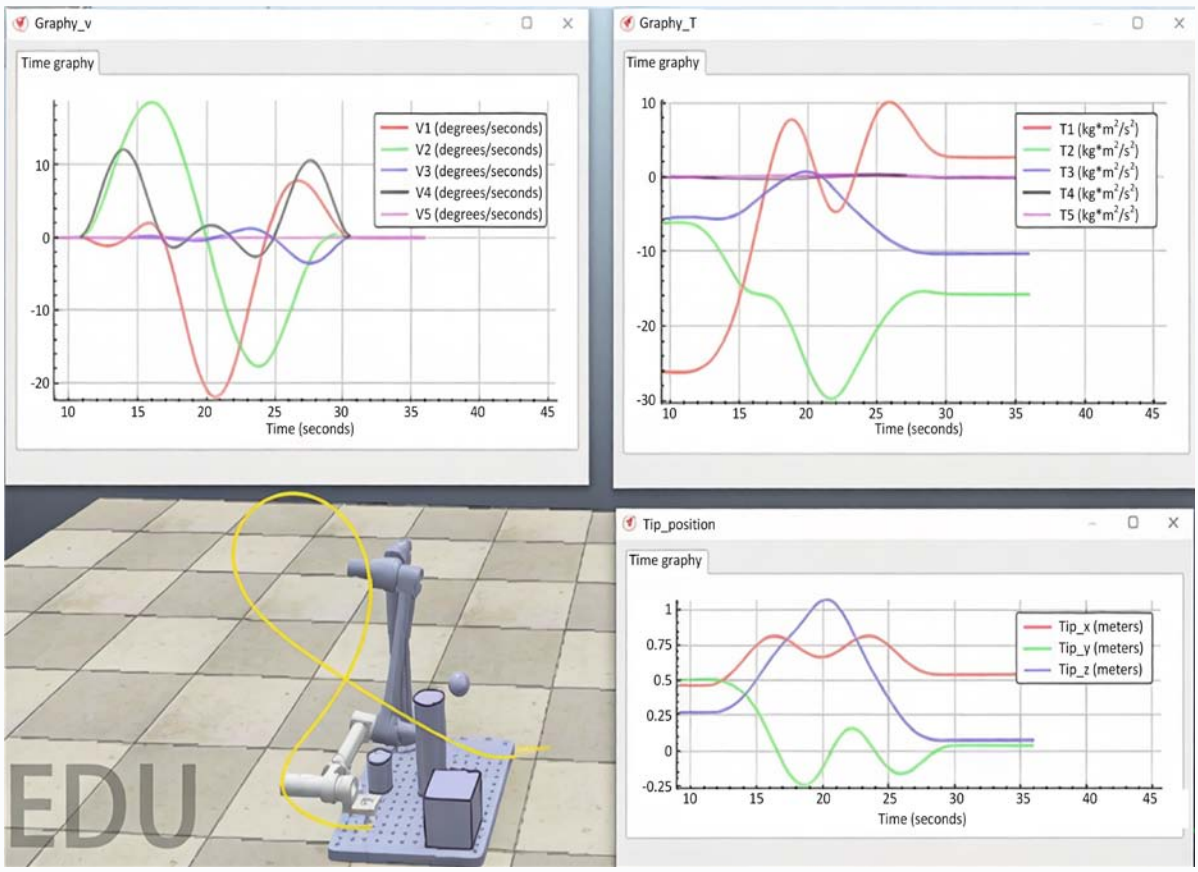

Figure 19: Experiment 4 simulation.

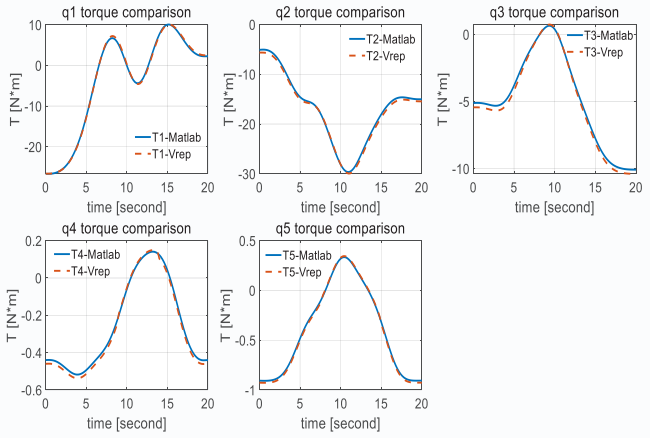

Figure 20: Experiment 4 torque comparison.

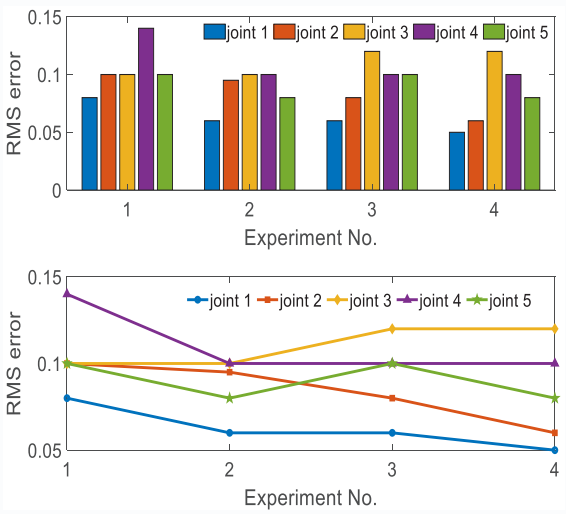

Figure 21: RMS error for each joint and trend of variation with experiment.

avoidance algorithm.

Conclusion

In this paper, a two-stage obstacle avoidance search algorithm is proposed based on the spatial grid method in the joint space of the robotic arm, with the collision of the first three joints as the main search basis and the collision ratio of the last two joints as the secondary discriminative condition. Moreover, considering the overall time-consuming nature of the algorithm search, the joint configurations and operation space are divided in advance according to the corresponding spacing, and the time-consuming database establishment process is placed in the offline stage, so that the actual obstacle avoidance route search process only needs to be based on the occupancy of different obstacles to be extracted from the entire offline database. For the discrete path points found, five non-uniform B-splines are fitted and the node intervals are redistributed using the NSGA algorithm to optimal the impact and energy consumption of the whole trajectory. Simulation verification is carried out using

CoppeliaSim simulation, and the results prove that the algorithm proposed in this paper can successfully achieve obstacle avoidance.

References

1. Cheng Z, Wang Y, Sun H, et al. Research status of robotic arm technology and its typical application analysis. Agricultural Machinery Using & Maintenance, 2021; (6): 19–22.
2. Guo Y, Lai G. Review of Joint Space Trajectory Planning and Optimization for Industrial Robot. Journal of Mechanical Transmission, 2020; 44(2): 154–165.
3. Long Z, Li X, Shuai T, et al. Review of Research State of Trajectory Planning for Industrial Robots. Mechanical Science and Technology for Aerospace Engineering, 2021; 40(7): 853–862.
4. Yin F, Liang Q, Cheng X, et al. Research on Mechanical Arm Joint Space Trajectory Planning Algorithm Based on Optimal Time. Machine Design & Research, 2017; 33(8): 12–15.
5. Peter H, Nils N, Bertram R. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 1968; 4(2): 100–107.
6. Karima B. Adaptive Dijkstra's Search Algorithm for MIMO detection. International Journal of Electrical and Computer Engineering Systems, 2022; 13(1): 9–17.
7. Knights V A, Gacovski Z, Deskovski S. Obstacles Avoidance Algorithm for Mobile Robots, Using the Potential Fields Method. Universal Journal of Electrical and Electronic Engineering, 2017; 5(4).
8. Gang L, Wang J. PRM path planning optimization algorithm research. WSEAS Transactions on Systems and Control, 2015; 10: 148–153.
9. Vojtech V, Jan F, Tomas K, et al. RRT-path – A Guided Rapidly Exploring Random Tree. Robot Motion and Control, 2009; 396: 307–316.
10. Ouyan J, He C, Long H, et al. A Review of Obstacle Avoidance Path Planning Algorithms for Robotic Arms. Journal of Guangdong Polytechnic Normal, 2024; 45(9): 22–32.
11. Wu Z, Meng Z, Zhao W, et al. Fast-RRT: A RRT-Based Optimal Path Finding Method. Applied Sciences, 2021; 11(24): 11777–11777.
12. Jingu K, Dongwoo L, Yongsik C, et al. Improved RRT-Connect Algorithm Based on Triangular Inequality for Robot Path Planning. Sensors, 2021; 21(2): 333–333.
13. Yang H, Jia Q, Zhang W. An Environmental Potential Field Based RRT Algorithm for UAV Path Planning. 37th Chinese Control Conference (CCC). 2018: 9922–9927.
14. Cheng Q, Zhang W, Liu H, et al. Research on the Path Planning Algorithm of a Manipulator Based on GMM/GMR-MPRM. Applied Sciences-Basel, 2021; 11(16).
15. Guo H, Liu Z, Huang P, et al. Robotic arm path planning based on improved RRT-Connect algorithm. Modular Machine Tool & Automatic Manufacturing Technique, 2025; (30): 41–45.
16. Klemm S, Oberländer J, Hermann A, et al. RRT*-Connect: Faster, asymptotically optimal motion planning. IEEE International Conference on Robotics and Biomimetics (ROBIO). 2015: 1670–1677.
17. Ma Y, Liang Y. A robot arm obstacle avoidance algorithm based on sixth degree polynomial trajectory planning. Journal of Northwestern Polytechnical University, 2020; 38(35): 392–400.
18. Qiang Z, Chen B, Liu X, et al. Optimal path planning for mobile robots based on improved potential field ant colony algorithm. Transactions of the Chinese Society for Agricultural Machinery, 2019; 50(36): 23–32+42.
19. Zhang J, Zhang Q, Wei X. Obstacle Avoidance Path Planning of Space Robot Based on Improved Particle Swarm Optimization. Symmetry, 2022; 14(5): 938–938.
20. Arup K. S, Amit K, Tanuka B, et al. Synergism of Firefly Algorithm and Q-Learning for Robot Arm Path Planning. Swarm and Evolutionary Computation, 2018; 43: 50–68.
21. Gao Q, Yuan Q, Sun Y, et al. Path planning algorithm of robot arm based on improved RRT* and BP neural network algorithm. Journal of King Saud University – Computer and Information Sciences, 2023; 35(31): 1319–1578.
22. Ji W, Deng C, Ge J, et al. Progress in robot joint space trajectory planning research. Machine Design and Manufacturing Engineering, 2022; 51(60): 15–23.
23. Huang W, Du W, Yang S, et al. Improved polynomial curve-fitting trajectory prediction algorithm. Systems Engineering and Electronics, 2024; 46(01): 280–289.
24. Wang X, Song J, Zheng J, et al. A study of robot trajectory fitting with improved B-spline curves. Transducer and Microsystem Technologies, 2021; 40(02): 41–43.
25. Wu G, Zhang N. Kinematically Constrained Jerk–Continuous S-Curve Trajectory Planning in Joint Space for Industrial Robots. Electronics, 2023; 12(5): 1135–1135.
26. Xiao Y, Zhu Y, Li W, et al. A new high-order continuous point-to-point motion trajectory planning algorithm. Journal of Harbin Institute of Technology, 2021; 53(62): 135–143.
27. Liu X, Lin S, Ou Y, et al. Trajectory planning for a double quintuple polynomial transition manipulator. Machinery Design & Manufacture, 2014; (64): 40–43.
28. Marek B, Paweł K, Krzysztof G. The Use of Asymmetric Polynomial Profiles for Planning a Smooth Trajectory. Applied Sciences, 2022; 12(23): 12284–12284.
29. Dong J, Wang T, Dong D, et al. Improved B-spline curves applied to 6R robot trajectory optimization. China Mechanical Engineering, 2018; 29(63): 193–200.
30. Zhang F, Zhang S, Li W, et al. Industrial robot trajectory planning based on improved particle swarm algorithm. Control and Instruments in Chemical Industry, 2024; 51(66): 631–637+680.
31. Wu F, Yuan L, Wu D, et al. Improved NSGA-II algorithm for multi-objective trajectory optimization of robotic arm. Mechanical Science and Technology for Aerospace Engineering, 2024; 3(65): 1–12.
32. Kucuk S. Optimal trajectory generation algorithm for serial and parallel manipulators. Robotics and Computer Integrated Manufacturing, 2017; 48: 219–232.
33. Huang Y, Gu Y, Zheng Z. Research on the Path Planning of Hair-Insertion Robot Arm Based on Ant Colony Optimization. 37th Chinese Control Conference (CCC), July 25–27, Wuhan, China. 2018; 5191–5195.
34. Shi H, Jin D, Yu Y, et al. A study of some basic problems of six-degree-of-freedom robotic arms. Mechanical Management and Development, 2024; 39(37): 19–22+25.
35. Nhouchi A, Said S. B, Abdallah M. A. B, et al. A real-time A* algorithm for trajectories generation and collision avoidance in uncertain environments for assembly applications. Computers & Industrial Engineering, 2025; 202: 110959–110959.
36. Gasparetto A, Zanotto V. A new method for smooth trajectory planning of robot manipulators. Mechanism and Machine Theory, 2007; 42(4): 455–471.
37. Xu H, Gabriel A, Man Q, et al. A versatile robot simulation software V-REP. Mechanical Engineering & Automation, 2018; (80): 47–48.