applied sciences

MDPI

Article

Trajectory Planning and Singularity Avoidance Algorithm for Robotic Arm Obstacle Avoidance Based on an Improved Fast Marching Tree

Baoju Wu [1], Xiaohui Wu [2,*], Nanmu Hui [1] and Xiaowei Han [1]

[1] Institute of Scientific and Technological Innovation, Shenyang University, Shenyang 110000, China; wbj@syu.edu.cn (B.W.); huinm@syu.edu.cn (N.H.); hanxw@syu.edu.cn (X.H.)
[2] School of Information Engineering, Shenyang University, Shenyang 110000, China
* Correspondence: wuxiaohui9802@163.com

Abstract: The quest for efficient and safe trajectory planning in robotic manipulation poses significant challenges, particularly in complex obstacle environments where the risk of encountering singularities and obstacles is high. Addressing this critical issue, our study presents a novel enhancement of the Fast Marching Tree (FMT) algorithm, ingeniously designed to navigate the complex terrain of Cartesian space with an unprecedented level of finesse. At the heart of our approach lies a sophisticated two-stage path point sampling strategy, ingeniously coupled with a singularity avoidance mechanism that leverages geometric perception to assess and mitigate the risk of encountering problematic configurations. This innovative method not only facilitates seamless obstacle navigation but also adeptly circumvents the perilous zones of singularity, ensuring a smooth and uninterrupted path for the robotic arm. To further refine the trajectory, we incorporate a quasi-uniform cubic *B*-spline curve, optimizing the path for both efficiency and smoothness. Our comprehensive simulation experiments underscore the superiority of our algorithm, showcasing its ability to consistently achieve shorter, more efficient paths while steadfastly avoiding obstacles and singularities. The practical applicability of our method is further corroborated through successful implementation in real-world robotic arm trajectory planning scenarios, highlighting its potential to revolutionize the field with its robustness and adaptability.

Keywords: robotic arm trajectory planning; singularity avoidance; FMT path searching algorithm; singularity metric

Citation: Wu, B.; Wu, X.; Hui, N.; Han, X. Trajectory Planning and Singularity Avoidance Algorithm for Robotic Arm Obstacle Avoidance Based on an Improved Fast Marching Tree. *Appl. Sci.* **2024**, *14*, 3241. https://doi.org/10.3390/app14083241

Academic Editors: Giovanni Boschetti, Matteo Bottin and Riccardo Minto

Received: 1 March 2024
Revised: 5 April 2024
Accepted: 8 April 2024
Published: 11 April 2024

1. Introduction

With the increasing use of robotic arms in various industrial applications, robotic arm trajectory planning has become a crucial topic of research. Trajectory planning for robotic arms can be performed either in Cartesian space or joint space. Compared to trajectory planning in joint space, Cartesian space trajectory planning offers more intuitive and feasible solutions with better motion repeatability [1]. Cartesian space trajectory planning is particularly useful for applications that require specific end-effector trajectories, such as welding and operations in confined spaces [2]. However, one of the challenges in Cartesian space trajectory planning is the presence of singularities, which are not encountered in trajectory planning in joint space [3].

Several approaches have been developed for obstacle avoidance planning for robotic arms in Cartesian space. Classic sampling-based methods include Rapidly Exploring Random Tree Star (RRT*) and the probabilistic roadmap method (PRM), and many algorithms have been developed based on these two algorithms. Janson et al. combined the respective advantages of RRT* and PRM algorithms and proposed a Fast Marching Tree (FMT) [4]. Gammell et al. introduced the Batch Informed Trees (BIT*) algorithm and applied it to dual-arm robot trajectory planning [5]. Rybus et al. used an improved version of BI-RRT*

for planning Cartesian trajectories of robotic arms [6]. Chen et al. proposed BI-FMT* and applied it to trajectory planning for spatial robotic arms [7]. Optimization-based methods have also been utilized for trajectory planning. For example, Jin et al. used Chaos Particle Swarm Optimization (CPSO) combined with damping least squares (DLS) and feedback compensation to avoid singularity [8]. Zhao et al. proposed a dynamic damping least squares method based on the minimum singular value to adjust Cartesian space trajectories in singular regions [9]. Riboli et al. used the least squares method and optimized the trajectory using *B*-splines [10]. Ju et al. introduced a model-based approach for robotic arm trajectory planning. They developed a predictive obstacle avoidance model that takes into account the geometric configuration of the arm. By considering motion state cost, frontal collision cost, and approach time cost, they formulated a cost function. This method also accounts for the deformation of the model when approaching singular points [11]. Learning-based approaches have also been explored. Fujii et al. used a novel neural network for gap inference to accelerate the generation of robotic arm motion trajectories [12]. Liu et al. proposed a model-free supervisor executor deep reinforcement learning method for path planning of UR5 robotic arms [13].

In the above discussion, several methods for obstacle avoidance planning of robotic arms in Cartesian space were discussed. Among them, the sampling-based method is simple, intuitive, and easy to understand and implement. Due to the randomness of the sampling, it also performs well in complex spaces. With a moderate number of sampling points, the computational complexity is not high. The optimization-based method requires setting appropriate optimization objectives. It consumes a lot of computation in complex spaces and is prone to becoming trapped in local optima. The model-based method relies on mathematical models established based on prior knowledge, and the accuracy of the modeling has an impact on the results. The learning-based method requires a large amount of data for training and has poor interpretability. Therefore, the sampling-based method has the advantages of speed, robustness, and wide applicability. As for the treatment of singularities in robotic arms, the mentioned methods do not explicitly identify or avoid singular points. Instead, they handle the singular regions by applying certain treatments to the Jacobian matrix to ensure smooth passage through the singular points.

Processing the Jacobian matrix in a certain way can allow the robotic arm to work in the singular region, such as through methods like damping least squares. However, these methods have issues with error accumulation and multiple solutions [14]. Therefore, when it is not necessary to pass through the singular region, it is better to select a reasonable trajectory that avoids singular points. Liu Y et al. decomposed the Jacobian matrix and eliminated singular points by determining whether the determinant of the Jacobian matrix is zero, ensuring that the trajectory solutions obtained through genetic algorithms were feasible [15]. Lu L et al. used the condition number index as an optimization indicator and proposed a high-order joint smooth trajectory planning method for singularity avoidance [16]. Beck F et al. proposed an artificial potential field method based on known singular configurations, incorporating manipulability indicators as optimization criteria for singularity avoidance [17]. Haviland J et al. proposed a robotic arm control method that maximizes manipulability indicators [18]. Manavalan J et al. presented a robotic arm control strategy that autonomously learns task constraints and plans trajectories by maximizing manipulability indicators [19]. Hao J analyzed the singular configuration of robot trajectories using dexterity indicators and applied them to robotic arm path planning [20]. Cao B et al. constructed the workspace of a robotic arm using flexibility indicators and applied it to grasping tasks [21]. To better utilize the geometric shape of the manipulability ellipsoid, Petrovi ć L et al. used Riemann distance to measure the proximity between a given configuration and a singular configuration, proposing geometry-aware singularity avoidance costs [22,23]. These methods primarily utilize optimization-based trajectory planning approaches for singularity avoidance without combining their optimization objectives with sample-based path planning methods. Moreover, these methods focus on improving the motion performance of the robotic arm and rarely discuss the issues of

obstacle avoidance and singularity avoidance simultaneously. Table 1 shows the methods used and problems addressed by all the scholars mentioned above.

Table 1. Problems addressed and methods used by different scholars in literature reviews.

Reference	Authors	Problem Addressed	Methods Used
[4]	Janson et al.	Merging RRT* and PRM for obstacle avoidance planning.	FMT*
[5]	Gammell.	Enhancing FMT* for dual manipulator obstacle avoidance.	BIT*
[6]	Rybus.	Applying bidirectional RRT algorithm to plan collision-free trajectory of robotic arms.	BI-RRT*
[7]	Chen et al.	Apply bidirectional FMT* to handle spacecraft power failure and arm deployment.	BI-FMT*
[8]	Jin et al.	Applying CPSO to solve the problem of singular points in the trajectory of robotic arms.	CPSO with DLS
[9]	Zhao et al.	Adjusting robotic arm trajectories based on minimum singular values.	Dynamic DLS
[10]	Riboli et al.	Trajectory planning of gantry system using *B*-spline and least squares method.	Least squares with *B*-splines
[11]	Ju et al.	Triangular collision planes and cost functions for obstacle avoidance planning.	Geometric-based predictable obstacle avoidance
[12]	Fujii et al.	Real-time trajectory smoothing via shortcutting for manipulators.	Learning-based approaches
[13]	Liu et al.	Model-free deep learning for UR5 robotic arm path planning.	Deep reinforcement learning
[15]	Liu Y et al.	Genetic algorithm with singularity avoidance for trajectory optimization.	AEGA-SA
[16]	Lu L et al.	Differential vector optimization for smooth, singularity-free trajectory planning.	High-order joint smooth trajectory planning method
[17]	Beck F et al.	Custom potential functions for singular configuration avoidance.	Improved artificial potential field method
[18]	Haviland J et al.	Learning-based control strategy for robotic arm task constraints.	A purely reactive method for maximizing manipulability
[19]	Manavalan J et al.	Planning with maximized manipulability without known constraints.	Learning-based methods combine manipulability index
[20]	Hao J	Addressing singularities with dexterity index and trajectory replanning.	Optimization methods using dexterity index
[21]	Cao B	Optimizing humanoid robot vision and operation through torso positioning.	Trajectory planning method combined with a dexterity map
[22,23]	Petrovi ć L et al.	Introducing geometric perception singularity index for trajectory planning.	Geometry-aware Singularity Avoidance costs

In response to the aforementioned issues, this paper integrates the geometry-aware singularity avoidance cost with the FMT* algorithm to generate a path plan for obstacle and singularity avoidance in the Cartesian space of the robotic arm. To ensure a more suitable quantity and distribution of sampling points in space, a two-stage sampling point generation strategy is proposed. To avoid interference between obstacles and the robotic arm's joints, a geometric envelope method is used for collision detection. To achieve smoother generated paths, a quasi-uniform cubic *B*-spline curve is employed for path optimization. These improvements enable the proposed algorithm to simultaneously achieve good performance and acceptable computational complexity. Through simulation experiments and real-world robotic arm verification, the proposed algorithm demonstrates shorter path lengths and stable obstacle avoidance capabilities compared to other methods. These experimental results showcase the superiority and potential of the proposed algorithm in robotic arm obstacle avoidance planning.

2. Methods

The trajectory planning and singularity avoidance algorithm for robotic arm obstacle avoidance based on an improved FMT is shown in Figure 1. The system receives inputs

of obstacle information in Cartesian space, along with the start and end coordinates of the robotic arm end effector. First, path points are sampled in the robotic arm workspace environment. After initial sampling, initial path generation, and secondary sampling, the sampling phase ends. The next phase is the path expansion phase, where the path points obtained from secondary sampling are connected. Starting from the root node, which is the start point of the robotic arm end effector, an improved cost function that includes singularity avoidance is used to expand the nodes. At each expansion, the path point with the minimum cost within the connection radius is searched. When there are no unconnected path points in the space, the path optimization phase is entered. Each complete path is checked for collisions to prevent collisions between the robotic arm links. Then, the cost of collision-free paths is calculated, and the path with the minimum cost is obtained. Finally, the trajectory is smoothed using *B*-spline curves to achieve a smoother motion.

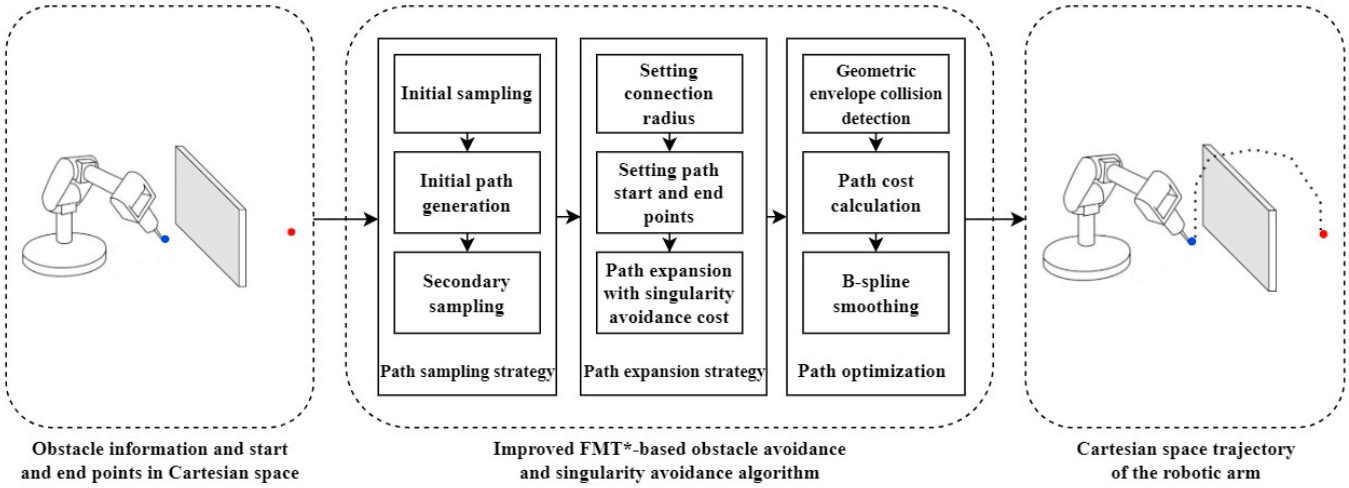


Figure 1. The workflow of improving the FMT algorithm is shown in the figure, where the blue dots represent the starting point of the robotic arm path planning, and the red dots represent the endpoint of the path planning.

The improved FMT algorithm is presented as shown in Algorithm 1. The inputs to the algorithm are the sampling space X_{free}, path search starting point X_{init}, path search goal point x_{goal}, number of sample points n, initial path expansion radius $R1$, and secondary path expansion radius $R2$. Initially, the standard FMT is used to generate a feasible path set π. Subsequently, the feasible path set is utilized to obtain a secondary sampling path point collection V. Afterward, the collection is used for secondary path expansion to obtain a candidate path set Π. Finally, the paths in the set are filtered and smoothed to obtain the optimal path.

Algorithm 1 Improved FMT*

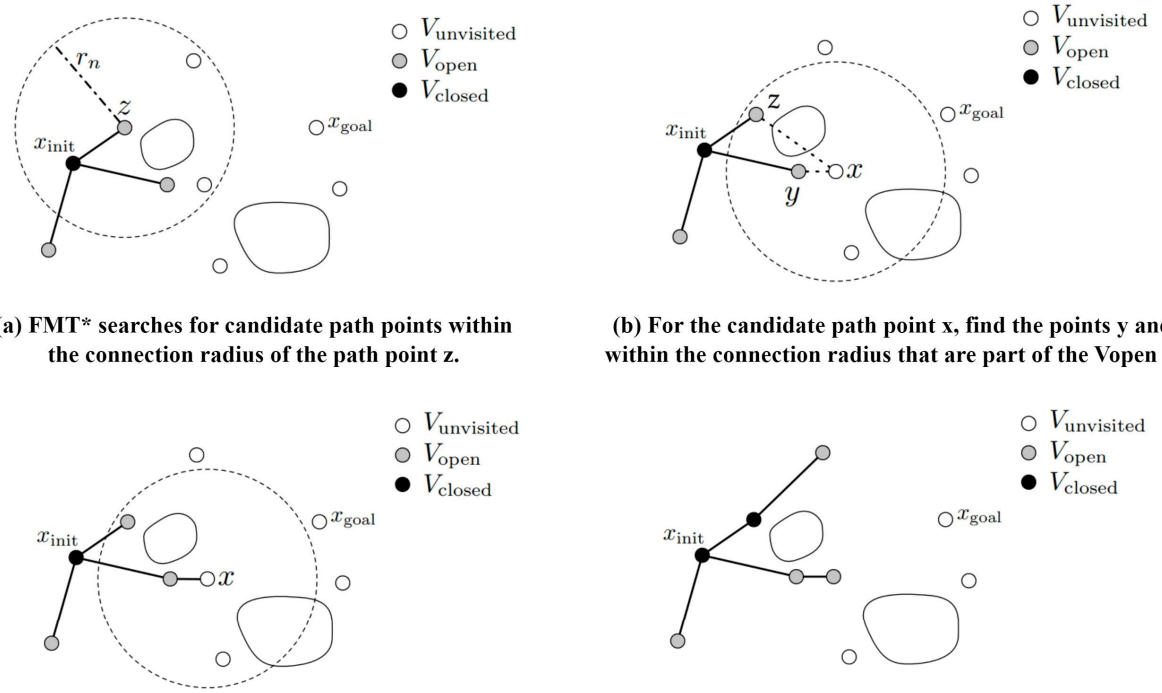
Data: X_{free}, X_{init}, X_{goal}, n, $R1$, $R2$
Result: *BestPath*
1: $\pi \leftarrow$ StandardFMT*(X_{free}, x_{init}, x_{goal}, n, $R1$)
2: $V \leftarrow$ SecondarySampling(X_{free}, π, n, *width*)
3: $\Pi \leftarrow$ SecondaryExpansion(V, x_{init}, x_{goal}, $R2$)
4: *BestPath* $\leftarrow$ PathOptimization(Π)

2.1. Principles of the FMT* Algorithm

The FMT algorithm searches for the optimal path by constructing a search tree. Firstly, a certain number of path points are randomly sampled from the global exploration area. Three sets are created to classify the path points: the $V_{unvisited}$ set consists of path points that have not been added to the tree, the V_{open} set consists of path points that have been

connected to the tree and need to be expanded, and the V_{close} set consists of path points that have been added to the tree and cannot be expanded.

One iteration of the algorithm is depicted in Figure 2. After the initial point x_{init} is expanded in the first round, it is moved from the V_{open} set to the V_{close} set. In the subsequent steps, a point z is selected from the V_{open} set. Within its connection radius, candidate path points (e.g., point x) are identified. All points in the V_{open} set that fall within the connection radius of the candidate path point x are then connected. The algorithm calculates the cost of each candidate connection and selects the minimum cost collision-free path. The newly added point is then classified into the V_{open}, V_{close}, or $V_{unvisited}$ set, and the iteration is completed.



(a) FMT* searches for candidate path points within the connection radius of the path point z.

(b) For the candidate path point x, find the points y and z within the connection radius that are part of the Vopen set.

(c) Calculate the collision-free path with the minimum cost within a radius of candidate path points x, and connect it.

(d) After the completion of connecting the newly added point, it is divided into three point sets, and this iteration is concluded.

Figure 2. Working principle of the FMT* algorithm.

FMT* enhances the limitations of RRT* and PRM* by combining their features. Like RRT*, FMT* utilizes a search tree for path exploration. However, instead of simply sampling and connecting a path point to the nearest node, FMT* incorporates a cost calculation process to ensure higher-quality paths. In contrast, both FMT* and PRM* involve sampling a certain number of paths in space. However, PRM* requires constructing a complete path graph and searching for the optimal path within it, which demands substantial computational resources.

During the process of connecting path points, the FMT* algorithm relies on a loss function to evaluate the quality of the path and selects the path point with the minimum loss to expand the search tree. This process adopts the idea of dynamic programming and follows the following state transition equation:

$$c(x) = \min\{c(y) + \mathrm{Cost}(y, x)\} \text{ where } \|y - x\| < r_n \tag{1}$$

In Equation (1), $y \in V_{opend}$, where y is a point that has been connected on the search tree, $x \in V_{unvisited}$, where x is a candidate path point within a radius r_n of y. $c(x)$ represents the lowest cost from the point x to the root node, and $c(y)$ represents the cost from node y to the root node. $Cost(y, x)$ is the cost from node y to node x, and the standard FMT*

algorithm uses the Euclidean distance from a node to the root node as the cost function for path expansion.

The pseudocode for the standard FMT* is listed in Algorithm 2. The initialization phase of the standard FMT* starts by resetting the flag (lines 2–3), followed by generating a set of points using random sampling in the sampling space X_{free} (line 4). Subsequently, a search tree T is initialized with the start point and the set of points (line 5). Set X_{init} as the initial frontier point z (line 6). During the path expansion phase of the standard FMT*, if the conditions for continuing path expansion are met, the Expand function is used to iteratively generate the search tree in the search space, with the initial expansion connection radius set to R1, and only using path length as its loss function. If enough feasible paths are found, the path search is deemed successful, and the search tree T and the path set π are returned as results; otherwise, an empty set is returned and the program terminates (lines 7–15).

Algorithm 2 Standard Fast Marching Tree (FMT*)

1: **function** STANDARD FMT*(X_{free}, x_{init}, x_{goal}, n, $R1$)
2: *success* = *FALSE*
3: *stop_expansion* = *FALSE*
4: $S \leftarrow x_{init} \cup x_{goal} \cup$ PSEUDO RANDOM SAMPLING(X_{free}, n)
5: $T \leftarrow$ INITIALIZE(S, x_{init})
6: $z \rightarrow x_{init}$
7: **while** (*stop_expansion* = *FALSE* and *success* = *FALSE*) **do**
8: $\{T, z, success, stop\_expansion\} \leftarrow$ EXPAND(X_{free}, x_{init}, x_{goal}, $R1$)
9: **if** (*success* = *TRUE*) **then**
10: $\pi \leftarrow$ PATH(x_{init}, x_{goal}, T)
11: **else**
12: $\pi \rightarrow \emptyset$
13: **end if**
14: **end while**
15: **return** π, T
16: **end function**

2.2. Improved Path Point Sampling Strategy

Although FMT* has made improvements compared to RRT* and PRM*, there are still opportunities for optimization in certain aspects. For example, FMT* uses a random sampling of path points in the sampling space, resulting in paths that are heavily influenced by randomness. Additionally, the fixed connection radius leads to lower path search efficiency. To address these issues, this paper proposes an improved path point sampling strategy that is performed in two stages.

2.2.1. Initial Sampling

The original FMT* algorithm utilizes pseudo-random sampling to sample a certain number of path points in the search space. Sobol sampling, on the other hand, is a deterministic low-discrepancy sampling method. Compared to random sampling, Sobol sampling can better fill the search space, reducing overlapping and redundant samples, as shown in Figure 3. Therefore, Sobol sampling finds wide applications in numerical integration, signal processing, computer graphics, and other fields.

The Sobol sequence can be represented as the product of the Van der Corput sequence and the Sobol-generating matrix C. For a Sobol sequence X_n with n numbers, it can be expressed as follows:

$$X_{\mathrm{n}} = [(\varphi_2, C(0)), (\varphi_2, C(1)) \ldots, (\varphi_2, C(n))] \tag{2}$$

where

$$\varphi_b, C(i) = (b^{-1} \ldots, b^{-M})[C(a_0(i) \ldots, a_{M-1}(i))^T] \tag{3}$$

In the Equation (2), φ_b represents the Van der Corput sequence in base b. The variable i represents the i-th number in the Van der Corput sequence (i = 0, 1, 2. . .n). In Equation (3), $a_M(i)$ represents the value of the M-th digit of the i-th number in the Van der Corput

sequence in base b. The Sobol-generating matrix C is calculated based on the generating polynomial and the direction numbers. For Sobol sampling in different dimensions, the generating matrix C is different. For example, a 2-dimensional Sobol sampling requires a $2 \times k$ generating matrix, where k represents the number of direction numbers. The calculation of the generating matrix is complex and will not be further discussed here. Joe S. et al. have found the highest-dimensional Sobol-generating matrix of dimension 21,201, which can be referenced for interested readers [24,25]. In practical applications, pre-calculated generating matrices are used, eliminating the need for complex calculations and resulting in higher sampling efficiency.

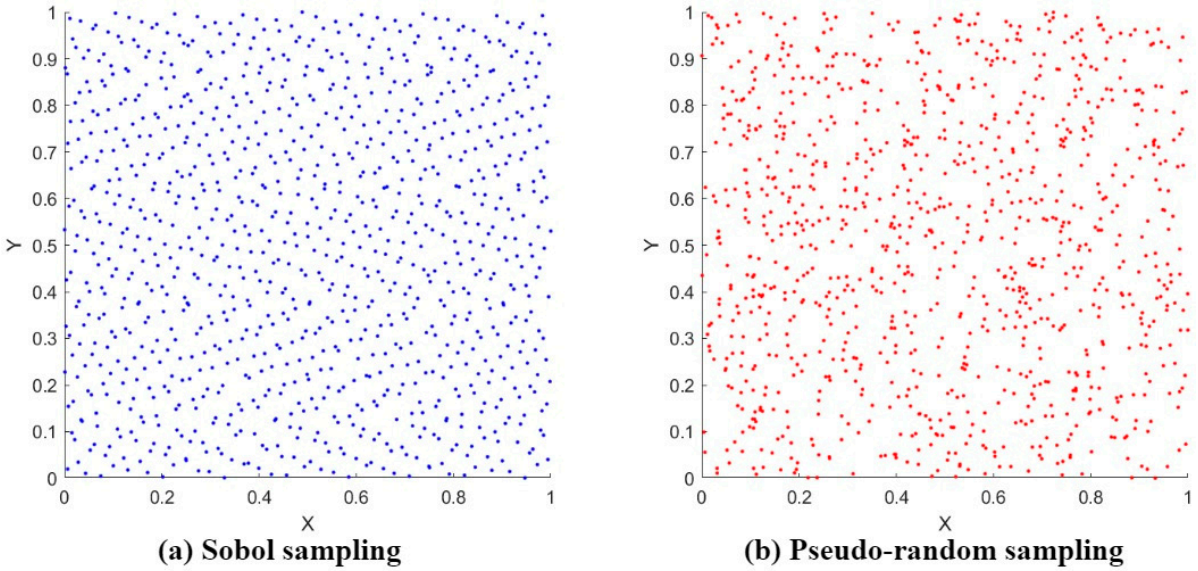


Figure 3. Comparison of Sobol sampling and pseudo-random sampling.

Using Sobol sampling in the search space for initial sampling, followed by setting a larger connection radius r_n, the original FMT* algorithm is used to generate the initial feasible path. Sobol sampling ensures a more uniform distribution of sampling points, while using a larger connection radius allows for faster expansion of the feasible paths. As a result, collision-free paths from the start to the goal can be quickly generated.

2.2.2. Secondary Sampling

After completing the initial sampling, a certain number of feasible paths are obtained. These paths serve as reference paths for the secondary sampling. The initial sampling points are cleared, and then secondary sampling is performed within a certain range of the reference paths, as shown in Figure 4. The process of secondary sampling can be represented by pseudocode.

The pseudocode for secondary path point sampling is shown in Algorithm 3. The process starts by segmenting and sorting the path set, followed by clearing other path points in the sampling space (lines 2–5). The width of all path segments is then expanded (line 7), and the secondary sampling space is defined as the intersection of the path space and sampling space (line 8). Sobol sampling is conducted in the secondary sampling space (line 9). For each sample, it is necessary to determine whether the sampling point falls within an obstacle; if it does, the point is deleted; otherwise, it is retained and added to the point set V. Finally, set V is returned as the result of secondary sampling (lines 10–17).

Algorithm 3 Secondary Sampling

function SECONDARY SAMPLING(X_{free}, π, n, *width*)
  path←INITAL PATH SEGMENTATION(π)//Segmenting the path formed by standard FMT*
  path_num←PATH SORT(π) //sort the path segments and return the total number
  CLEAR(X_{free}) //clearing initial sampling points
  $i = 0$
  while ($i <$ *path_num*) **do**
    X_{Path}←RESIZE PATH WIDTH(*width*)//resize the width of the initial sampling paths
    $X_{Path} \cap X_{free} = X_{Sample}$
    Secondary_Point←SOBOL SAMPLING(X_{Sample}, *point_num*)
    if OUTSIDE OBSTACLE(*SecondaryPoint*) == *TRUE* **then**
      DELETE(*SecondaryPoint*)
    else
      V←APPEND(*SecondaryPoint*)
    end if
    $i = i + 1$
  end while
return V
end function

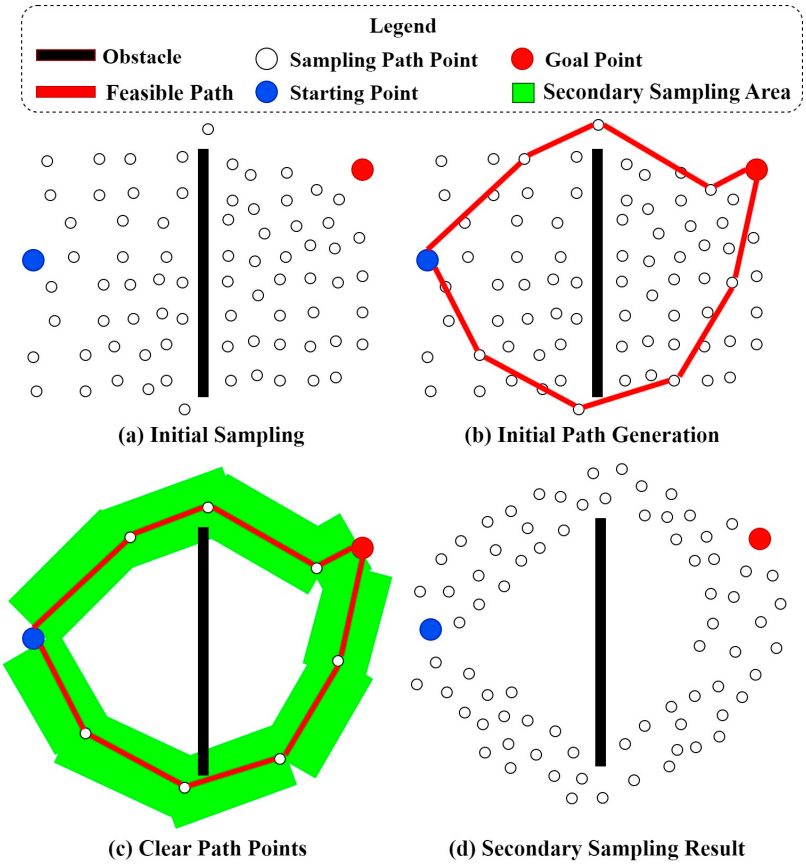


Figure 4. Secondary sampling diagram.

From Figure 4, it can be seen that the secondary sampling path points are distributed around the initial feasible path, which effectively improves the quality of path searching.

2.3. Path Connection

The proposal adopts a two-stage approach for path point sampling and path connection. In the initial sampling stage, Sobol sampling is used throughout the sampling space. After the initial sampling, the Euclidean distance is used as the cost for the initial path connection. In the secondary sampling stage, the sampling is performed around the paths connected in the initial stage, and the secondary path connection uses both Euclidean

distance and a geometric perceptual singularity metric as costs. Different connection radii are set for the two stages of path connection.

The pseudocode for the secondary path expansion is presented in Algorithm 4. The algorithm begins with the initialization phase, where all points in set V except X_{init} are placed into the unvisited set $V_{unvisited}$ (line 2), X_{init} is placed into V_{open} (line 3), and V_{close} is initialized as an empty set (line 4). The initial frontier point z is set to X_{init} (line 5), and points within the connection radius $R2$ of the frontier point z are noted as N_z (line 6). While the current frontier point z is not the goal point x_{goal}, for a point x in N_z, the set N_x represents the points within the connection radius R2 (lines 7–9). The points in N_x that belong to V_{open} are noted as N_y (line 10), and then the point y_{min} in N_y with the minimum cost to expand is found (line 11). If the attempted connection path between y_{min} and x is collision-free (line 12), the connection between y_{min} and x is added to the tree T (line 13), and the cumulative cost $c(x)$ is updated based on the minimum cost from y_{min} to x (line 16). If node x successfully connects, it is moved to the open list V_{open}, while node y is moved into V_{close} (lines 19–20). If V_{open} is empty, indicating there are no more points available for expansion, the stop_expansion flag is returned (line 22). Finally, the frontier point z is changed in preparation for the next expansion (line 23). After completing all loops, the path set Π, search tree T, and the success flag are returned.

Algorithm 4 Secondary Expansion

1: **function** SECONDARY EXPANSION(V, x_{init}, x_{goal}, $R2$)
2: $V_{unvisited} \leftarrow V \backslash \{x_{init}\}$
3: $V_{open} \leftarrow \{x_{init}\}$
4: $V_{dosed} \leftarrow \emptyset$
5: $z \leftarrow x_{init}$
6: $N_z \leftarrow Near(V \backslash \{z\}, z, R2)$
7: **while** ($z \neq x_{goal}$) **do**
8: **for** $x \in N_z$ **do**
9: $N_x \leftarrow Near(V \backslash \{x\})$
10: $N_y \leftarrow N_x \cap V_{open}$
11: $y_{min} \leftarrow argmin_{y \in Ny} \{c(y) + Cost(y, x)\}$
12: **if** $CollisionFree(y_{min}, x)$ **then**
13: $T \leftarrow T \cup \{(y_{min}, x)\}$
14: $V_{open,new} \leftarrow V_{open,new} \cup \{x\}$
15: $V_{unvisited} \leftarrow V_{unvisited} \backslash \{x\}$
16: $c(x) = c(y_{min}) + cost(y_{min}, x) + sigcost(x)$
17: **end if**
18: **end for**
19: $V_{open} \leftarrow \{V_{open} \cup V_{open,new}\} \backslash \{z\}$
20: $V_{closed} \leftarrow V_{closed} \cup \{z\}$
21: **if** $V_{open} = \emptyset$ **then**
22: **return** *stop_expansion*
23: **end if**
24: $z \leftarrow argmin_{y \in Vopen} \{z\}$
25: **end while**
26 : **return** Π, T, *success*
27: **end function**

2.3.1. Two-Stage Connection Parameter Settings

In the path connection process of FMT, the connection radius r_n is a constant that remains unchanged throughout, which limits the flexibility of path exploration. For instance, when the distance between the start and goal points is large, a small connection radius would require a significant number of iterations. Conversely, if the distance between the start and goal points is small, using a larger connection radius would result in overly linear optimal paths. Figure 5 depicts the search tree and optimal paths generated by FMT with different connection radii.

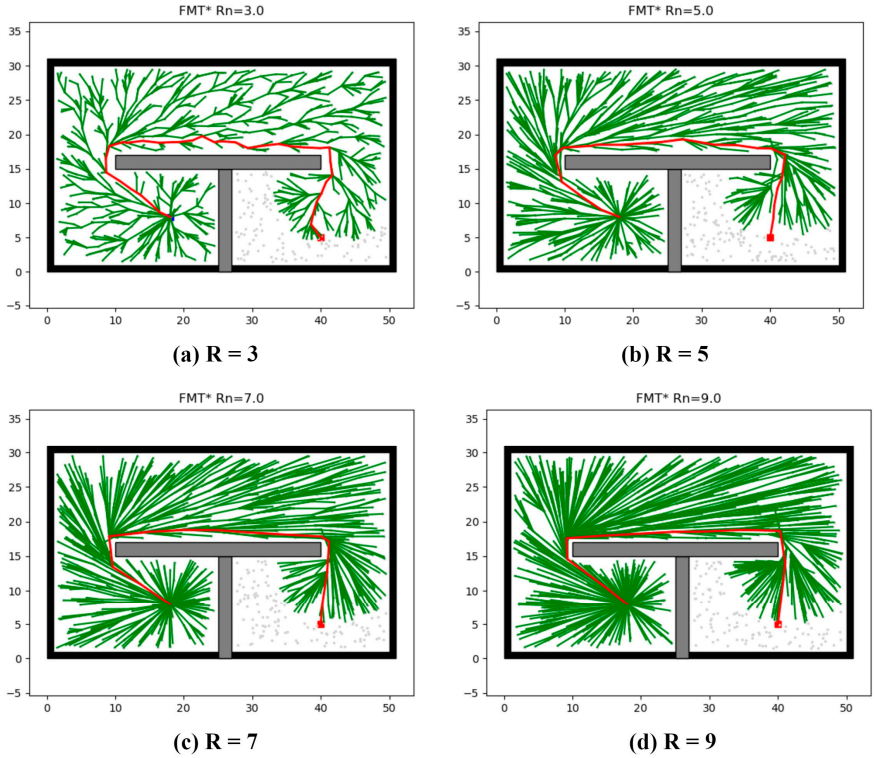


Figure 5. Comparison of search trees and optimal paths with different connection radii, and the red path is the optimal path found.

To enhance the flexibility and efficiency of path expansion, different connection radii are set for each stage using Equation (4):

$$r_n = \frac{\lambda V_{space}}{N_{sample}} \tag{4}$$

N_{sample} represents the number of sampled points, V_{space} represents the volume of the sampling space, and λ represents the connection coefficient. After the initial sampling, in order to find more feasible paths from the start to the goal point within a smaller number of iterations, λ is set to 5. This allows for larger connection step sizes in the search tree, resulting in fewer iterations needed to find feasible paths. After the secondary sampling, a set of path points is formed around the feasible paths. To explore more path points in the search tree, λ is set to 2.5. This restricts the search range for each step, leading to increased iterations, but it also brings diversity to the path search.

2.3.2. Avoiding Singularities Cost

The original FMT* algorithm only considers the Euclidean distance as the cost for path expansion, resulting in the generation of the shortest path. However, for trajectory planning of robotic arms in Cartesian space, avoiding singularities is an important consideration. Therefore, the singularity index can be incorporated into the cost function of the FMT* algorithm.

Defining the joint space Q and the operational space X (in Cartesian space), the following relationship holds:

$$\mathrm{f} : Q \rightarrow X \tag{5}$$

By using the forward kinematics matrix f, the joint space Q of the robotic arm can be mapped to the operational space X.

$$J(q) = \frac{\partial f}{\partial q} \tag{6}$$

The Jacobian matrix $J(q)$ is the partial derivative of the forward kinematics matrix f concerning the joint angle vector q, determined by the configuration of the robot's joints. The singularity of the robotic arm can be determined by the determinant of the Jacobian matrix. If the determinant of the Jacobian matrix is zero, i.e., $J(q) = 0$, it indicates that the robotic arm is in a singular configuration.

According to the theory of singular value decomposition of matrices, the Jacobian matrix of any joint configuration can be decomposed into singular values.

$$J(q) = U \sum V \tag{7}$$

where

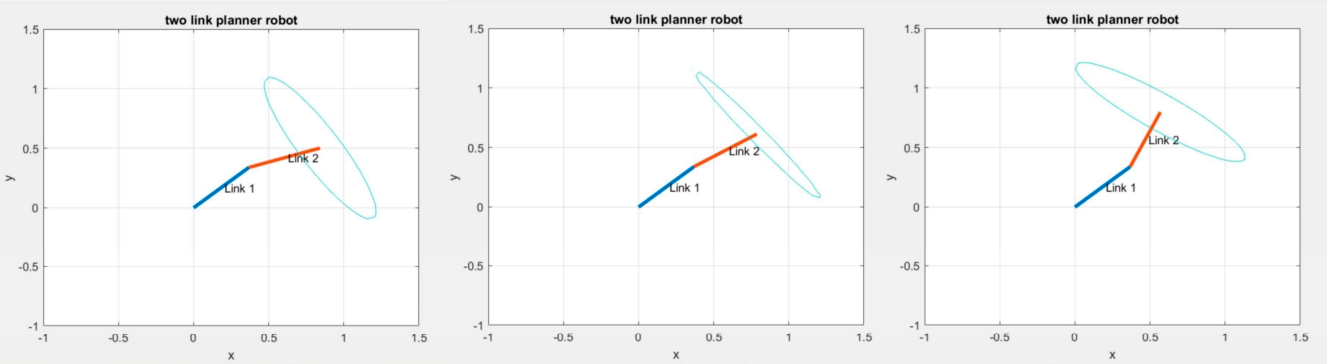
$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_m & 0 \end{bmatrix} \tag{8}$$

$U \in \Re^{m \times m}$ and $V \in \Re^{n \times n}$ are both orthogonal matrices, Σ is a diagonal matrix with singular values of the Jacobian matrix $J(q)$ as its elements, and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \geq 0$.

While the Jacobian determinant can qualitatively determine singularity, many singularity indices use the singular values of the Jacobian matrix to quantitatively measure the distance between the robotic arm and the singular configuration.

The condition number $K(J)$ is defined as the ratio between the maximum singular value σ_1 and the minimum singular value σ_m of the Jacobian matrix $J(q)$. The dexterity index $D(J)$ is derived from the condition number index and is related to the minimum condition number of the intermediate joint angles $q_i (i = 2, 3, 4, 5)$. The manipulability index $W(J)$ is the determinant of the product of the Jacobian matrix and its transpose. When the manipulability index $W(J) = 0$, the robotic arm is in a singular region, and the larger the value of $W(J)$, the further the distance from the singularity region.

The aforementioned singularity indices are all related to the manipulability ellipsoid. The manipulability ellipsoid can be described by the singular value matrix, and the principal axes of the manipulability ellipsoid are determined by the singular values σ_i of the Jacobian matrix. Figure 6 shows the manipulability ellipsoid of a two-link robotic arm under different joint configurations.



(a) The manipulability ellipsoid before passing through the singularity point

(b) The manipulability ellipsoid when passing through the singularity point

(c) The manipulability ellipsoid after passing through the singularity point

Figure 6. Manipulability ellipsoids under different joint configurations.

From Figure 6, it can be observed that as the two-link robotic arm moves closer to the singularity point, the area of the manipulability ellipsoid becomes smaller and flatter in shape. Conversely, when the arm moves further away from the singularity point, the area of the manipulability ellipsoid becomes larger and approaches a more spherical shape. The shape of the manipulability ellipsoid is related to the singular values of the Jacobian matrix; thus, the singularity of the robotic arm can also be determined using the shape of

the manipulability ellipsoid. Recently, geometrically aware singularity indices have been proposed [22,23].

For a trajectory in joint space, the geometrically aware singularity cost can be represented using Equation (9):

$$C[q(t)] = \sum_{i=0}^{n} d\left(k, J(q_i)J^T(q_i)\right)^2 = \sum_{i=0}^{n} \left\| \log\left(k^{-\frac{1}{4}} J(q_i)J^T(q_i)\right) \right\|_F^2 \tag{9}$$

$C[q(t)]$ represents the singularity cost function of the trajectory $q(t)$. q_i ($i = 0, 1 \ldots n$) represents each planning point on the trajectory. $J(q_i)$ represents the Jacobian matrix at the planning point q_i. $d(A, B)$ represents the Riemann distance between matrices A and B. $\|\ldots\|_F$ represents the Frobenius norm. k is a diagonal matrix with elements σ_k as given values, satisfying the following equation:

$$k = \begin{bmatrix} {\sigma_k}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & {\sigma_k}^2 \end{bmatrix} \text{ where } \sigma_k \geq \sigma_{\max} \tag{10}$$

Equation (9) uses the square of the manipulability index multiplied by a scaling factor and takes the square of the Frobenius norm as the cost. The essence of this equation is to use the Riemann distance between the matrix k and the matrix $J(q_i)J^T(q_i)$ as the cost. The matrix k represents the ideal case where all singular values of the Jacobian matrix of the robotic arm are greater than the maximum value σ_{max}. In this case, the manipulability ellipsoid is in its most ideal shape. On the other hand, the matrix $J(q_i)J^T(q_i)$ represents the manipulability ellipsoid at the planning point q_i. Therefore, the higher the Riemann distance between the manipulability ellipsoid at q_i and the ideal manipulability ellipsoid, the higher the singularity cost.

For trajectory planning in Cartesian space, the trajectory can be transformed to joint space using inverse kinematics f^{-1}:

$$q(t) = f^{-1}(x(t)) \tag{11}$$

Thus, the equation for the geometrically aware singularity cost in Cartesian space is given by Equation (12):

$$C[x(t)] = \sum_{i=0}^{n} d\left(k, J\left(f^{-1}(x_i)\right)J^T\left(f^{-1}(x_i)\right)\right)^2 = \sum_{i=0}^{n} \left\| \log\left(k^{-\frac{1}{4}} J\left(f^{-1}(x_i)\right)J^T\left(f^{-1}(x_i)\right)\right) \right\|_F^2 \tag{12}$$

where $x(t)$ represents the end effector trajectory in Cartesian space, and x_i represents the i-th end effector pose on the trajectory. When $|J| = 0$, it is not possible to compute the inverse kinematics solution, so $C[x(t)]$ is set to infinity.

Finally, the singularity index mentioned above is incorporated into the cost function of FMT* path planning, forming a new cost function $c(x)^*$ with singularity avoidance, defined as Equation (13):

$$c(x)^* = \min\{c(y) + \mathrm{Cost}(y, x) + sig\cos t(x)\} \tag{13}$$

where

$$sig\cos t(x) = \left\| \log\left(k^{-\frac{1}{4}} J\left(f^{-1}(x_x)\right)J^T\left(f^{-1}(x_x)\right)\right) \right\|_F^2 \tag{14}$$

x_x represents the end effector pose of the robotic arm at path point x. Compared to the original cost function, which only used the Euclidean distance from the path point x to the root node as the cost, the improved cost function also added the singularity index of Riemannian geometric perception as the cost function, so it can avoid the singularity of the robotic arm.

2.4. Path Optimization

To obtain the optimal path from the end effector trajectories generated by the improved FMT* algorithm, the following steps are taken. Firstly, the geometric envelope method is employed to check for interference between the robotic arm links and obstacles. Trajectories that result in collisions are removed. Then, the cost of each remaining path is calculated, and the path with the lowest cost is selected. Finally, the chosen path is smoothed using the *B*-spline algorithm.

2.4.1. Collision Detection

In terms of collision detection, the end effector trajectories are first input into a simulation environment built based on the real working environment of the robotic arm. The simulated robotic arm moves along the planned path, and the interference between each link and obstacle is detected. This process can generally be carried out using a geometric envelope method. As shown in Figure 7, the robotic arm is wrapped with simple geometric shapes, and collision detection is performed using envelope models. The advantage of this method is that it requires less computational effort for collision detection, as it only needs to calculate the minimum distance between the envelope model and the obstacles to determine if a collision occurs. If a collision is detected at a certain point along the trajectory, the inverse kinematics of the robotic arm at that point can be traversed to check if all joint combinations lead to collisions. If a collision-free inverse solution cannot be found, the trajectory is discarded.

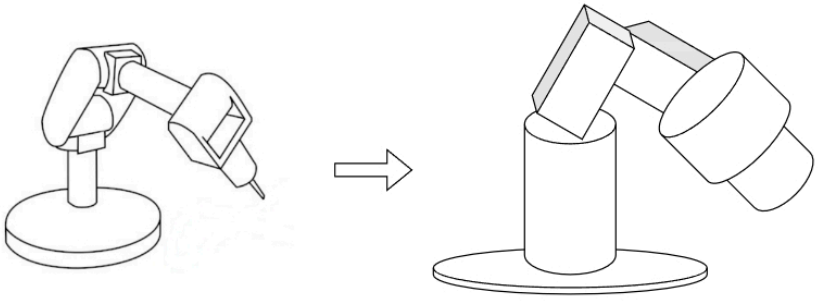


Figure 7. Geometric envelope method illustration.

2.4.2. Path Cost Calculation

After completing the collision detection, the cost of all collision-free trajectories is calculated. Since the cost of each path point has already been calculated during the expansion of the path points, the total path cost is the sum of the costs of all path points on that path, as shown in Equation (15):

$$C_{path} = \sum_{i=0}^{n} c(x_i)^* \tag{15}$$

$c(x_i)^*$ represents the cost of the i-th path point in the improved FMT* algorithm. This cost is determined by the distance from the path point to the root node and the distance from the path point to the singularity region. Using this equation, the path with the minimum cost can be obtained.

2.4.3. Path Smoothing

To make the final path smoother, *B*-spline curves are used to make the robotic arm trajectory smoother. The *B*-spline curve can be represented using the Equation (16):

$$p(u) = \sum_{i=0}^{n} P_i B_{i,k}(u) \tag{16}$$

where $P_i(i \in [0, n])$ represents the i-th control point of the B-spline curve, $B_{i,k}(u)$ represents the i-th k-th order B-spline basis function. The basis function can be defined using the de Boor–Cox recursion formula:

$$B_{i,k}(u) = \begin{cases} \begin{cases} 1, \ u_i \le u \le u_{i+1} \\ 0, \ \text{otherwise} \end{cases} & k = 0 \\ \frac{u-u_i}{u_{i+k}-u_i} B_{i,k-1}(u) + \frac{u_{i+k+1}-u}{u_{i+k+1}-u_{i+1}} B_{i+1,k-1}(u) & k \ge 1 \end{cases} \tag{17}$$

In Equation (17), if both the numerator and denominator are zero, then the term is assumed to be 0. If the denominator is zero and the numerator is not zero, then the term is assumed to be 1. $u_i(i\epsilon[0, n+k])$ is called the knot vector, which is defined as Equation (18):

$$[u_0, u_1, \cdots, u_k, u_{k+1}, \cdots, u_n, u_{n+1}, \cdots, u_{n+k}] \tag{18}$$

Based on the knot vector, n + k segments of the B-spline curve can be generated. The knot vector can control the shape of the B-spline curve and its local characteristics near the control points. Commonly used knot vectors include uniform knot vectors and quasi-uniform knot vectors. A uniform knot vector is a vector with equal intervals between the nodes, which can create a uniformly distributed curve. A quasi-uniform knot vector has duplicate elements at both ends of the knot vector, which can make the tangent of the spline curve at the beginning and end parallel to the line connecting the control points.

In this paper, the path points generated by the improved FMT* algorithm are used as control points, and quasi-uniform cubic B-spline curves are used as the curves for trajectory planning to make the planned trajectory smoother.

2.5. Computational Complexity

2.5.1. Time Complexity Analysis

The improved FMT algorithm proposed in this article first uses standard FMT* to generate feasible paths. For the standard FMT*, suppose the number of sampling points is n, and if the connection radius is set to the maximum distance between sample points, the time complexity is $O(n \log(n))$. Then, in the second sampling step, it is necessary to segment and sort the feasible paths, and expand the width of the feasible paths. The time complexity of this step is linearly related to the number of path points, so its time complexity can be seen as $O(n)$. In the subsequent collision detection phase, each path point needs to be filtered using a cycle, so its time complexity is also $O(n)$. For the secondary sampling path expansion, compared with the path expansion of the original FMT* algorithm, the improved FMT* uses a singularity avoidance cost function based on geometric perception, which needs to calculate the inverse kinematics solution of the inner point of the connection radius in each iteration cycle, so its time complexity is $O(n^2)$. Therefore, the total time complexity of improving FMT is as Equation (19):

$$O(n \log(n)) + O(n) + O(n) + O(n^2) \approx O(n^2) \tag{19}$$

2.5.2. Space Complexity Analysis

At the beginning of the algorithm, it needs to be in free space X_{free}. Generate a set of sample points internally. Each sample point needs to store its coordinates in space, which is usually proportional to the dimension (d) of the problem. Therefore, the space required to store all sample points is $O(nd)$. However, when analyzing the overall spatial complexity, it is common to consider (d) as a constant, thus simplifying the spatial complexity of sample point storage $O(n)$. The FMT algorithm constructs a path by checking the potential connections between sample points, which requires storing connection information between point pairs. In the worst-case scenario, if each point is connected to all other points, the number of connections may be close to $O(n^2)$. However, due to the fact that the FMT algorithm uses a radius-based neighborhood search, not all pairs of points will form a connection, which determines whether points can be connected. In fact, as the number of

sample points increases, the average number of connections at each point is influenced by the choice of connection radius, but it does not increase in a square relationship with the number of points. The spatial complexity of improving FMT* satisfies Equation (20):

$$O(n) < O < O(n^2) \tag{20}$$

3. Experiment and Result

This paper first verifies the obstacle avoidance trajectory planning performance and singularity avoidance performance of the improved FMT* algorithm through simulation experiments on a personal computer. Then, the algorithm proposed in this paper is applied to the obstacle avoidance trajectory planning of the AUBO-I5 robotic arm.

3.1. Obstacle Avoidance Trajectory Planning Simulation Experiment

In order to evaluate the obstacle avoidance trajectory planning performance of the improved FMT* algorithm in Cartesian space, a three-dimensional path planning simulation environment was built on a computer based on the actual scenario. The length, width, and height of this rectangular space were set to 20 m, 20 m, and 5 m, respectively. A green point was set as the starting point and a red point as the destination. The coordinates of the starting point and destination were (10, 8, 1) and (12, 14, 1), respectively. There was a wall-shaped obstacle with a height of 3 m between the starting point and the destination. In the same simulation environment, the algorithm proposed in this paper was compared with three other path-planning algorithms in a comparative experiment. The results of the path planning are shown in Figure 8, where the green line segments represent the sampled paths and the red line segments represent the optimal path. From Figure 8, it can be seen that the path generated by the algorithm in this paper has a shorter length compared to the other algorithms, and the sampled paths are more concentrated.

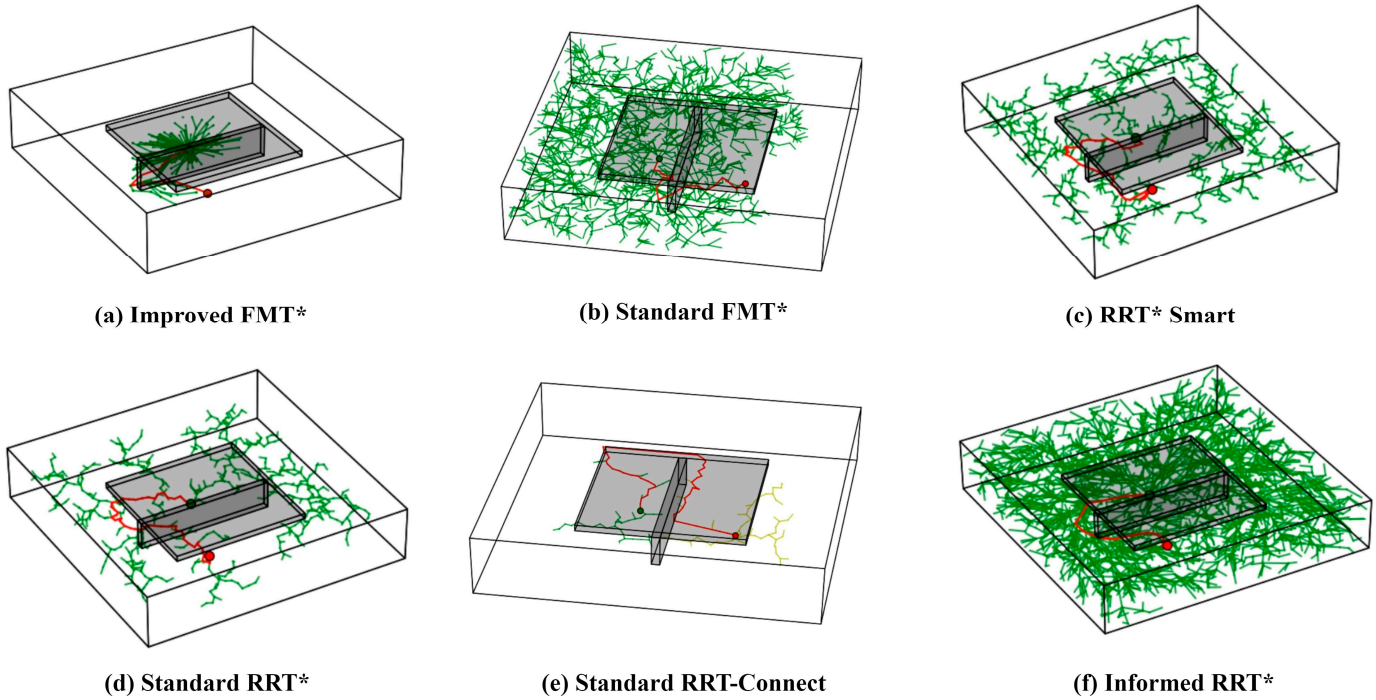


Figure 8. Comparison of different path search algorithms with the red path representing the optimal path found by the algorithm.

By conducting 20 experiments using each algorithm in the same environment, the experimental results shown in Table 2 were obtained. From the experimental results, it can be seen that the average path length of the improved FMT* algorithm is 18.8 m, which is 13.3% shorter than the standard FMT*, 8.3% shorter than the standard RRT*, and

22.3% shorter than the standard RRT-connect. This indicates that the path planned by the improved FMT* algorithm has a shorter length compared to the other three algorithms. Moreover, in terms of the variance of path length, the improved FMT* algorithm has better stability, as the path length does not vary significantly in each planning iteration, demonstrating good stability. In terms of the time consumed for path planning, the improved FMT* algorithm ranks second among the four algorithms. It does not have an advantage over the standard FMT* and standard RRT-connect, which may be due to the increased computational requirements of the algorithm. However, the increase in computation time compared to the standard FMT* is only 5.6%, which is acceptable.

Table 2. Experimental results of obstacle avoidance path planning under different algorithms.

Algorithm	Improved FMT*	Standard FMT*	RRT* Smart	Standard RRT*	Standard RRT-Connect	Informed RRT*
Parameters	Initial sample point: 1000 Secondary samp point: 500 Initial connection radius: 0.3 (m) Secondary connection radius: 0.15 (m)	Sample point: 1000 Connection radius: 0.3 (m)	Sample point: 1000 Connection radius: 0.3 (m)	Step length: 0.3 (m) Maximum iteration count: 4000	Step length: 0.3 (m) Maximum iteration count: 4000	Sample point: 1000 Connection radius: 0.3 (m)
Average time consumption	1.67 (s)	1.58 (s)	1.72 (s)	2.21 (s)	0.98 (s)	1.64 (s)
Time variance	0.16 (s^2)	0.11 (s^2)	0.13 (s)	0.38 (s^2)	0.09 (s^2)	0.21 (s^2)
Average path length	18.8 (m)	21.7 (m)	22.1 (m)	23.6 (m)	25.2 (m)	20.8 (m)
Path length variance	1.12 (m^2)	1.20 (m^2)	1.18 (m^2)	1.14 (m^2)	1.39 (m^2)	1.15 (m^2)

3.2. Singular Point Avoidance Simulation Experiment

To validate the effectiveness of the improved FMT* algorithm in avoiding singular points, experiments were conducted using a 6-degree-of-freedom (6-DOF) robotic arm model in MATLAB (version R2023b). Firstly, the initial and final poses were set in the workspace of the robotic arm, and a straight-line trajectory was planned. The start and end points were (0.5, 0.5, 1.2) and (−0.5, −0.5, 1.2), respectively. This trajectory passes through a singular point at the shoulder of the robotic arm, with coordinates (0.18, 0.18, 1.2). The MATLAB simulation system framework is shown in Figure 9.

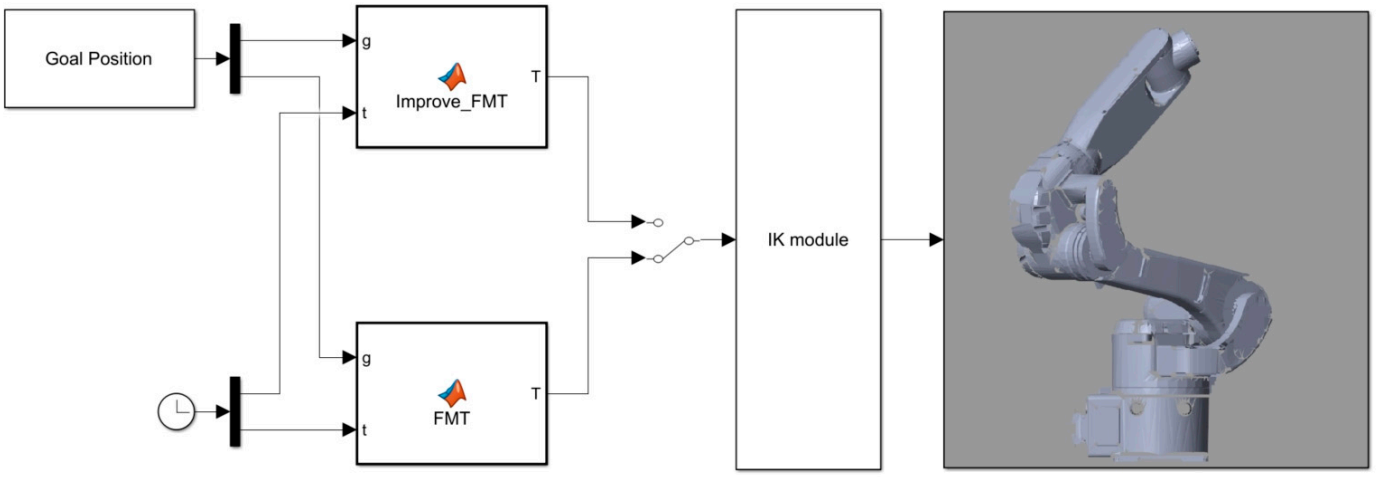
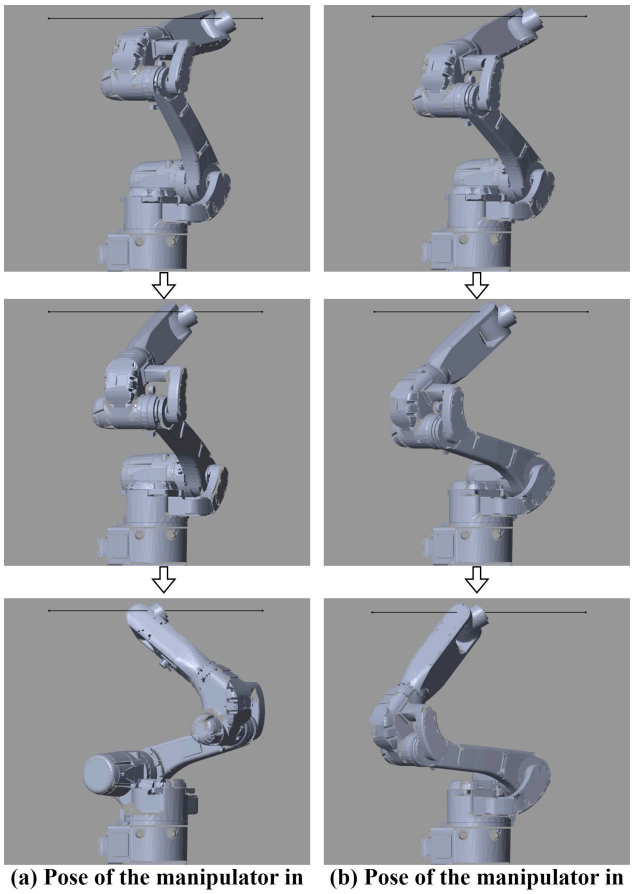


Figure 9. Simulation experiment Simulink system diagram.

As shown in Figure 10a, it can be observed that without singular point avoidance, joint 1, joint 4, joint 5, and joint 6 undergo significant changes in joint angles after passing through

the singular point. As illustrated in Figure 10b, by using the singular point avoidance trajectory planning algorithm proposed in this paper, the end effector of the robotic arm successfully avoids the singular point. This ensures that the joint configuration of the robotic arm remains closer to the initial configuration, avoiding drastic changes in joint angles and ensuring the stability and safety of the robotic arm's operation.



(a) Pose of the manipulator in Experimental Group 1 **(b) Pose of the manipulator in Experimental Group 2**

Figure 10. Comparison of joint configurations of the robotic arm with and without the application of the singular point avoidance algorithm.

Applying the proposed algorithm for singular point avoidance, the trajectory planning results for the given singular point are depicted in Figure 11. The red dashed line represents experimental group 1, which did not utilize the singular point avoidance algorithm, while the blue solid line represents experimental group 2, which employed the singular point avoidance algorithm. Since the cost function of this algorithm depends on the singularity and path length, the planned trajectory using the algorithm avoids the singular point and maintains the shortest path in the non-singular region.

The variation curves of joint angles and angular velocities for the robotic arm in the simulation experiment are shown in Figure 12. The red dashed line represents experimental group 1, while the blue solid line represents experimental group 2. In Figure 12a, the joint angle curves can be observed, and it is evident that the joint curves in experimental group 2 are smoother compared to those in experimental group 1, especially in joint 1. In experimental group 1, joint 1 rotates by 3.1 rad, which is equivalent to 177°, while in experimental group 2, joint 1 rotates by 0.3 rad, which is equivalent to 17.2°. In Figure 12b, the joint angular velocity curves are shown. In experimental group 1, significant angular velocity changes occur in joints 1, 4, 5, and 6, which can be attributed to passing through the shoulder singularity point. This result is consistent with theoretical analysis. In contrast, in

experimental group 2, the trajectory does not pass through the shoulder singularity point, resulting in overall smoother changes in joint angular velocities.

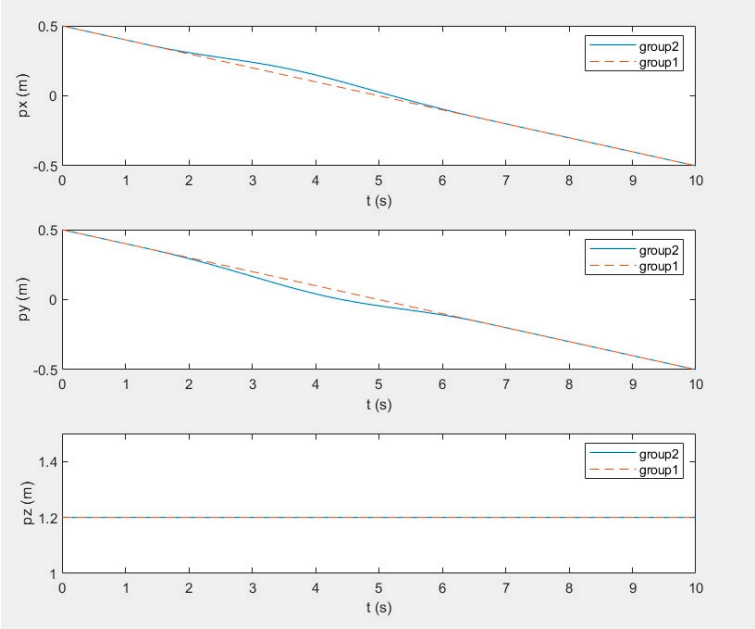
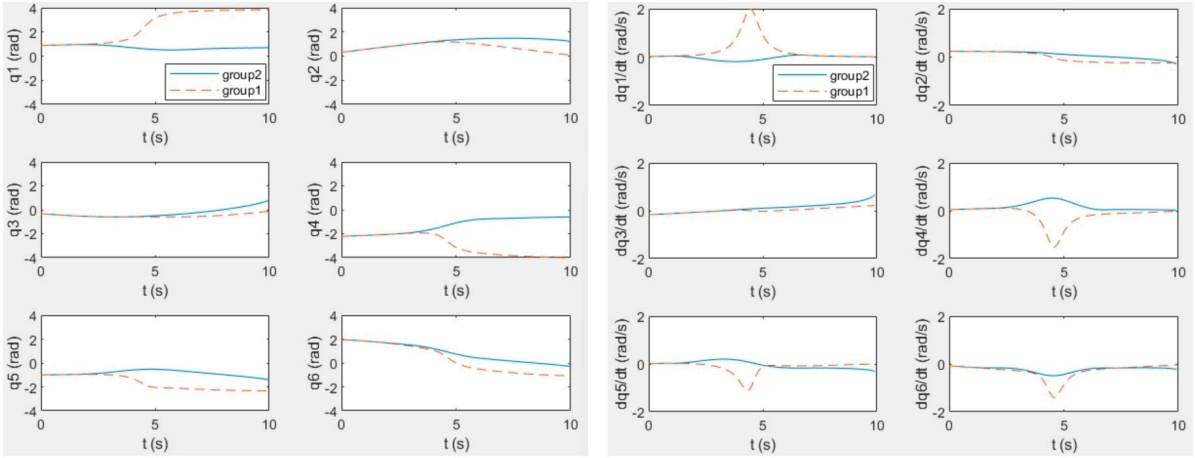


Figure 11. Comparison of trajectory planning results of the robotic arm with and without the application of the singular point avoidance algorithm.



(a)Joint Angle Variation Curve **(b)Joint Angular Velocity Variation Curve**

Figure 12. Variation of joint angles and angular velocities of the robotic arm in simulation experiment.

3.3. Robotic Arm Trajectory Planning Experiment in the Real World

An improved FMT* trajectory planning algorithm was effectively implemented for the AUBO-I5 robotic arm. As for the hardware, a personal computer running served as the primary controller for the robotic arm, while the AUBO-I5 robotic arm and its associated control cabinet acted as the execution mechanism. On the software side, a C++ program utilizing the Moveit interface was developed to incorporate the enhanced FMT* algorithm, with visualization performed through RVIZ. The program was developed and executed within the ROS Melodic environment, specifically on Ubuntu 18.04.6 LTS, and was written adhering to the C++11 standard.

As shown in Figure 13, obstacles were created in the working environment of the robotic arm, and the initial and target poses of the robotic arm were set. Then, the improved FMT* algorithm was used to plan a collision-free trajectory from the start point to the

endpoint. Finally, the robotic arm's working environment and trajectory planning were visualized using RVIZ. Since the algorithm in this study also has singularity avoidance capabilities, the collision-free trajectory planner can avoid the singular points of the robotic arm, thus improving the stability of the robotic arm.

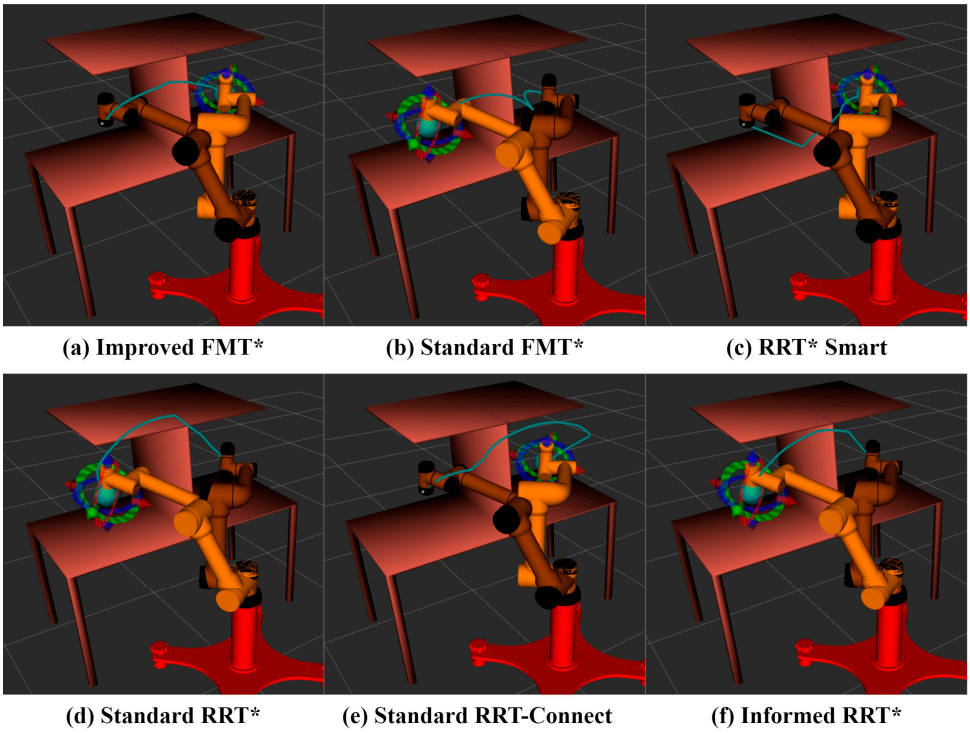


Figure 13. Visualization of the end-effector trajectory generated by the improved FMT* using RVIZ.

The generated robotic arm trajectory is converted into specific commands using Moveit and sent to the robotic arm control cabinet, thereby driving the robotic arm to follow the predetermined path. The actual motion of the robotic arm is illustrated in Figure 14.

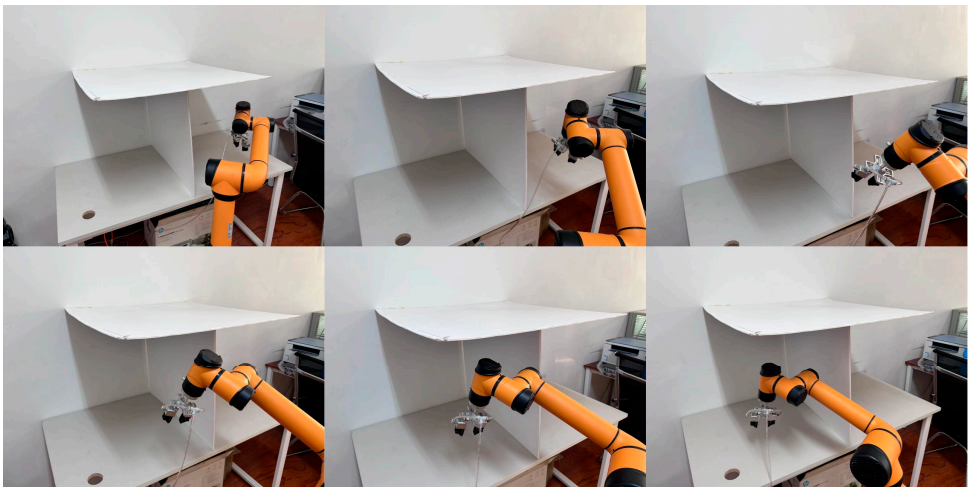


Figure 14. The robotic arm executes the trajectory planning generated on the PC.

4. Discussion

In the intricate field of path planning, the improved Fast Marching Tree (FMT*) algorithm introduced in our paper emerges as a notably efficient alternative to both optimization-based and learning-based methodologies. Its principal advantage lies in the elimination of the necessity for either a pre-constructed mathematical model, which demands extensive prior knowledge, or a data-intensive model that requires substantial train-

ing. This distinctive characteristic markedly reduces the computational resource footprint, enhancing the algorithm's ease of deployment and broadening its application spectrum.

When scrutinized alongside analogous sampling-based path-planning methods, specifically, the standard FMT, Rapidly Exploring Random Tree (RRT), and RRT-connect algorithms. Our improved FMT* algorithm demonstrates a clear superiority in generating more concise paths, as quantitatively evidenced in Table 1. This outcome signifies not only an optimization of the path's length but also an indirect contribution to the efficiency and safety of robotic operations. It is noteworthy, however, that the improved FMT* does not boast the fastest computation times among its peers. Yet, this slight concession in speed is counterbalanced by its unique capability for singularity avoidance, a critical feature absent in its counterparts that justifies the marginal increase in computational effort.

The algorithm's efficacy, particularly in singularity avoidance, was rigorously validated through simulation experiments employing a 6-degree-of-freedom robotic arm model in MATLAB. The results, illustrated in Figures 10 and 11, unequivocally demonstrate how our algorithm significantly elevates the robotic arm's motion performance by skillfully navigating away from singularities. Furthermore, real-world applicability tests conducted on the AUBO-I5 robotic arm, as depicted in Figures 12 and 13, confirmed the algorithm's practical effectiveness. The smooth avoidance of obstacles and singular regions, coupled with successful target point acquisition, attests to the robustness and real-world viability of the improved FMT$*$ algorithm.

This discussion affirms the improved FMT$*$ algorithm's status as a highly effective, computationally efficient solution for robotic arm trajectory planning, outperforming traditional models in both theoretical and practical arenas. Its ability to reduce path length while incorporating singularity avoidance into its computational framework represents a significant leap forward, paving the way for future advancements in robot arm trajectory planning.

5. Conclusions

In this study, we introduced an augmented version of the Fast Marching Tree (FMT*) algorithm, meticulously engineered for the sophisticated task of Cartesian space trajectory planning in robotic arms, with a special focus on evading both physical obstacles and the mathematical quagmires known as singularities. This enhancement is primarily anchored in the strategic application of Sobel sampling—a technique renowned for its capability to yield a uniform distribution of sampling points across the space. By integrating this technique into a novel two-stage sampling point generation strategy, we have significantly advanced the efficiency and effectiveness of trajectory planning in complex environments.

A pivotal innovation of our work is the refinement of the FMT* algorithm's path connection strategy through the incorporation of a geometric perception singularity index. This modification imbues the algorithm with the dual capability to skirt around obstacles while meticulously avoiding singular regions, ensuring a safer and more reliable path for robotic arm movement. The path selection process is equally robust, employing a comprehensive approach that leverages collision detection and path cost analysis to identify the most optimal trajectory. The application of quasi-uniform cubic B-splines further enhances the trajectory, ensuring smoothness and continuity essential for the precise execution of tasks by robotic arms.

Notwithstanding these advancements, our improved FMT* algorithm is not devoid of limitations. A notable challenge is the potential discrepancy between the optimal path in Cartesian space and the efficiency of that path when translated into joint space—a critical consideration for the practical application of robotic arms. Moreover, the reliance on sampling for path generation introduces a degree of variability influenced by initial sampling parameters. This sensitivity can result in significant variations in outcomes, emphasizing the impact of the stochastic nature of sampling-based methods on the predictability and consistency of the optimal path.

Future endeavors to ameliorate these limitations could explore adaptive sampling techniques that dynamically refine based on environmental complexity or investigate more sophisticated metrics for joint space efficiency. Additionally, integrating predictive models or machine learning algorithms to optimize initial sampling parameters may offer a pathway to reducing variability and enhancing the consistency of the algorithm's outcomes.

Author Contributions: Conceptualization, B.W.; Writing—Original Draft, X.W.; Writing—Review and Editing, N.H.; Funding acquisition X.H. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by The Liaoning Provincial Research Foundation for Applied Basic Research, China (No. 2023JH2/101300205).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: All authors would like to express their gratitude to everyone who contributed to this study for their insightful remarks.

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

References

1. Du, Z.-C.; Ouyang, G.-Y.; Xue, J.; Yao, Y.-B. A review on kinematic, workspace, trajectory planning and path planning of hyper-redundant manipulators. In Proceedings of the 2020 10th Institute of Electrical and Electronics Engineers International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), Xi'an, China, 10–13 October 2020; pp. 444–449. [CrossRef]
2. Park, C.; Rabe, F.; Sharma, S.; Scheurer, C.; Zimmermann, U.E.; Manocha, D. Parallel cartesian planning in dynamic environments using constrained trajectory planning. In Proceedings of the 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, Republic of Korea, 3–5 November 2015; pp. 983–990. [CrossRef]
3. Khan, A.T.; Li, S.; Kadry, S.; Nam, Y. Control framework for trajectory planning of soft manipulator using optimized RRT algorithm. *IEEE Access* **2020**, *8*, 171730–171743. [CrossRef]
4. Janson, L.; Schmerling, E.; Clark, A.; Pavone, M. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *Int. J. Robot. Res.* **2015**, *34*, 883–921. [CrossRef]
5. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 3067–3074. [CrossRef]
6. Rybus, T. Point-to-point motion planning of a free-floating space manipulator using the rapidly-exploring random trees (RRT) method. *Robotica* **2020**, *38*, 957–982. [CrossRef]
7. Chen, N.; Zhang, Y.; Cheng, W. Space detumbling robot arm deployment path planning based on Bi-FMT* algorithm. *Micromachines* **2021**, *12*, 1231. [CrossRef]
8. Jin, R.; Rocco, P.; Geng, Y. Cartesian trajectory planning of space robots using a multi-objective optimization. *Aerosp. Sci. Technol.* **2021**, *108*, 106360. [CrossRef]
9. Zhao, H.; Zhang, B.; Yin, X.; Zhang, Z.; Xia, Q.; Zhang, F. Singularity Analysis and Singularity Avoidance Trajectory Planning for Industrial Robots. In Proceedings of the 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021; pp. 6164–6169. [CrossRef]
10. Riboli, M.; Jaccard, M.; Silvestri, M.; Aimi, A.; Malara, C. Collision-free and smooth motion planning of dual-arm Cartesian robot based on *B*-spline representation. *Robot. Auton. Syst.* **2023**, *170*, 104534. [CrossRef]
11. Ju, F.; Jin, H.; Wang, B.; Zhao, J. A Predictable Obstacle Avoidance Model Based on Geometric Configuration of Redundant Manipulators for Motion Planning. *Sensors* **2023**, *23*, 4642. [CrossRef] [PubMed]
12. Fujii, S.; Pham, Q.-C. Realtime trajectory smoothing with neural nets. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 7248–7254. [CrossRef]
13. Liu, W.; Niu, H.; Mahyuddin, M.N.; Herrmann, G.; Carrasco, J. A model-free deep reinforcement learning approach for robotic manipulators path planning. In Proceedings of the 2021 21st International Conference on Control, Automation and Systems (ICCAS), Jeju Island, Republic of Korea, 12–15 October 2021; pp. 512–517. [CrossRef]
14. Dai, Y.; Xiang, C.; Zhang, Y.; Jiang, Y.; Qu, W.; Zhang, Q. A Review of spatial robotic arm trajectory planning. *Aerospace* **2022**, *9*, 361. [CrossRef]

15. Liu, Y.; Guo, C.; Weng, Y. Online time-optimal trajectory planning for robotic manipulators using adaptive elite genetic algorithm with singularity avoidance. *IEEE Access* **2019**, *7*, 146301–146308. [CrossRef]
16. Lu, L.; Zhang, L.; Fan, C.; Wang, H. High-order joint-smooth trajectory planning method considering tool-orientation constraints and singularity avoidance for robot surface machining. *J. Manuf. Process.* **2022**, *80*, 789–804. [CrossRef]
17. Beck, F.; Vu, M.N.; Hartl-Nesic, C.; Kugi, A. Singularity avoidance with application to online trajectory optimization for serial manipulators. *IFAC-Papers* **2023**, *56*, 284–291. [CrossRef]
18. Haviland, J.; Corke, P. A purely-reactive manipulability-maximising motion controller. *arXiv* **2020**, arXiv:2002.11901.
19. Manavalan, J.; Howard, M. Learning singularity avoidance. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 6849–6854. [CrossRef]
20. Hao, J.; Yang, W.-X.; Guo, Z.-D.; Cao, T.-T.; Chen, J.-H. Singularity Analysis of Scanning Trajectory and Avoidance Method for Ultrasonic Testing Robot. In Proceedings of the 2020 IEEE Far East NDT New Technology & Application Forum (FENDT), Kunming, China, 20–22 November 2020; pp. 199–203. [CrossRef]
21. Cao, B.; Sun, K.; Gu, Y.; Jin, M.; Liu, H. Humanoid Robot Torso Motion Planning Based on Manipulator Pose Dexterity Index. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *853*, 012040. [CrossRef]
22. Petrović, L.; Marić, F.; Marković, I.; Kelly, J.; Petrović, I. Trajectory optimization with geometry-aware singularity avoidance for robot motion planning. In Proceedings of the 2021 21st International Conference on Control, Automation and Systems (ICCAS), Jeju Island, Republic of Korea, 12–15 October 2021; pp. 1760–1765. [CrossRef]
23. Marić, F.; Petrović, L.; Guberina, M.; Kelly, J.; Petrović, I. A Riemannian metric for geometry-aware singularity avoidance by articulated robots. *Robot. Auton. Syst.* **2021**, *145*, 103865. [CrossRef]
24. Joe, S.; Kuo, F.Y. Remark on algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Trans. Math. Softw. (TOMS)* **2003**, *29*, 49–57. [CrossRef]
25. Joe, S.; Kuo, F.Y. Constructing Sobol sequences with better two-dimensional projections. *SIAM J. Sci. Comput.* **2008**, *30*, 2635–2654. [CrossRef]