

Article

Path Planning of a Mechanical Arm Based on an Improved Artificial Potential Field and a Rapid Expansion Random Tree Hybrid Algorithm

Qingni Yuan, Junhui Yi *, Ruitong Sun and Huan Bai

Key Laboratory of Modern Manufacturing Technology of Ministry of Education, Guizhou University, Guiyang 550025, China; qnyuan@gzu.edu.cn (Q.Y.); sunruitong2018@163.com (R.S.); baihuan2010@163.com (H.B.)

* Correspondence: junhuiyi163@163.com

Abstract: To improve the path planning efficiency of a robotic arm in three-dimensional space and improve the obstacle avoidance ability, this paper proposes an improved artificial potential field and rapid expansion random tree (APF-RRT) hybrid algorithm for the mechanical arm path planning method. The improved APF algorithm (I-APF) introduces a heuristic method based on the number of adjacent obstacles to escape from local minima, which solves the local minimum problem of the APF method and improves the search speed. The improved RRT algorithm (I-RRT) changes the selection method of the nearest neighbor node by introducing a triangular nearest neighbor node selection method, adopts an adaptive step and generates a virtual new node strategy to explore the path, and removes redundant path nodes generated by the RRT algorithm, which effectively improves the obstacle avoidance ability and efficiency of the algorithm. Bezier curves are used to fit the final generated path. Finally, an experimental analysis based on Python shows that the search time of the hybrid algorithm in a multi-obstacle environment is reduced to 2.8 s from 37.8 s (classic RRT algorithm), 10.1 s (RRT* algorithm), and 7.4 s (P_RRT* algorithm), and the success rate and efficiency of the search are both significantly improved. Furthermore, the hybrid algorithm is simulated in a robot operating system (ROS) using the UR5 mechanical arm, and the results prove the effectiveness and reliability of the hybrid algorithm.

Keywords: mechanical arm; path planning; artificial potential field method; rapid expansion random tree algorithm; virtual new node



Citation: Yuan, Q.; Yi, J.; Sun, R.; Bai, H. Path Planning of a Mechanical Arm Based on an Improved Artificial Potential Field and a Rapid Expansion Random Tree Hybrid Algorithm. *Algorithms* **2021**, *14*, 321. <https://doi.org/10.3390/a14110321>

Academic Editor: Rui Araújo

Received: 10 October 2021

Accepted: 31 October 2021

Published: 1 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, China's logistics industry has developed rapidly to meet the increasing demand for e-commerce in response to the internet economy and the rapid development of warehousing automation technology [1]. The intelligent robotic arm industry is developing rapidly. The robotic arm, as its name implies, is designed to imitate a human arm for moving, grasping, lifting, and loading objects, among other operations [2]. Therefore, robotic arms are widely used in the logistics and warehousing industry. To grasp a specified object and place it in the specified position, it is necessary to bypass complex obstacles and find an efficient and collision-free path, which is very simple for humans but presents many technical problems that need to be considered for the robotic arm. Thus, for the robotic arm, its path planning is one of the most important technical problems. Successful path planning can greatly improve the storage and grabbing efficiency and has long been a research hotspot in robotic applications.

2. Related Research

Among the existing path planning algorithms, the artificial potential field (APF) method for path planning was first proposed by Khatib [3] in 1986. The idea is to use

virtual force to make the robot navigate obstacles. The disadvantages of this algorithm are as follows: The algorithm easily falls into local minima or oscillates, and it is difficult to reach the target point when there are obstacles nearby. The rapid expansion random tree (RRT) algorithm was first proposed by the American professor LaValle [4] in 1998. The RRT method is a path planning algorithm based on sampling and an efficient multi-dimensional space with complete probability but is not optimal. However, the randomness of the RRT method causes it to be blind and exhibit a low efficiency; in addition, the resulting path is tortuous and not smooth enough, and the search speed is slow in a narrow area. A large number of scholars have made different improvements to these two algorithms. Zheng et al. [5] proposed a new minimum criterion and designed an improved virtual obstacle local path planning method to overcome the tendency of the APF algorithm to easily fall into local minima and other shortcomings. Sun et al. [6] proposed the use of dynamic windows to improve the APF method to solve the problem of being trapped in local minima. Zhang et al. [7] proposed a curved path planning algorithm for overtaking cars based on an improved APF method, and an optimal guaranteed performance control strategy for tracking the curved paths for overtaking cars based on linear robust control theory was proposed. Han et al. [8] proposed an improved APF method to solve the problems of large swinging trajectories and easily falling into local minima that are encountered by the classic APF method in unmanned aerial vehicle (UAV) trajectory planning. Zhang et al. [9] proposed a path planning method for multiple underwater unmanned vehicles (UUVs) in a three-dimensional environment based on the “domain”, which solves the disadvantages of unreachable targets near obstacles, local minima, and oscillations encountered in the classic APF method. Tian et al. [10] proposed an overall configuration planning method of continuum hyper-redundant manipulators (CHRM)s based on an improved APF method that avoids complicated inverse kinematics and vastly reduces the computational complexity. Li et al. [11] proposed a path planning method for mobile mechanical arms based on the sparse node RRT algorithm that solves the problem of excessively searching in the local space and reduces the number of invalid nodes. Ge et al. [12] proposed a free-floating space robot (FFSR) trajectory planning method based on the dynamic RRT* algorithm, which can rapidly generate a feasible robot movement trajectory. Gan et al. [13] proposed a 1–0 goal-bias RRT algorithm to reduce the computational time and complexity, even in complex environments. Qureshi et al. [14] proposed the potential function-based RRT* (P-RRT*) method by incorporating the APF algorithm into the RRT* method. The proposed algorithm allows a considerable decrease in the number of iterations and thus leads to more efficient memory utilization and an accelerated convergence rate. Jeong et al. [15] proposed Quick-RRT* (Q-RRT*), a modified RRT* algorithm that generates a better initial solution and converges to the optimal solution faster than RRT*. Q-RRT* enlarges the set of possible parent vertices by considering not only a set of vertices contained in a hypersphere, as in RRT*, but also their ancestry up to a user-defined parameter, thus resulting in paths with less cost than those of RRT*. Wang et al. [16] proposed a novel learning-based multi-RRT (LM-RRT) approach for robot path planning in narrow passages. The LM-RRT approach models the tree selection process as a multi-armed bandit problem and uses a reinforcement learning algorithm that learns action values and selects actions with an improved epsilon-greedy (epsilon (t)-greedy) strategy. Lee et al. [17] proposed a motion planning algorithm by exploiting RRT stars (RRT stars) and dynamic movement primitives (DMPs). Hao et al. [18] proposed a Dubins-RRT* algorithm to involve the construction of a recovery path for an agricultural mobile robot (AMR) under kinematic constraints. The planned path avoids obstacles and incurs the minimum cost from a rendezvous point to the recovery platform. However, in general, the problems in the above research are as follows. (1) The improved APF algorithms cannot readily solve the problem of local minima in the search process, and the local minima cannot be adjusted in a timely manner. In addition, when there are obstacles at both the target point and the starting point, the obstacles cannot be effectively avoided to reach the target point quickly, and the generated path is relatively tortuous. (2) The improved RRT algorithms cannot quickly find a reliable path in a complex,

multi-obstacle environment. Moreover, the algorithm cannot be adjusted well for different environmental conditions, which is not conducive to improving the algorithm efficiency, and the generated path is not sufficiently smooth, causing the mechanical arm to undergo impacts and become damaged in actual operation and severely shortening the service life of the mechanical arm.

In response to the above problems, the present paper proposes an improved hybrid three-dimensional path planning algorithm for mechanical arms that combines the APF method and the RRT algorithm. The proposed algorithm is used to solve the path planning problem of the manipulator in an environment with complex obstacles. Compared with the existing path planning algorithms, the main contributions of this article are as follows:

1. In the I-APF (I-APF) method, a heuristic method based on the number of adjacent obstacles to break away from the local minimum is proposed so that the algorithm can quickly eliminate local minima and break away from obstacles.
2. In the I-RRT (I-RRT) algorithm, a triangular nearest neighbor node selection method is proposed, which improves the convergence of the algorithm.
3. Based on the I-APF algorithm and the I-RRT algorithm, a hybrid algorithm is proposed that combines these two improved algorithms to search for the optimal path. First, the distance between the nearest neighbor node and the obstacle is judged. According to the different distances, the I-APF method and the I-RRT algorithm are used to expand the path, which improves the search speed and obstacle avoidance ability of the algorithm.

The rest of this article is organized as follows. Section 3 introduces the classic APF method and the classic RRT algorithm. Then, Section 4 introduces the improved APF and RRT hybrid algorithm. In Section 5, to smooth the path, the path is fitted with a Bezier curve. In Section 6, the planned path is verified by an experimental simulation using Python language development tools and robot operating system (ROS) tools.

3. Background

3.1. Principles of the Classic APF Method

The APF method makes the object move and reach the target point under the action of a force field, which includes a gravitational field and a repulsive field [19–21].

The gravitational field function $U_{att}(q)$ is as follows:

$$U_{att}(q) = \frac{1}{2}\varepsilon\rho^2(q, q_{goal}) \quad (1)$$

where ε is the scale factor, $\rho(q, q_{goal})$ represents the Euclidean distance between the target object q_{goal} and the current position q , and the gravitation $F_{att}(q)$ of the corresponding target object is the derivative of the gravitational field:

$$F_{att}(q) = -\nabla(U_{att}(q)) = \varepsilon(q_{goal} - q) \quad (2)$$

The repulsive field $U_{rep}(q)$ function is as follows:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(q, q_{obs})} - \frac{1}{\rho_0}\right)^2, & \text{if } \rho(q, q_{obs}) \leq \rho_0 \\ 0, & \text{if } \rho(q, q_{obs}) > \rho_0 \end{cases} \quad (3)$$

where η is the repulsion scale factor, $\rho(q, q_{obs})$ represents the Euclidean distance between obstacle q_{obs} and current position q , and ρ_0 represents the influence radius of each obstacle.

Then, the repulsion $F_{rep}(q)$ is the derivative of the repulsive field:

$$F_{rep}(q) = -\nabla U_{rep}(q) = \begin{cases} \eta\left(\frac{1}{\rho(q, q_{obs})} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q, q_{obs})}\nabla\rho(q, q_{obs}), & \text{if } \rho(q, q_{obs}) \leq \rho_0 \\ 0, & \text{if } \rho(q, q_{obs}) > \rho_0 \end{cases} \quad (4)$$

where $\nabla\rho(q,q_{obs})$ represents the derivative of $\rho(q,q_{obs})$, the target object generates gravitation, the object is guided to move towards the target object, and the obstacle generates repulsion to avoid the obstacle, as shown in Figure 1. The resultant force field $U(q)$ (resultant force $F(q)$) that an object receives at any point in the field is equal to the vector sum of the target object's gravitational field (gravity) and the total repulsive field (repulsion) of obstacles encountered. The formula is as follows:

$$U(q) = U_{att}(q) + U_{rep}(q) \quad (5)$$

$$F(q) = -\nabla U(q) \quad (6)$$

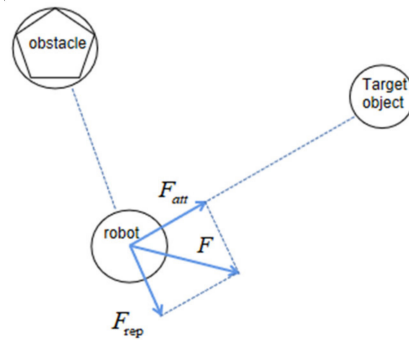


Figure 1. Diagram illustrating the principle of the APF method.

3.2. Principles of the Classic RRT Algorithm

The principle of the classic RRT algorithm is to use an initial point as the root node q_{init} . In each subsequent expansion, a random node q_{rand} is randomly generated in the map. The nearest function selects a node $q_{nearest}$ to q_{rand} based on the Euclidean distance on the existing random tree and expands this node in the direction of q_{rand} with a step distance through the extend function to obtain a new node q_{new} . Evaluating whether q_{new} collides with an obstacle proceeds as follows. If it collides, growth is abandoned, q_{new} is removed, and a random node is regenerated; if there is no collision, q_{new} is added to the random tree, and q_{new} 's parent node is assigned to $q_{nearest}$, as shown in Figure 2. When the distance between the node of the random tree and the target point is less than a specific value m , the program is terminated, as a collision-free path from the starting point to the target point has been obtained. The above steps are followed until reaching the target point, as shown in Figure 3.

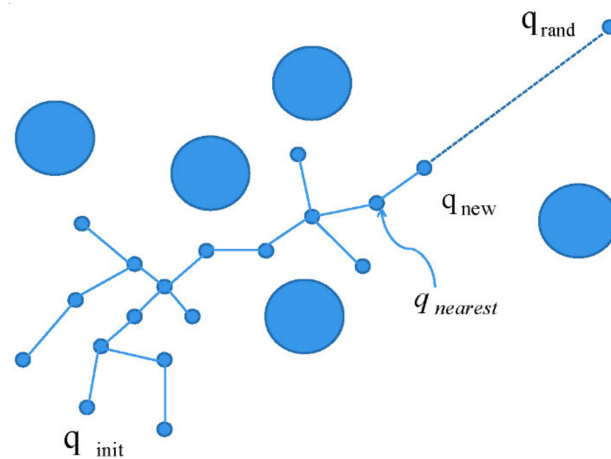


Figure 2. Classic RRT mind map.

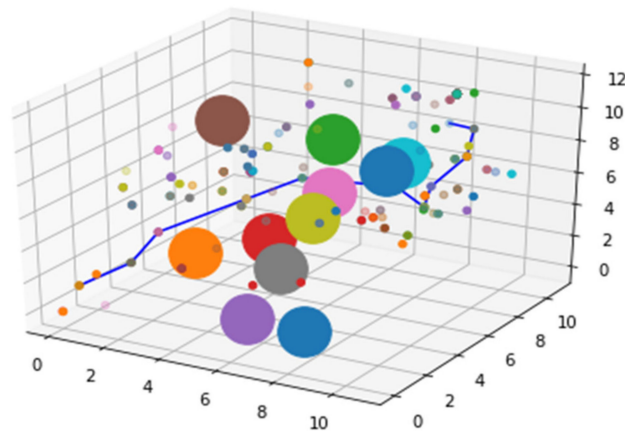


Figure 3. Path planning diagram of the classic RRT algorithm.

4. Improved Algorithms

This section first conducts a collision detection analysis of obstacles, and the results are used to plan the path of the UR5 manipulator. An I-APF method is proposed to avoid local minima of APF, and an I-RRT algorithm is proposed to improve upon the slow convergence speed and poor search efficiency of the classic RRT algorithm. However, the I-APF method leads to tortuous paths and unreachable targets when there are obstacles at the start and end points, and the I-RRT algorithm still has a low search efficiency in an environment with complex obstacles. Notably, the hybrid algorithm that combines the I-APF method and the I-RRT algorithm overcomes the shortcomings of these respective algorithms. Finally, the principles and implementation steps of the hybrid algorithm are given below.

4.1. Collision Detection

A collision detection analysis is carried out. This article uses the universal robot UR5 mechanical arm for the research, uses the geometric envelope in space to simplify the obstacles and mechanical arm model, establishes an environment perception model through sensors, establishes an environmental model map, and divides the map into an obstacle space and obstacle-free space [22]. Usually, obstacles are irregular. To facilitate the calculation, the obstacles are usually idealized as an enclosed ball, and the joints of the robotic arm are idealized as cylindrical. In this way, the computational load can be reduced [23]. Suppose the coordinates of the center of the sphere are (x, y, z) , the radii of the spheres are R_1 and R_2 , and the radius of the cylinder is r ; then, the distances between the coordinates of the centers of the two spheres and the central axis of the cylinder are calculated, denoted d_1 and d_2 . As shown in Figure 4, when $d > r + R$, the robotic arm does not collide with the obstacle; otherwise, the arm collides with the obstacle. This method can greatly improve the computational efficiency.

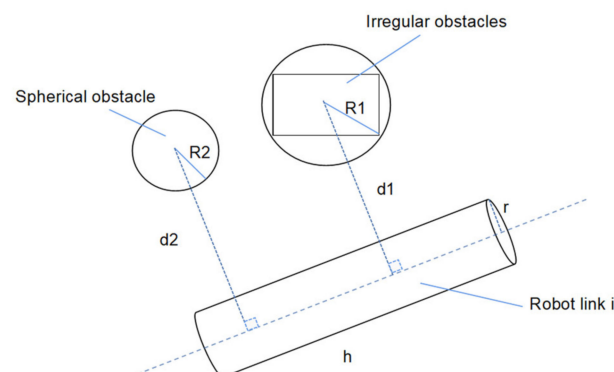


Figure 4. Collision detection model.

4.2. I-APF Algorithm

According to the principles of the classic APF method, the algorithm has some shortcomings. When there is an obstacle at the starting point, the repulsive force causes the path to be tortuous. When the target point is near an obstacle, the obstacle makes it difficult for the object to reach the target point. When the gravitational force and the repulsive force on the object are equal, objects fall into local minima.

In view of the local minima problem of the classic algorithm, some scholars have proposed the idea of using virtual sub-targets [24] to make the robot escape from the local minimum. However, the location of the virtual sub-target is random, which inevitably leads to the blindness of the algorithm; that is, the robot deviates from the target or enters an obstacle area. This paper proposes a heuristic method for deviating from local minima based on the number of adjacent obstacles. The method includes three steps. Step 1: Make a judgement on the local minimum. Step 2: If the object is stuck, take the local minimum as the center, draw a circle with twice the step length in the RRT algorithm as the radius, record the number of obstacles in the circle as O , and record the total number of obstacles in the space as S . Step 3: Introduce a heuristic function to calculate the new potential field force. The specific process is as follows.

When the object falls into a local minimum, the gravitational field and repulsive force field received by the robotic arm are equal in size but in opposite directions. By judging the size and direction of the gravitational field and repulsive force field of the robotic arm, whether the robotic arm falls into a local minimum can be evaluated. When the robotic arm falls into a local minimum, the number of nearby obstacles is obtained by using twice the step length in the RRT algorithm as the radius, and then a heuristic function is established to calculate the new force field. The heuristic function F is as follows:

$$F = \alpha(O/S)M + \beta(1 - O/S)N \quad (7)$$

where α and β are the repulsion scale factor and the gravitational scale factor, respectively, O is the number of obstacles, S is the total number of obstacles in space, M represents the repulsive force of the obstacles in the circle on the robotic arm, and N represents the gravitational force of the target point on the robotic arm. According to the heuristic function, the more obstacles there are in the circle, the greater the repulsive force of the robot arm will be, and the path will expand in the direction deviating from the obstacle to the target point, which is beneficial for the robot arm to quickly avoid the obstacle. Figure 5 shows a path planning diagram of the I-APF method.

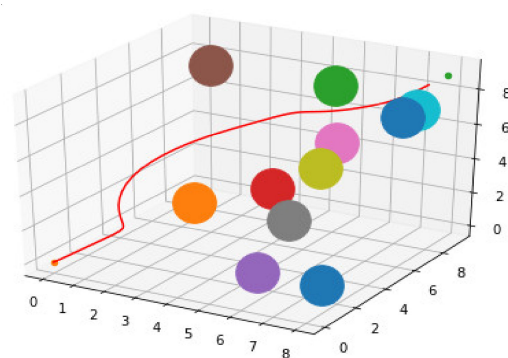


Figure 5. I-APF method path diagram.

4.3. I-RRT Algorithm

The principle of the classic RRT algorithm shows that the algorithm has randomness and a strong obstacle avoidance ability, but this inevitably increases the search time and reduces the efficiency, and the search path is long and tortuous, which takes up a relatively large amount of memory. When there are many obstacles, the efficiency of the algorithm is

greatly reduced. Therefore, this paper improves on the classic RRT algorithm and proposed the I-RRT algorithm. The specific improvements are as follows.

4.3.1. Path Collision Detection

The classic RRT algorithm performs collision detection for nodes but does not conduct collision detection along the path. If the path results in a collision with obstacles, the generated path does not meet the actual requirements. In this paper, collision detection is performed on the path in the `Collision_check_line()` function. The specific method is to pass between two nodes in the random tree, calculate the Euclidean distance between the two nodes in the three-dimensional space, and set the division rate (Discretepoint). The number of divided points is equal to the distance between the two points divided by the division rate. The path is divided into many points, and whether these points collide with obstacles is evaluated.

4.3.2. Goal-Bias Strategy

According to the classic algorithm, the random tree is expanded by selecting random points, resulting in a low path search efficiency. Therefore, this paper adopts the idea of target bias [25]. Random point generation is used to select the target point with a certain probability $P_{\text{goalDampleRate}}$ and can effectively reduce the blindness of the algorithm. When the random probability is greater than $P_{\text{goalDampleRate}}$, a random point is generated, and when the random probability is less than $P_{\text{goalDampleRate}}$, the target point is taken as the random point, which speeds up the convergence rate.

4.3.3. Triangular Nearest Neighbor Node Selection Strategy

The traditional nearest neighbor node selection strategy uses the node in the tree with the closest Euclidean distance to the random sampling point as the nearest neighbor node. This paper proposes a triangular nearest neighbor node selection method, and the specific steps are as follows: If a random sampling point is not the target point, the connections between the random sampling point, the node in the tree, and the target point are established to form a triangular area; then, the sum of the distances of the three sides of the triangle is calculated as the cost function of the nodes. Then, the cost function can be expressed as:

$$\text{Cost}(q) = (||q_{\text{goal}} - q_{\text{rand}}||^2 + ||q_{\text{rand}} - q||^2 + ||q_{\text{goal}} - q||^2) \quad (8)$$

Then, the nearest neighbor node in the triangle is:

$$q_{\text{nearest}} = \{q \in T_{\text{node}} \mid \text{Cost}(q_i) = \min\{\text{Cost}(q)\}\} \quad (9)$$

The triangular nearest neighbor node selection can combine the nearest Euclidean distance to the random point and the nearest Euclidean distance to the target point to select the nearest neighbor node, which improves the convergence efficiency of the algorithm and further reduces the blindness of the algorithm.

4.3.4. Adaptive Step Size

The classic algorithm has a fixed step length for expansion; approaching obstacles cannot be avoided well, causing collisions and occupying a large amount of memory, and the generated step length grows towards a random point without directionality. Both of these issues lead to a reduction in the efficiency of the algorithm, so an adaptive step size strategy is adopted [26]. When the minimum distance between q_{nearest} and the obstacle is greater than the step size, the obstacle is marked as s , the idea of gravity in the APF method is introduced into the adaptive step size of the RRT algorithm [27], and the random tree is guided to grow towards the target. On the basis of the original RRT algorithm growing towards random points, the step component $G(n)$ towards the target is added so that the

new node has a tendency to deviate towards the target point. The formula of step F(n) is defined as follows:

$$F(n) = R(n) + G(n) \quad (10)$$

where F(n) represents the traction step length of the nth node, R(n) represents the traction step length of the nth node by a random point, and G(n) represents the traction step length of the nth node by the target object. The gravitational potential energy U of the target point to the nearest node is known by the APF method and is defined as follows:

$$U = \frac{1}{2} K_p * \|x_{goal} - x_{near}\|^2 \quad (11)$$

where k_p represents the gravitational coefficient and $\|x_{goal} - x_{near}\|$ represents the absolute value of the Euclidean distance between the position vector x_{goal} of the target point and the position vector x_{near} of the nearest node. Then, the gravitational force G is the derivative of the gravitational potential energy U , namely,

$$G = K_p * \|x_{goal} - x_{near}\| \quad (12)$$

Then

$$G(n) = \rho * k_p \frac{x_{goal} - x_{near}}{\|x_{goal} - x_{near}\|} \quad (13)$$

where ρ represents the step length of random growth; then, $R(n)$ can also be deduced as

$$R(n) = \rho * \frac{x_{rand} - x_{near}}{\|x_{rand} - x_{near}\|} \quad (14)$$

where $\|x_{rand} - x_{near}\|$ represents the absolute value of the Euclidean distance between the position vector x_{rand} of the random point and the position vector x_{near} of the nearest node. By inserting Equations (13) and (14) into Equation (10) to obtain the traction step length of the nearest node as Equation (15), defined as $F_1(n)$, the following formula is obtained:

$$F_1(n) = \rho * \frac{x_{rand} - x_{near}}{\|x_{rand} - x_{near}\|} + \rho * k_p \frac{x_{goal} - x_{near}}{\|x_{goal} - x_{near}\|} \quad (15)$$

When the minimum distance to the obstacle is less than the step length, two situations are possible.

Case 1: First, a virtual new node is generated according to the above steps, and whether the distance between the virtual new node and obstacle s is less than the distance between the nearest node and the obstacle is evaluated. If the distance is less than the distance between the nearest node and the obstacle, then the virtual new node has a tendency to approach the obstacle is proven. Then, the virtual new node is removed, the step length of the new node growth is changed, and the step length is reduced on the basis of the step length $F_1(n)$, which is defined as the step length $F_2(n)$. The formula is as follows

$$F_2(n) = \frac{\text{dist}_2}{\text{dist}_1} \left(\rho * \frac{x_{rand} - x_{near}}{\|x_{rand} - x_{near}\|} + \rho * k_p \frac{x_{goal} - x_{near}}{\|x_{goal} - x_{near}\|} \right) \quad (16)$$

where dist_1 represents the distance between the nearest node and obstacles and dist_2 represents the distance between the virtual new node and obstacles. Whether the node collides with an obstacle is evaluated. If it collides, a random node is regenerated; if there is no collision, a new node is added to the random tree.

Case 2: If the distance between the virtual new node and obstacles is greater than the distance between the nearest node and obstacles, it proves that the new node has a tendency to grow away from the obstacle. Then, the step length of the new node generation

adopts the step length of the classic RRT algorithm, defined as the step length $F_3(n)$, and the formula is as follows:

$$F_3(n) = \rho * \frac{x_{rand} - x_{near}}{\|x_{rand} - x_{near}\|} \quad (17)$$

Then, whether the node collides with an obstacle is evaluated; if it collides, a random node is regenerated, and if there is no collision, a new node is added to the random tree.

4.3.5. Removing Redundant Nodes

Due to the randomness of the classic RRT algorithm, the path may oscillate. Redundant nodes are removed to process the path from the starting point to the target point generated by the random tree [28]. Starting from the first node q_{init} , the subsequent path nodes are connected in turn, the second node is ignored, and the third node is connected. If the object does not collide with the obstacle, the second node on the path is deleted. If there is a collision, then the node is retained, the parent node of the collision point is used as the new evaluation node, and the above steps are repeated until the target point is reached. The final series of reserved nodes is saved into Path 2 and connected to obtain the path after removing redundant nodes, as shown in Figure 6. The collision here refers to the collision detection of the path mentioned in Section 4.3.1. Figure 7 shows the path diagram generated by the I-RRT algorithm. Algorithm 1 shows the pseudocode of the I-RRT algorithm.

Algorithm 1: I-RRT

```

1. initialization
2.  $S \leftarrow q_{init}$ 
3. while true do
4.    $q(rand) = random() \text{ or } end();$ 
5.    $q(rand) = random()$ 
6.    $q(nearest) \leftarrow Triangle\_nearest\_list\_index(node\_list, q(rand));$ 
7.    $dist1 \leftarrow Nearest\ distance(q(nearest), obstacles);$ 
8.   if  $dist1 > expandDis$ 
9.      $q(new) \leftarrow Extend(expandDis, q(nearest), Direction\ Angle)$ 
10.  else
11.     $q(new) = virtual(Extend(expandDis, q(nearest), Direction\ Angle))$ 
12.     $dist2 \leftarrow Nearest\_distance(q(new), obstacles)$ 
13.    if  $dist1 > dist2$ 
14.       $q(new) \leftarrow Extend(expanddis, q(nearest), Direction\ Angle)$ 
15.    else
16.       $q(new) \leftarrow Extend(expandDis, q(nearest), Direction\ Angle)$ 
17.    if  $collision\_check(q(new))$ 
18.       $Tree.add(q(new))$ 
19.    if  $dist(q(new), q(end)) < expandDis$ 
20.      return Goal
21.    else
22.      continue
23.  return path
24.  $path2 \leftarrow remove\_redundant\_nodes(path)$ 
25. return Graph

```

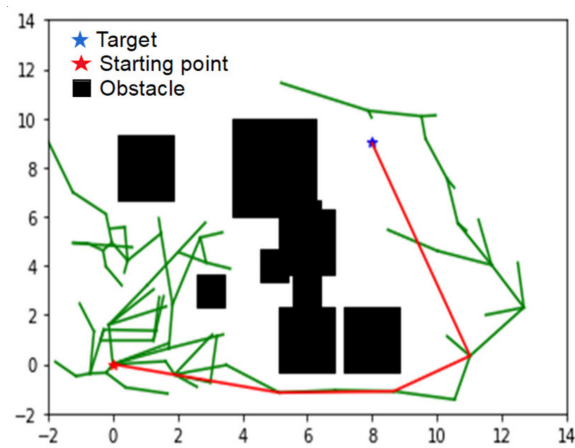


Figure 6. Diagram of the I-RRT algorithm implementation to remove redundant nodes.

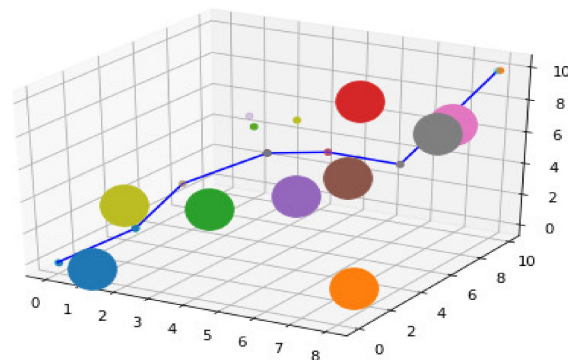


Figure 7. I-RRT algorithm generation path diagram.

4.4. Improved Hybrid Algorithm of the APF and RRT

The I-APF method deals with the local minimum problem in the classic algorithm, but when there are obstacles at the starting point, the repulsion causes a tortuous path. When there are obstacles at the target point, the target is unreachable, and vibration occurs. The I-RRT algorithm significantly improves the search efficiency, but its efficiency is still low in the case of multiple obstacles. This article adopts the strategy of combining the two improved algorithms (I-APF, I-RRT) to give full play to the advantages of the two algorithms to avoid defects.

4.4.1. Principle of the Improved Hybrid Algorithm

The principle of the hybrid algorithm is as follows. First, the tree node is initialized, and the distance between the nearest node and the obstacle is continuously detected. If it is detected that the minimum distance between the current node and the obstacle is greater than twice the step length, it means that there is no obstacle near the current node, and the I-APF method is used for rapid expansion. If the minimum distance between the current node and the obstacle is less than twice the step length, the I-RRT algorithm is adopted to make full use of the efficient obstacle avoidance ability of the RRT algorithm, and the above steps are repeated until the target point is reached. The hybrid algorithm can effectively improve the efficiency of path searching and resolve the following problem of the APF method: When there are obstacles at the starting point, the repulsive force causes a tortuous path, and the target is unreachable when there is an obstacle at the target point. Furthermore, the hybrid algorithm also solves the problem of the classic RRT algorithm, which has a significantly lower efficiency when there are more obstacles, and the generated path is shorter and smoother.

4.4.2. Improved Hybrid Algorithm Implementation

According to the principle of the improved APF and RRT hybrid algorithm, the specific implementation steps are described as follows.

Step 1: Initialize the parameters, and define the obstacle environment, starting point, target point, step length, and target sampling rate.

Step 2: Determine the distance between the current node and the obstacle. If the distance between the current node and the nearest obstacle is greater than twice the step length, execute Step 3. If the distance between the current node and the nearest obstacle is less than twice the step length, execute Step 4.

Step 3: Use the I-APF method to search and move forward under the combined force of the target and obstacle. (1) Calculate the gravitational and repulsive forces. (2) Determine whether the gravitational and repulsive forces experienced by the current node are equal and opposite. If they are equal, the object falls into a local minimum, the heuristic method based on the number of adjacent obstacles is used to escape from the local minimum so that the algorithm escapes from the local minimum, and the end effector of the robotic arm is guided to continue to move. If they are not equal, proceed to Step (3). (3) If the distance between the current node and the target point is less than the step length or if the distance between the nearest node and the nearest obstacle is less than twice the step length, then the I-APF method search process is ended, the path node obtained by the APF method is added to the Pathpath, and the Pathpath and the latest node $q(\text{new})$ are returned. Otherwise, jump to Step (1).

Step 4: If the distance between the current node and the nearest obstacle is less than twice the step length, the I-RRT algorithm is used to search for the path. (1) Initialize the tree, set the initial node and target point, and define the step size, target sampling rate, and segmentation rate. (2) Start the iteration and sample the state space. When the random probability is less than the target sampling rate, the sampling point selects the target point. If it is greater than the target sampling rate, random sampling points in the space are selected. (3) Select the nearest neighbor node according to the triangle nearest neighbor node selection method and calculate the distance dist_1 between this node and the nearest obstacle. (4) Determine whether the distance dist_1 is greater than the step length. If so, the step length $F_1(n)$ is used for expansion. If not, generate a virtual new node according to the step length $F_1(n)$ and calculate the distance dist_2 between the virtual new node and the nearest obstacle. (5) If $\text{dist}_2 < \text{dist}_1$, the new node has a tendency to move towards obstacles. Remove the virtual new node and use step $F_2(n)$ to expand. If $\text{dist}_2 > \text{dist}_1$, the new node has a tendency to move away from obstacles; remove the virtual new node and use the step size $F_3(n)$ to expand. (6) Determine whether the new node collides with obstacles; if there is a collision, skip to Step (2). Re-sample random points; if there is no collision, add the new node to the tree and assign the parent node of the new node to the nearest node (q_{nearest}). (7) When the distance between the new node and the target point is less than the step length or the distance between the nearest node and the nearest obstacle is greater than twice the step length, end the iterative process; otherwise, skip to Step (2). (8) Use the collision detection method of the path to perform the process of removing redundant nodes on the generated path to obtain the processed path. (9) Add the processed path node to the Pathpath and return the Pathpath and the latest node $q(\text{new})$.

Step 5: Determine whether the distance between the new node and the target point is less than the step length. If so, reach the target point, connect the new node and the target point, output the path graph, obtain a collision-free path from the start point to the end point, and exit the program. Otherwise, skip to Step 2.

According to the specific implementation steps of the hybrid algorithm, Figure 8 shows the flowchart of the hybrid algorithm and Algorithm 2 shows the pseudocode of the hybrid algorithm.

Algorithm 2: APF-RRT

```

1. Initialization;
2. While True do
3.   if distance(q(new),obstacle_list)>2*expandDis
4.     q(new) ← q(recent_node);
5.     q(new) ← Extend( $F_{att}$ ,  $F_{rep}$ )
6.     Path.append(q(new))
7.     if dist(q(new),q(end))< expandDis
8.       return Goal
9.     else
10.      Continue
11.   return Path recent_node
12.   Pathpath.append(path)
13.   else
14.     q(new) ← q(recent_node);
15.     q(rand)= random() or end();
16.     if q(rand)= random()
17.       q(nearest)← Triangle_nearest_list_index(node_list,q(rand));
18.       dist1 ← Nearest distance(q(nearest), obstacles);
19.       if dist1 > expandDis
20.         q(new)← Extend(expandDis,q(nearest), Direction Angle)
21.       else
22.         q(new)=virtual(Extend(expandDis,q(nearest),Direction Angle))
23.         dist2← Nearest_distance(q(new), obstacles)
24.         if dist1 > dist2
25.           q(new) ← Extend(expanddis, q(nearest), Direction Angle)
26.         else
27.           q(new) ← Extend(expandDis,q(nearest),Direction Angle)
28.       if collision_check(q(new))
29.         Tree.add(q(new))
30.       if dist(q(new),q(end))< expandDis
31.         return Goal path
32.       else
33.         continue
34.   path2← remove_redundant_nodes(path)
35.   return Path2 recent_node
36.   Pathpath.append(path2)
37. return Pathpath Graph

```

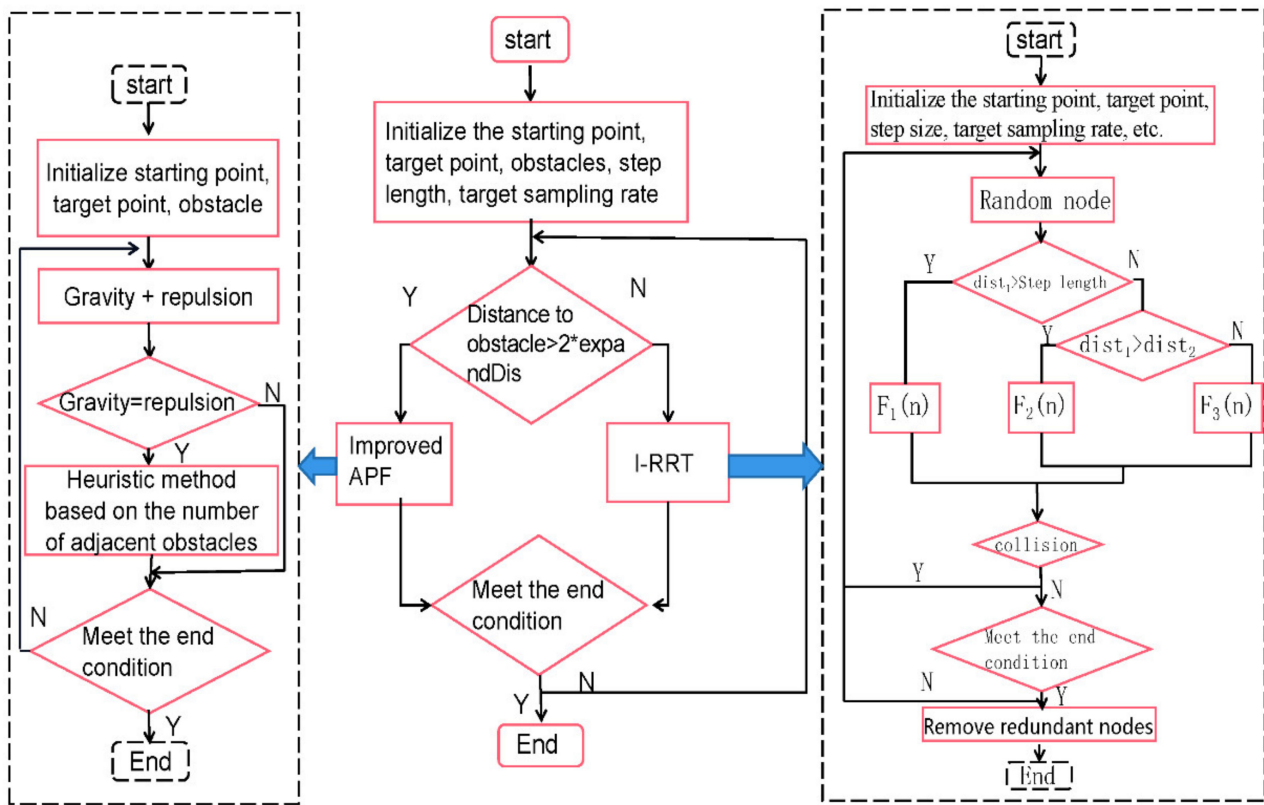


Figure 8. APF-RRT algorithm flow chart.

5. Bezier Curve Path Smoothing

Aiming at the phenomenon that the path generated by the algorithm has turning points and is not sufficiently smooth, reducing the performance of the robot arm due to its acceleration in actual operation, this paper uses Bezier curves to smooth the path [17,29]. Smoothing is realized on the basis of the original path, and $n + 1$ nodes obtain the formula of an n -order Bezier curve:

$$C(u) = \sum_{i=0}^n B_{n,i}(u)p_i, \quad u \in [0, 1] \tag{18}$$

where P_i represents $n + 1$ points in space and the weight coefficient $B_{n,i}(u)$ with the parameter u is the Bernstein basis function. The calculation method is as follows:

$$B_{n,i}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i} \tag{19}$$

The final generated curve has a relationship with each of the $n + 1$ points. These points determine the final direction of the curve and are called control points. The Bessel order in Equation (18) is n and is controlled by the $n + 1$ control points. The start point and end point correspond to $u = 0$ and $u = 1$, respectively. The curve after Bezier fitting is shown in Figure 9a. The slight gap between the fitted curve and the original path may risk collision between the path and the obstacle. It can be seen from the figure that the fitting curve is likely to result in a collision with the obstacle only when the obstacle environment is very complex. For the simulation experiment, the success rate of the 200-path fitting experiment in this paper is 100%.

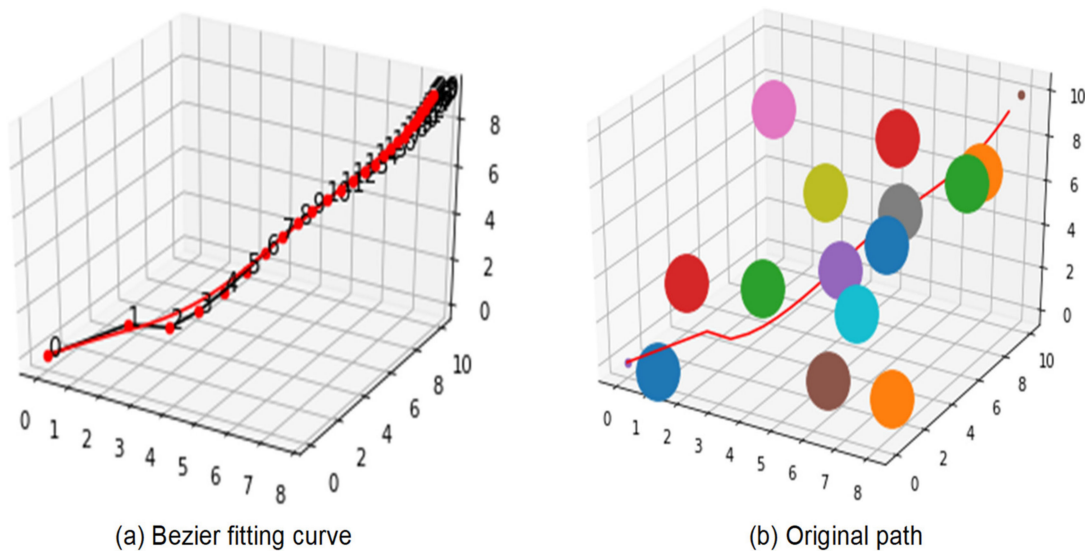


Figure 9. Comparison of the Bezier fitted curve and original path.

6. Simulation and Experiment (Python and ROS Simulation)

In this section, the improved APF and RRT hybrid algorithm is verified experimentally. To verify that the algorithm can maintain excellent results in a multi-obstacle environment, this hybrid algorithm is compared with the RRT, RRT*, and P_RRT* algorithms. The RRT* algorithm is a landmark algorithm among the improved RRT algorithms. It has a higher convergence rate and has been widely studied by scholars, while the P_RRT* algorithm introduces the idea of the APF method on the basis of RRT*, making the P_RRT* algorithm one of the path planning algorithms with the highest convergence efficiency. Experiments were carried out on different obstacle environments to verify the effectiveness of the algorithm. This experiment used Python language development tools on a laboratory desktop HP computer with 4-GB memory and an Intel(R) Core (TM)i5-6500 CPU @3.20 GHz–3.19 GHz to find a smooth and collision-free path. Combining theory with reality, this work chose a simulation robot, the UR5 of the Danish UAO Company, for simulation experiments. The UR5 is a six-degree-of-freedom manipulator. Table 1 shows the motion control parameters of the UR5 robotic arm.

Table 1. Manipulator motion control parameter table.

Parameter	Value
Search step ρ	1.0 dm
Starting point	(0,0,0)/dm
Ending point	(8,10,10)/dm
Gravitational coefficient ε , K_p	0.05
Repulsion coefficient η	100.0
α	0.4
β	0.6
Obstacle influence radius ρ_0	0.3 dm
Target sampling rate	0.1

6.1. Experiments and Analysis in Python

Experiment 1. A comparison among the algorithms for different numbers of obstacles was carried out to verify that the improved APF and RRT hybrid algorithm maintained a better search effect in the case of multiple obstacles. With a gradually increasing number of obstacles, the advantages and disadvantages of the proposed path planning algorithm were compared with those of the classic RRT algorithm, the RRT* method, and the P_RRT* algorithm. Each group of experiments was performed 200 times, and the average of the

results is shown in Table 2, where the search success rate was that the search was within 100 s of a successful search, and the search was unsuccessful outside of 100 s.

Table 2. Comparison of various algorithms for different obstacle numbers.

Obstacle	Average Search Time				Average Path Length				Average Number of Sampling Nodes				Search Success Rate			
	RRT		RRT*		RRT*		P_RRT*		P_RRT*		APF-RRT		APF-RRT			
10	20.0	22.5	70.4	100%	7.7	21.0	20.4	100%	6.9	20.6	20.5	100%	2.0	18.0	10.3	100%
12	21.0	22.8	74.1	100%	8.13	21.2	20.6	100%	7.3	20.7	20.8	100%	2.1	18.0	10.4	100%
14	37.8	23.5	95.7	95%	10.1	21.8	21.9	100%	7.4	20.7	21.0	100%	2.8	18.0	11.5	100%
16	40.9	23.6	111.2	90%	10.3	21.8	23.8	100%	7.8	21.1	21.3	100%	3.2	19.1	11.9	100%

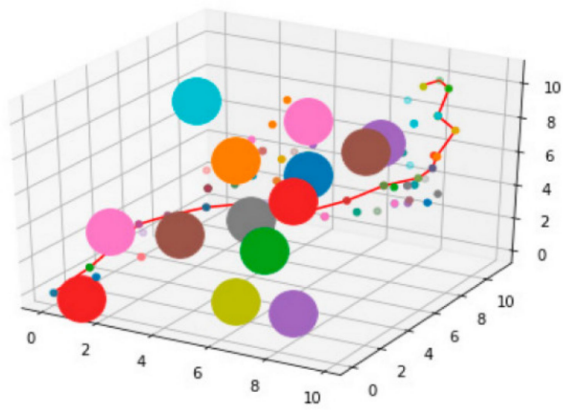
According to Table 2, the search time, path length, number of sampling nodes, and search success rate of the algorithms are compared. The search time of the classic RRT algorithm was relatively short when there are few obstacles. When the number of obstacles gradually increased, the search time greatly increased, and the search success rate decreased. Compared with the classic RRT algorithm, the RRT* method was superior. The average search time and the average number of sampling nodes were greatly improved, and the algorithm maintained good results when the number of obstacles gradually increased. Compared with the RRT* algorithm, the P_RRT* algorithm exhibited further improvements, and the search efficiency was higher. With an increasing number of obstacles, stable search results can be maintained, but compared with the improved APF and RRT hybrid algorithm in this article, it still had the disadvantages of a low search efficiency, tortuous paths, and high average number of sampling nodes, which consumed more computing memory. The hybrid algorithm in this paper showed a better effect when the number of obstacles gradually increased.

Experiment 2. To analyze the effectiveness of the improved APF and RRT hybrid algorithm, this work compared various algorithms under the same conditions of obstacles and step lengths. Each set of experiments was performed 200 times, and the average of the results is shown in Table 3.

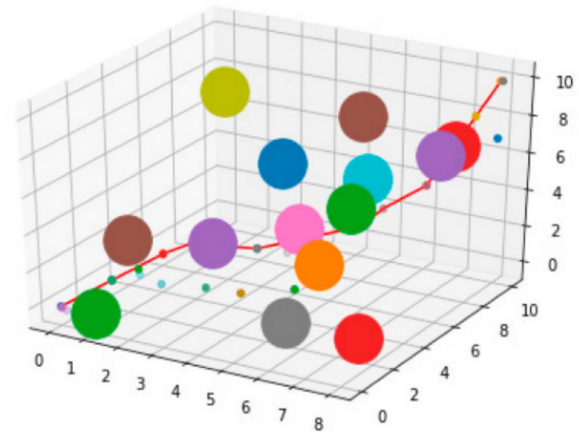
Table 3. Comparison of various algorithms under the same conditions.

	Average Search Time	Average Number of Sampling Nodes	Average Path Length	Search Success Rate
RRT	37.8	95.7	23.5	95%
RRT*	10.1	21.9	21.8	100%
P_RRT*	7.4	21.0	20.7	100%
APF-RRT	2.8	11.5	18.0	100%

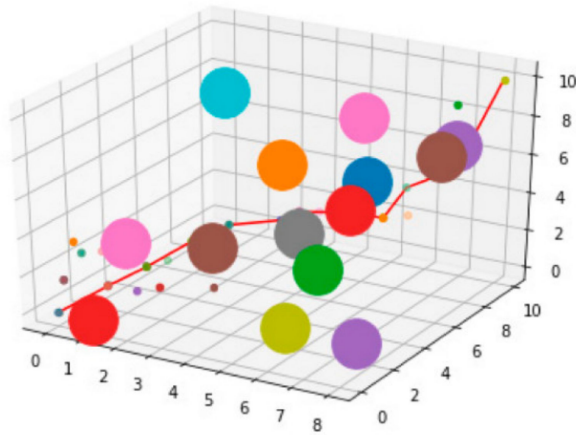
According to the data in Table 3, the classic RRT algorithm had a slow search speed, a large number of sampling nodes, and tortuous paths when there were many obstacles, as shown in Figure 10a. Compared with the classic RRT algorithm, the RRT* algorithm had a great improvement in the average search time, the path planning efficiency was higher, and the generated path was smoother, as shown in Figure 10b. The average search time of the P_RRT* algorithm was shorter than that of the RRT* algorithm. However, compared with the improved APF and RRT hybrid algorithm in this paper, there were still shortcomings, such as a low search efficiency, tortuous paths, and a large demand on the computing memory. The improved hybrid algorithm in this paper still showed excellent results in the case of many obstacles, the search efficiency was higher, the path was shorter and smoother, and it overcame the phenomenon of tortuous paths and unreachable targets in the APF method when there were obstacles near the starting point and target point. To a certain extent, the blindness of the RRT algorithm was reduced, and the efficient obstacle avoidance ability of the RRT algorithm was fully utilized, as shown in Figure 10d.



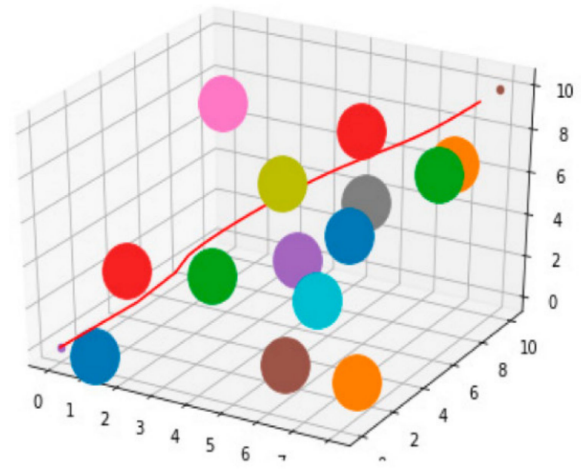
(a) Classic RRT algorithm



(c) P_RRT* algorithm



(b) RRT* algorithm



(d) APF-RRT

Figure 10. Comparison of paths generated by the different algorithms under the same conditions, such as the number of obstacles and step lengths.

The comparison and analysis with the classic RRT algorithm, RRT* algorithm, and P_RRT* algorithm verified the effectiveness of the improved hybrid algorithm in this paper.

Experiment 3. The reliability of the algorithm was evaluated by changing the step length of the RRT algorithm in the hybrid algorithm. Each step was carried out 200 times. The experimental results are shown in Table 4, and the trajectory of different step lengths is shown in Figure 11.

Table 4. Comparison of hybrid algorithms with different step sizes.

Step	Average Search Time	Average Number of Sampling Nodes	Average Path Length	Search Success Rate
0.6	6.2	13.8	19.4	100%
0.8	5.9	12.6	19.1	100%
1.0	3.2	11.9	19.1	100%
1.2	5.6	12.0	19.1	100%
1.4	6.7	12.7	19.2	100%

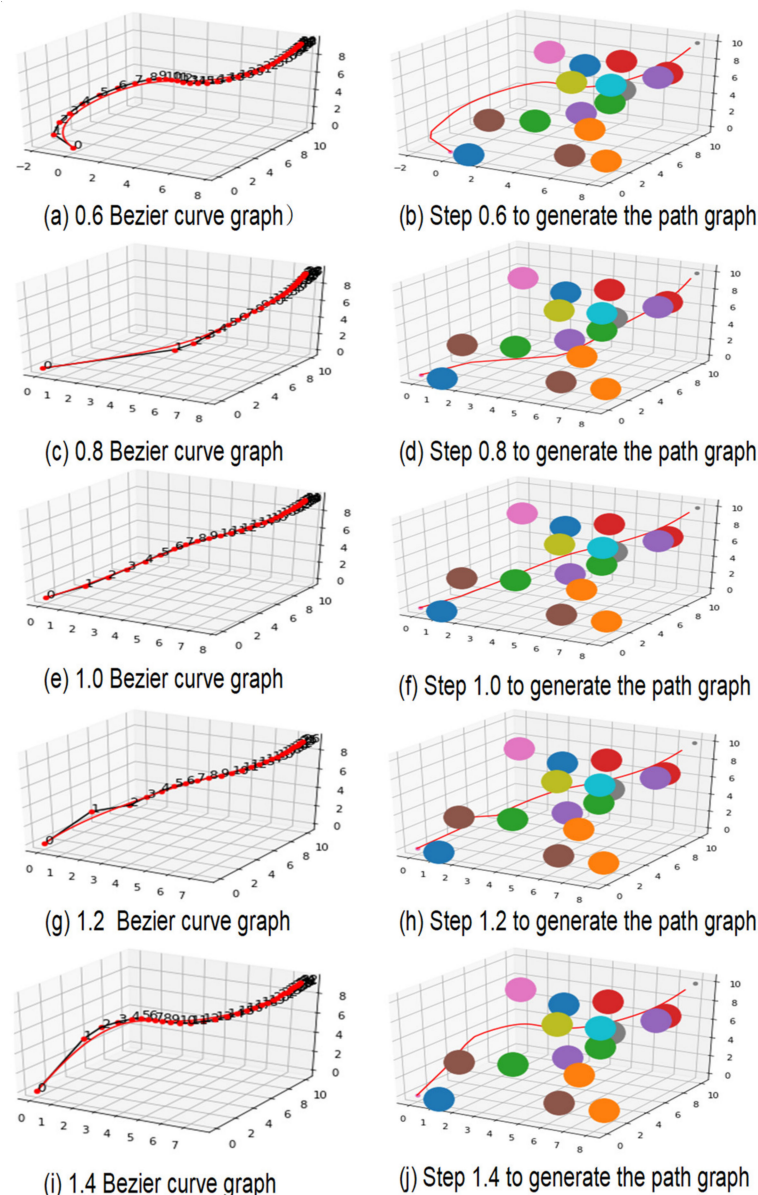


Figure 11. Comparison of hybrid algorithms with different step sizes.

As shown in Table 4, the search time was the smallest when the step size was 1.0 (42.8~52.2% shorter than at the other step sizes). The search time at the left end with a step size of 1.0 gradually decreased, while the search time at the right end had an increasing trend due to the excessive step size. The search time of the path did not decrease with increasing step size, and when the step size was 1.0, the search path was the shortest and the time was the shortest. In summary, when the step size was 1.0, the algorithm achieved the optimal effect. Therefore, the selection of the step size is still very important for the hybrid algorithm, and the experiment verified the reliability of the algorithm.

Figure 11 shows the trajectory diagrams of different step lengths fitted with Bezier curves. The effect is best when the step length is 1.0 in the figure: The trajectory is smooth, and the path is the shortest.

6.2. ROS Simulation Experiment

This section took the UR5 mechanical arm as the research object, conducted a simulation analysis in the ROS, and set up the scenes required for robotic arm motion planning in MoveIt. The objects were added in MoveIt by creating a topic publisher, setting the

basic shape and position of required obstacles and target objects, and publishing object information. The experiment was demonstrated by the visualization tool Rviz in the ROS. First, the UR5 robotic arm model was loaded, the simulation function was enabled, and the start point and end point of the robotic arm were set. As shown in Figure 12, the gray robotic arm was the pose of the starting point, and the yellow robotic arm was the pose of the ending point. The obstacles in the picture are a table and eight regular-shaped cubes, and the green cuboid is the grasping target. This scenario was a locally restricted test scenario with obstacle constraints and platform constraints. Before motion planning, the error transformation matrix was used to compensate for the parking error of the manipulator, and the positions of the start point and end point of the manipulator were the postures after compensation. The improved APF and RRT hybrid algorithm was added to the Open Motion Planning Library (OMPL), and the corresponding YAML Ain't Markup Language (YAML) was modified. The Kinematics and Dynamics Library (KDL) solver that comes with MoveIt was used to solve the angle changes of each joint during the movement from the starting point to the ending point. A smooth and collision-free path from the starting point to the target point was obtained, as shown in Figure 13. The motion trajectory was smooth and did not collide with obstacles.

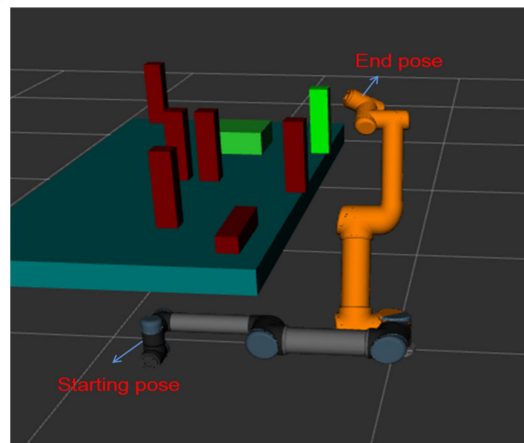


Figure 12. The poses of the start and end points of the UR5.

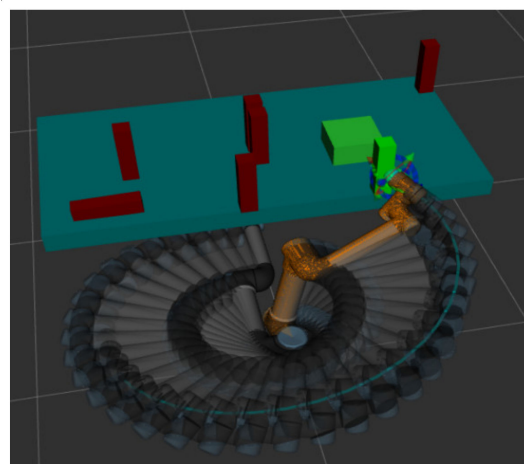


Figure 13. The trajectory diagram of the UR5 robotic arm.

Figure 14 shows the changes in the six joints during the movement of the robotic arm. The position change of each joint was relatively stable and met the real movement needs of the robotic arm. The position of the joints at the start and end positions are shown in Table 5. Table 6 shows the average search time and search success rate of different algorithms in the

same obstacle environment. Each set of experiments was carried out 20 times. Simulation experiments verified the feasibility of the algorithm.

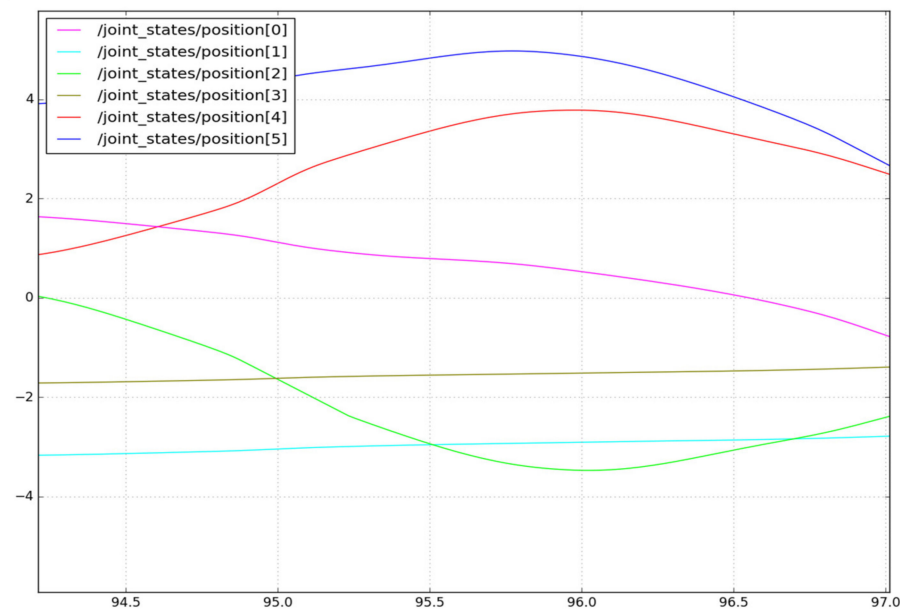


Figure 14. The position change of each joint of the robotic arm.

Table 5. Initiation and termination of the robotic arm joint.

	Joint 0	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5
The starting position	0.00112	0.00322	−0.00114	5.793×10^{-5}	4.422×10^{-6}	5.521×10^{-6}
The ending position	−1.42552	−0.67803	−0.27888	4.10375	−0.70497	−3.14746

Table 6. Search time and search success rate of different algorithms.

	RRT	RRT*	P_RRT*	APF-RRT
Search time	19.1	9.3	6.2	3.0
Search success rate	90%	100%	100%	100%

7. Conclusions and Discussions

This paper improves the classic RRT algorithm and the classic APF algorithm and combines the two improved algorithms. The combined hybrid algorithm made full use of the efficient obstacle avoidance ability of the RRT algorithm and the efficient guidance ability of the APF method. Moreover, it avoided the disadvantages of each algorithm, and the comparison and analysis with the other three algorithms verified the effectiveness of the improved algorithm in this paper.

7.1. Discussions

To resolve the shortcomings of the classic APF method and the classic RRT algorithm, this paper proposes an improved path planning method that combines the APF method and the RRT algorithm for the path planning of the manipulator. First, the distance between the obstacle and the nearest node was evaluated. Through the distance value, the I-APF and I-RRT algorithms were used to explore the path. This simultaneously solved the shortcoming that the APF method cannot reach the target point when there are obstacles at the target point and made full use of the efficient obstacle avoidance ability of the RRT algorithm. The two algorithms alternated exploring the path until the target point was reached. The I-APF method introduced the heuristic method of breaking away from the

local minimum based on the number of adjacent obstacles to solve the local minimum problem in the algorithm. The I-RRT algorithm included the triangular nearest neighbor node selection strategy, which effectively improved the obstacle avoidance ability and efficiency of the algorithm. In the same obstacle environment, compared with the classic RRT algorithm, the search time of the improved APF and RRT hybrid algorithm was reduced by 92.5%, the number of sampling nodes was reduced by 87.9%, and the path length was reduced by 23.4%. Compared with the RRT* algorithm, the search time of the APF and RRT hybrid algorithm was reduced by 72.2%, the number of sampling nodes was reduced by 47.4%, and the path length was reduced by 17.4%. Compared with the P_RRT* algorithm, the search time of the APF and RRT hybrid algorithm was reduced by 62.1%, the number of sampling nodes was reduced by 45.2%, and the path length was reduced by 13.0%. With an increase in the number of obstacles, the improved hybrid algorithm also showed the excellent effect of steady increases in the search time, number of sampling nodes, and path length.

7.2. Conclusions

The improved hybrid algorithm gives full play to the advantages of the individual algorithms while avoiding the disadvantages of the individual algorithms and is more suitable for the path planning of robotic arms. However, in a complex environment with more obstacles, the number of sampling nodes of the improved hybrid algorithm increases significantly. This is because the number of nodes randomly sampled by the RRT algorithm in the improved hybrid algorithm increases, increasing the consumption of computational memory and reducing the search efficiency. Therefore, it is recommended that future work focus on how to reduce the number of sampling nodes of the improved hybrid algorithm to reduce memory consumption and improve the efficiency of the algorithm.

This article focuses on the research of robotic arm path planning in a three-dimensional environment, which can be used in a variety of unstructured environments, such as warehouse automation and handling on production lines. Future research will focus on adaptive path planning in a dynamic environment.

Author Contributions: Conceptualization, Q.Y., J.Y., R.S. and H.B.; Data curation, Q.Y., J.Y., R.S. and H.B.; Formal analysis, Q.Y., J.Y. and H.B.; Funding acquisition, Q.Y.; Investigation, R.S.; Methodology, Q.Y. and J.Y.; Project administration, Q.Y. and J.Y.; Supervision, Q.Y., J.Y., R.S. and H.B.; Visualization, Q.Y. and J.Y.; Writing—original draft, J.Y.; Writing—review and editing, J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 51865004), Guizhou Province Education Department Science and Technology Talents Support Project (Branch Support KY [2017]062), and Horizontal Topic (K19-0204-001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Zeng, A.; Yu, K.-T.; Song, S.; Suo, D.; Walker, E.; Rodriguez, A.; Xiao, J. Multi-view self-supervised deep learning for 6D pose estimation in the Amazon Picking Challenge. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1386–1393.
2. Wang, M.; Hou, Z. Continuous Trajectory Point Control Research of Six Degree of Freedom Mechanical Intelligent Arm Position. *Int. J. Precis. Eng. Manuf.* **2018**, *19*, 221–226. [[CrossRef](#)]
3. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*; Springer: New York, NY, USA, 1986; pp. 396–404.
4. LaValle, S.M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. *Computer Science Dept.* **1998**, *11*, 4.

5. Zheng, Y.; Shao, X.; Chen, Z.; Zhang, J. Improvements on the virtual obstacle method. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420911763. [CrossRef]
6. Sun, J.; Liu, G.; Tian, G.; Zhang, J. Smart Obstacle Avoidance Using a Danger Index for a Dynamic Environment. *Appl. Sci.* **2019**, *9*, 1589. [CrossRef]
7. Zhang, J.; Wang, C.; Zhao, J. Overtaking path planning and tracking control of automobile curve based on improved artificial potential field method. *Automob. Eng.* **2021**, *43*, 546–552.
8. Han, Y.; Li, S. UAV trajectory planning based on improved artificial potential field method. *Syst. Eng. Electron. Technol.* **2021**, 1–9. Available online: <https://scjg.cnki.net/kcms/detail/detail.aspx?filename=XTYD20210528005&dbcode=CJFQ&dbname=CAPJ2021&v=> (accessed on 20 September 2021). (In Chinese).
9. Zhang, W.; Wei, S.; Zeng, J.; Wang, N. Multi-UUV path planning based on improved artificial potential field method. *Int. J. Robot. Autom.* **2021**, *36*. [CrossRef]
10. Tian, Y.; Zhu, X.; Meng, D.; Wang, X.; Liang, B. An Overall Configuration Planning Method of Continuum Hyper-Redundant Manipulators Based on Improved Artificial Potential Field Method. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4867–4874. [CrossRef]
11. Li, Y.; Wang, S.; Jiang, L. Motion planning of mobile manipulator based on RRT with sparse nodes. *China Mech. Eng.* **2021**, *32*, 1462.
12. Ge, J.; Liu, L.; Dong, X.; Tian, W.; Lu, T. Free-floating space robot trajectory planning based on dynamic RRT*. *J. Aeronaut.* **2021**, *42*, 176–185.
13. Gan, Y.; Zhang, B.; Ke, C.; Zhu, X.; He, W.; Ihara, T. Research on Robot Motion Planning Based on RRT Algorithm with Nonholonomic Constraints. *Neural Process. Lett.* **2021**, *53*, 3011–3029. [CrossRef]
14. Qureshi, A.H.; Ayaz, Y. Potential functions based sampling heuristic for optimal path planning. *Auton. Robot.* **2015**, *40*, 1079–1093. [CrossRef]
15. Jeong, I.-B.; Lee, S.-J.; Kim, J.-H. Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. *Expert Syst. Appl.* **2019**, *123*, 82–90. [CrossRef]
16. Wang, W.; Zuo, L.; Xu, X. A Learning-based Multi-RRT Approach for Robot Path Planning in Narrow Passages. *J. Intell. Robot. Syst.* **2017**, *90*, 81–100. [CrossRef]
17. Lee, H.; Kim, H.; Kim, H.J. Planning and Control for Collision-Free Cooperative Aerial Transportation. *IEEE Trans. Autom. Sci. Eng.* **2016**, *15*, 189–201.
18. Hao, B.; Yan, Z. Recovery path planning for an agricultural mobile robot by Dubins-RRT* algorithm. *Int. J. Robot. Autom.* **2018**, *33*. [CrossRef]
19. Zhou, H.; Zhou, S.; Yu, J.; Zhang, Z.; Liu, Z. Zhou Trajectory Optimization of Pickup Manipulator in Obstacle Environment Based on Improved Artificial Potential Field Method. *Appl. Sci.* **2020**, *10*, 935. [CrossRef]
20. Yao, Q.; Zheng, Z.; Qi, L.; Yuan, H.T.; Guo, X.; Zhao, M.; Liu, Z.; Yang, T. Path Planning Method With Improved Artificial Potential Field—A Reinforcement Learning Perspective. *IEEE Access* **2020**, *8*, 135513–135523. [CrossRef]
21. Wang, H.; Hao, C.; Zhang, P.; Zhang, M. Mobile robot path planning based on A* algorithm and artificial potential field method. *China Mech. Eng.* **2019**, *30*, 2489–2496.
22. Li, Y.; Wei, W.; Gao, Y.; Wang, D.; Fan, Z. PQ-RRT*: An improved path planning algorithm for mobile robots. *Expert Syst. Appl.* **2020**, *152*, 113425. [CrossRef]
23. Chen, M.; Zhang, Q.; Zhang, G. Research on obstacle avoidance path planning of manipulator under multi-obstacle environment. *Comput. Integr. Manuf. Syst.* **2021**, *27*, 990–998.
24. Long, Z. Virtual target point-based obstacle-avoidance method for manipulator systems in a cluttered environment. *Eng. Optim.* **2019**, *52*, 1–17. [CrossRef]
25. Jia, Z.; Lin, P.; Liu, J.; Liang, L. Online cooperative path planning for multi-quadrotors in an unknown dynamic environment. *Proc. Inst. Mech. Eng. Part G: J. Aeronaut. Eng.* **2021**, *2021*, 09544100211016615. [CrossRef]
26. Zhang, Z.; Wu, D.; Gu, J. A Path-Planning Strategy for Unmanned Surface Vehicles Based on an Adaptive Hybrid Dynamic Step size and Target Attractive Force-RRT Algorithm. *J. Mar. Sci. Eng.* **2019**, *7*, 132.
27. Li, Y.; Xu, D. Cooperative path planning of dual-arm robot based on gravitational adaptive step size RRT. *Robot* **2020**, *42*, 606–616.
28. Shi, Y.; Li, Q.; Bu, S.; Yang, J.; Zhu, L. Research on Intelligent Vehicle Path Planning Based on Rapidly-Exploring Random Tree. *Math. Probl. Eng.* **2020**, *2020*, 5910503. [CrossRef]
29. Yuan, J.; Yang, L.; Tang, X.; Chen, A. Intelligent vehicle motion planning based on improved RRT* and driving trajectory optimization. *Acta Autom. Sin.* **2020**, *46*, 1–10. (In Chinese)