actuators

MDPI

Article

Multi-Objective Point Motion Planning for Assembly Robotic Arm Based on IPQ-RRT* Connect Algorithm

Qinglei Zhang, Haodong Li *, Jianguo Duan, Jiyun Qin and Ying Zhou

Logistics Engineering College, Shanghai Maritime University, Shanghai 201306, China; qlzhang@shmtu.edu.cn (Q.Z.); jgduan@shmtu.edu.cn (J.D.); jyqin@shmtu.edu.cn (J.Q.); zhouying@shmtu.edu.cn (Y.Z.)

* Correspondence: 202130210083@stu.shmtu.edu.cn; Tel.: +86-191-5540-7760

Abstract: Six-axis industrial assembly robotic arms are pivotal in the manufacturing sector, playing a crucial role in the production line. The IPQ-RRT* connect motion planning algorithm for the robotic arm is proposed to improve the assembly process by reducing the time of motion planning and improving the assembly efficiency. The new IPQ-RRT* connect algorithm improves the original PQ-RRT* algorithm applied to UAVs in two dimensions by adding a node-greedy bidirectional scaling strategy. An obstacle detection range is set on the node-greedy bidirectional scaling strategy, in which the existence of obstacles is judged, and different sampling strategies are used according to the judgment results to get rid of obstacles faster, while bidirectional sampling can further improve the operation efficiency of the algorithm. In addition, effective collision detection is realized by combining the hierarchical wraparound box method. Finally, the Bezier curve is utilized to smooth the trajectory of the assembly robotic arm, which improves the trajectory quality while ensuring that the assembly robotic arm does not collide with obstacles. This paper takes the actual assembly process of an intelligent assembly platform as an example and proves the feasibility and effectiveness of the algorithm through simulation experiments and real I5 assembly robotic arm experiments.

Keywords: assembly industrial robotic arm; rapidly exploring random tree; multi-target point path planning

check for updates

Citation: Zhang, Q.; Li, H.; Duan, J.; Qin, J.; Zhou, Y. Multi-Objective Point Motion Planning for Assembly Robotic Arm Based on IPQ-RRT* Connect Algorithm. *Actuators* **2023**, *12*, 459. https://doi.org/10.3390/act12120459

Academic Editor: Zhuming Bi

Received: 15 November 2023
Revised: 7 December 2023
Accepted: 8 December 2023
Published: 9 December 2023

Copyright: © 2023 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

With the advent of Industry 4.0, robotic arms are used in a variety of industries, such as medicine, industrial production, and aerospace [1–3]. At this stage, with the research and application of robotic arms, the efficiency of industrial production has been greatly improved [4–6]. Motion planning is a crucial area of research in the industrial robotic arm field. Over the years, as more industrial robotic arm applications have surfaced, the demand for motion planning of industrial robotic arms has increased to accommodate varied scenario requirements. Among them, the assembly environment is one of several application scenarios, as it involves multiple assembly target points and necessitates that the industrial robotic arm avoids colliding with obstacles in the scene between those target points.

In the field of three-dimensional motion planning for robotic arms, the sampling-based graph search algorithm has been shown to be effective in high-dimensional and complex environments compared to other algorithms [7]. The method includes the rapidly exploring random tree (RRT) [8] and the probabilistic roadmap method (PRM) [9]. Compared to the PRM algorithm, the RRT algorithm does not require the construction of a path map. It can effectively solve the path planning problem of a robotic arm in high dimensions and possesses strong exploration capabilities and a high probability of completeness in high-dimensional space. However, the algorithm's randomness can result in problems of blindness and slow search speeds, as well as insufficient exploration abilities for complex environments.

In response to the limitations of the RRT algorithm, researchers have proposed various solutions. For instance, some have incorporated the greedy algorithm into the RRT algorithm to achieve the RRT-connect dual-tree algorithm. RRT-connect generates two trees from the start and end points, respectively, which speeds up the path searching speed, but in essence, it is still a tree derived from the RRT algorithm, which does not guarantee the optimality of the path and the cost of the tree's growth [10]. Wang et al. proposed a variable step size RRT method, which speeds up the convergence of the RRT algorithm but cannot control the step size of their tree in the workspace [11]. Therefore, Karaman et al., in order to address the poor asymptotic optimality of RRT, added ChooseParent and Rewire optimization modules to optimize the path length by re-selecting the parent node and rewiring. As long as the search time is sufficient, a near-optimal path can be obtained. However, the problem of low search efficiency persists [12]. However, it is undeniable that RRT* is an important milestone in the study of the RRT family of algorithms. The RRT*Smart algorithm adds smart algorithms to the sampling process. The addition of smart algorithms speeds up the convergence of the RRT* algorithm and reduces the path cost, but it also leads to a reduction in the probability of searching for different configurations due to the dependence on the quality of the initial solution, violating the uniform sampling assumption of RRT* [13]. To ensure a better sampling space for generating RRT*, Gammell et al. proposed the Informed-RRT* algorithm, which uses RRT* to solve the initial solution and generates an elliptic state-space region determined by the initial point, the target point, and the length of the path. This algorithm speeds up convergence to the optimal solution as the optimization range shrinks, but it still relies on undirected exploration and struggles to handle complex environments [14]. A fusion algorithm combining potential field and particle swarm optimization (PSO) can predict dynamic obstacles and obtain satisfactory paths [15].

Khatib proposed the artificial potential field (APF) method in 1986 for path planning. Assuming a joint force comprising the obstacle's repulsive force and the goal point's gravitational force, the algorithm guides the robot through the obstacle. However, the method is prone to local minima or oscillations, and there may be instances where the robot cannot reach the goal point [16]. To enhance search efficiency, Qureshi et al. combined RRT* with the artificial potential field (APF) method in an intelligence-based path planning algorithm, resulting in the P-RRT* algorithm. With the addition of APF, the growth of the tree has directionality, enabling faster convergence speed compared to the RRT* algorithm. However, the issue of the artificial field potential method falling into local minima remains unresolved [17].

In addition, Jeong proposed Quick RRT*, which uses trigonometric inequalities to improve the ChooseParent and Rewire procedures, allowing Quick RRT* to have a faster convergence rate compared to RRT* [18].PQ-RRT* combines P-RRT * with Quick-RRT*, allowing the algorithm to produce better initial solutions, which can quickly converge to a relatively optimal solution [19]. Since PQ-RRT* only considers static path planning and does not take path planning in dynamic environments into account, it has limitations. Therefore, Yu et al. improved New Potential Quick-RRT* (NPQ-RRT*) for UAV applications [20]. After completing the path planning, in order to ensure the quality of the path, it is also necessary to optimize the planning path with smoothing [21,22]. Since the paths are optimized in a two-dimensional environment, they have high-dimensional limitations. Guo et al. proposed an optimal B-spline curve to produce smoother and shorter paths, which is particularly suitable for closed paths, and applied to robotic arms [23].

On the basis of the above study, for the problems of low search efficiency and slow convergence speed of RRT series algorithms in robotic arm motion planning, the Improved Potential Quick-RRT* connect (IPQ-RRT* connect) algorithm adds a node-greedy bidirectional scaling strategy on the basis of the PQ-RRT* algorithm. This strategy uses a bidirectional algorithm to scan the obstacles on the map and set an obstacle detection range in the process of random tree sampling, adopting different node sampling methods according to different detection results to realize dynamically coordinated double random

tree growth. In addition, the hierarchical wraparound box method is used to ensure the effectiveness of collision detection of the assembly robotic arm in the assembly process. Finally, the Bezier curve method is used to smooth the trajectory curve to realize the motion planning of the assembly robotic arm.

2. Problem Description

During the assembly process of an assembly robotic arm, it needs to pass through several target points to accomplish different tasks. The following problems need to be solved during the operation of the robotic arm. First of all, a series of target points, as well as task sequencing, should be set up, and the running path of the robotic arm should arrive at these target points in sequence according to the task requirements. In addition, the robotic arm should not collide with obstacles during the running process, and it should keep running smoothly.

Throughout this paper, assume that $\mathbb{R}$ denotes a set of real numbers, $\mathbb{N}$ denotes a set of natural numbers, and $X_{nearest}$ denotes a vector space.

Then, assume that $X \subseteq \mathbb{R}_d$ is a three-dimensional bitmap space, $d \in \mathbb{N}$, $2 \leq d \leq 3$. Assuming that X_{obs} is the obstacle area, the accessible space is denoted as X_{free}. X_{init} and X_{goal} are the initial configuration and target area. Let the continuous function $\sigma : [0, 1] \rightarrow X$, A continuous function σ is called a collision-free path if it is collision-free in a three-dimensional bitmap space.

Problem 1 (Task Guidance). *Tasks need to be sequenced during the assembly process to guide the end-effector of the assembly industrial robotic arm through these target points in order to complete the assembly task.*

Problem 2 (Feasible Path Planning). *Set a triplet* $\left\{X_{init}, X_{obs}, X_{goal}\right\}$*, find a feasible path in three-dimensional bitmap space, unify all feasible paths into a set* $\sum feasible$*, and report success if a feasible path exists in three-dimensional space* X*, otherwise, report failure of path search.*

Problem 3 (Optimal Path Planning). *Set up a triplet* $\left\{X_{init}, X_{obs}, X_{goal}\right\}$ *and a cost function* C *to compute a feasible path* σ^*, $C(\sigma^*) = \min\{c(\sigma^*) \in \sum feasible\}$.

Problem 4 (Fast Path Planning). *Find the optimal path solution in the shortest time* $t \in \mathbb{R}$ *in the same three-dimensional bitmap space.*

Problem 5 (Jitter Reduction). *While satisfying the above issues, the smoothness of the robotic arm path needs to be satisfied to minimize the jitter of the robotic arm.*

3. Methodology

In this paper, we first analyzed the overall assembly task and determined the target points that the assembly robotic arm needs to pass through. The paths between these target points are planned according to the IPQ-RRT* connect algorithm. Then, according to the parameters of the assembly robotic arm, the assembly robotic arm model was optimized using the hierarchical wraparound box method to prevent collisions with obstacles during the operation of the robotic arm. Finally, the paths completed by IPQ-RRT* connect planning were optimized using Bezier curves to make the overall paths smoother and prevent the assembly robotic arm from jittering during movement.

3.1. Assembly Task

According to the assembly process, the assembly flow chart is listed to give a realistic basis for the setting and selection of target points afterward. The task flow chart is shown in Figure 1.

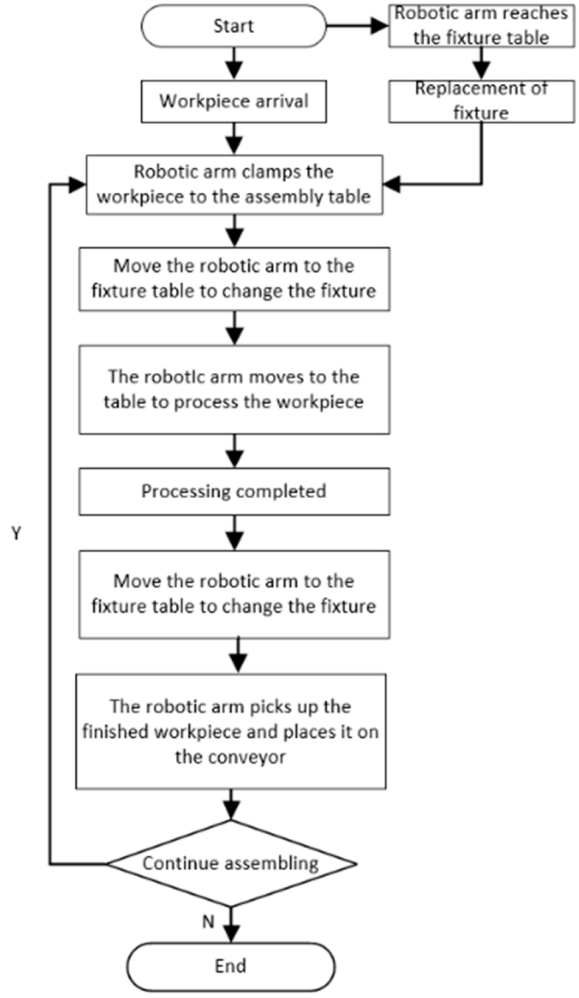


Figure 1. Task flow chart.

From the flowchart shown in Figure 1, take the intelligent assembly test bench as an example; in order to clearly express the tasks to be accomplished by each assembly robotic arm, the total flowchart can be simplified, as shown in Figure 2.

1: The robotic arm moves from the initial position to the fixture table to change the gripper;
2: The robotic arm clamps unassembled workpieces;
3: The robotic arm places the clamped workpiece onto the assembly table;
4: The robotic arm moves to the fixture table to replace the gripper with the screwdriver;
5: The robotic arm moves to the assembly table to complete the assembly of the workpiece;
6: When the assembly is complete, the robotic arm moves to the fixture table to be replaced by the gripper;
7: The robotic arm returns to the assembly table to clamp the processed workpiece;
8: Place the clamped workpiece onto the conveyor and wait for the next task to begin.

Based on the analysis above, we identified four target points, which are the starting points of the conveyor, assembly table, fixture table and assembly robotic arm, as well as the paths that need to be traveled to complete the assembly task. In addition, we determined that only four paths between these four target points need to be planned for motion.

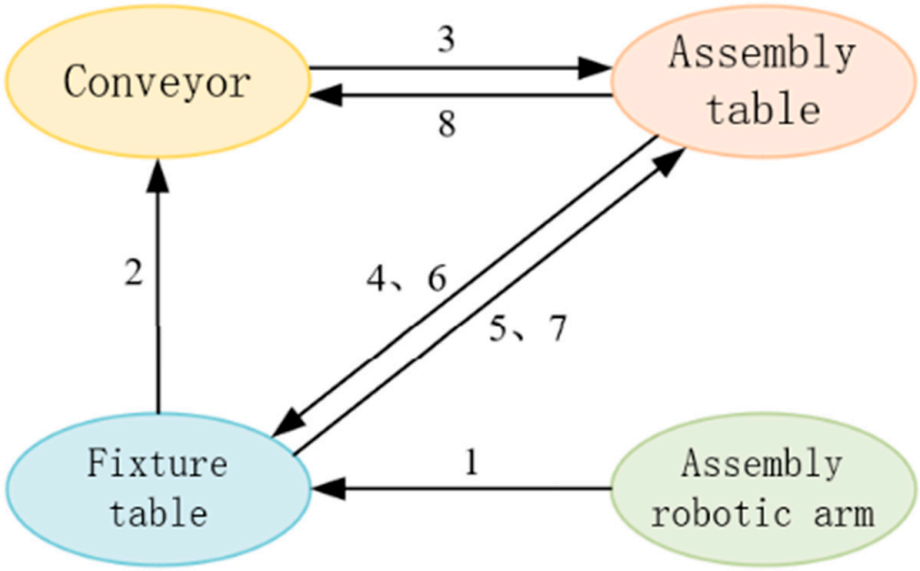


Figure 2. Simplified diagram of the task flow.

3.2. IPQ-RRT Connect Algorithm for Robotic Arms*

The PQ-RRT* algorithm employs a heuristic function to guide path searching and a priority queue to expedite the process. Its primary concept is to convert the search problem into a shortest path problem, followed by utilizing a priority queue to locate the shortest possible route. However, the PQ-RRT* algorithm employs a random and disorderly approach to sampling points, inevitably leading to increased overall running time. IPQ-RRT* connect joins the node-greedy bidirectional scaling strategy, divides the process of algorithmic path search into two parts, joins the obstacle detection function, and samples according to different obstacle detection results with different random sampling methods to get rid of obstacles faster, and reduces the generation of redundant branches to speed up the efficiency of random tree expansion.

3.2.1. PQ-RRT*

The rapidly exploring random tree is a method of generating the rapidly exploring random tree in C space and using random sampling and collision detection to extend its nodes, ultimately finding a collision-free path by constantly backtracking to the parent node of a particular tree node when it reaches the goal point. The PQ-RRT* algorithm adds the objective attraction function RGD and the deep parent node search function to the original RRT* algorithm. The addition of these two functions speeds up the convergence to the optimal solution and produces better initial solutions. The pseudocode for PQ-RRT* algorithm is presented in Algorithm 1.

The PQ-RRT* runs as follows:

(1) First, a random point X_{rand} is selected on the map using the SampleFree function, and then an adjusted random point X_{prand} is obtained by the target attraction function (RGD), under the effect of gravity at the target point. In the RGD function, the NearstestObstacle function is used to compute the Euclidean equation of the distance from X_{prand} to the obstacle X_{obs}, and the distance parameter m represents the number of iterations, d_{obs} denotes the set distance and μ denotes the step size. The pseudocode of RGD is shown in Algorithm 2. Then, the distance evaluation function Nearest is used to return the node $X_{nearest}$ that is positively closest to X_{prand}.

(2) Use the steer function to connect the two points $X_{nearest}$ and X_{prand}, return a line segment σ. After detecting no collisions with obstacles using the CollisionFree function, use the Near function to detect all nodes centered on X_{prand} within a radius r and return a set X_{near}.

(3) Use the Ancestry function to search deeply into the parent nodes of all nodes in X_{near} and return a set of nodes $X_{sparent}$. Then, the ChooseParent function is used to compare the cost of each path and determine the node X_{parent} and the path σ_{parent} based on the result of the comparison.

(4) Finally, the final path diagram G is generated using the Rewire-PQ-RRT* function.

Algorithm 1: PQ-RRT* algorithm.

$PQ-RRT^*$
$1: V \leftarrow \{x_{init}\}; E \leftarrow \phi;$
$2:$ **for** $i = 1$ *to* n **do**
$3:\quad x_{rand} \leftarrow \text{SampleFree}\ (i);$
$4:\quad x_{prand} \leftarrow \text{RGD}\ (x_{rand});$
$5:\quad x_{nearest} \leftarrow \text{Nearest}\ (V, x_{prand});$
$6:\quad \sigma \leftarrow \text{steer}\ (x_{nearest}, x_{prand});$
$7:\quad$ **if** CollisionFree (σ) **then**
$8:\quad\quad X_{near} \leftarrow \text{Near}\ (V, x_{prand}, r);$
$9:\quad\quad X_{parent} \leftarrow \text{Ancestry}\ (G, X_{near});$
$10:\quad\quad (x_{parent}, \sigma_{parent}) \leftarrow \text{ChooseParent}\ (X_{near} \cup X_{sparent}, x_{nearest}, \sigma);$
$11:\quad\quad V \leftarrow V \cup \{x_{rand}\};$
$12:\quad\quad E \leftarrow E \cup \{x_{parent}, x_{rand}\};$
$13:\quad\quad G \leftarrow \text{Rewire} - PQ - RRT^*(G, x_{prand}, X_{near});$
$14:\quad$ **end if**
$15:$ **end for**
$16:$ return $G = (V, E);$

Algorithm 2: RGD function.

$RGD(x_{rand})$
$1: x_{prand} \leftarrow x_{rand}$
$2:$ **for** $n = 1$ *to* m **do**
$3:\quad \vec{F}_{att} \leftarrow (x_{goal} - x_{prand});$
$4:\quad d_{min} \leftarrow \text{NearestObstacle}\ (X_{obs}, x_{prand});$
$5:\quad$ **if** $d_{min} \leq d_{obs}$ **then**
$6:\quad\quad$ **return** $x_{prand};$
$7:\quad$ **else**
$8:\quad\quad x_{prand} \leftarrow x_{prand} + \mu \frac{\vec{F}_{att}}{\left|\vec{F}_{att}\right|};$
$9:\quad$ **end if**
$10:$ **end for**
$11:$ **return** $x_{prand};$

3.2.2. Node-Greedy Bidirectional Scaling Strategy

In motion planning algorithms, the node-greedy bidirectional scaling strategy is a heuristic search strategy commonly used for robotic arm path planning. The purpose of the node-greedy bidirectional scaling strategy for robotic arms is to set up an objective function for the sampling of the sampling points in the sampling process in the three-dimensional search space in conjunction with the dual-tree strategy and use the node-greedy bidirectional scaling strategy to more efficiently find a path to the goal location from the starting point and the end point, respectively, and eventually, use the ConnectTwoTree function to connect the two trees to form a complete path. The pseudocode for the IPQ-RRT* connect algorithm is presented in Algorithm 3.

In the implementation of the algorithm for robotic arm planning, the node-greedy bidirectional scaling strategy using a node-greedy approach includes a function for detecting obstacles. Detecting obstacles during tree growth can be used to avoid unsafe situations such as collision when the robotic arm is in motion, and at the same time, it can also reduce

the time and energy consumption of the robotic arm motion and improve the efficiency of the robotic arm motion.

The role of the node-greedy bidirectional scaling strategy is due to the randomly generated tree nodes being randomly distributed in the map. The generated random points are too scattered, so in order to speed up the generation of the tree, reduce the generation of redundant branches, and guide the tree to get out of the way of obstacles as soon as possible to get around the obstacles, the node-greedy bidirectional scaling strategy is added to the algorithm to improve the search efficiency of the fast search of the random tree. The pseudocode for the Greedynodes function is presented in Algorithm 4. The node-greedy bidirectional scaling strategy is applied to two trees, and the expansion process is divided into two processes in the dual-tree random point picking:

1. When no obstacle is detected in the obstacle detection function Objudge within the d_{ob} range, then the random points X_{rand1} and X_{rand2} are selected as the target point bias point output of X_{goal} and X_{init}, respectively, and the target bias point will guide the two trees to grow towards the start and end points, respectively. The aim is to reduce the generation of unnecessary trees and improve the convergence.
2. When an obstacle is detected within dob by the obstacle detection function Objudge, the random points X_{rand1}. and X_{rand2} select the nodes in the other tree. X_{rand2} and X_{rand1}, respectively, as the target point bias point outputs, and the target bias point guides the expansion of the two trees to the other node, respectively. The aim is to get rid of obstacles faster and reduce the algorithm running time.
3. After going through the above two processes, two new nodes X_{l1} and X_{l2} are output, which are brought into the RGD function for further processing.

Algorithm 3: IPQ-RRT* connect algorithm.

$V_1 \leftarrow \{x_{init}\}$;: $V_2 \leftarrow \{x_{goal}\}$; $E_1 \leftarrow \phi$; $E_2 \leftarrow \phi$;
$G_1 = (V_1, E_1)$; $G2 = (V_2, E_2)$;
for $i = 1$ *to* n **do**
  $x_{\text{rand1}} \leftarrow \text{SampleFree}\ (i)$; $x_{\text{rand2}} \leftarrow \text{SampleFree}\ (i)$;
  $x_{l1}, x_{l2} \leftarrow \text{Greedynodes}\ (x_{rand1}, x_{rand2})$;
  $x_{\text{prand1}} \leftarrow \text{RGD}\ (x_{l1})$; $x_{\text{prand2}} \leftarrow \text{RGD}\ (x_{l2})$;
  $x_{\text{nearest1}} \leftarrow \text{Nearest}\ (V_1, x_{prand1})$; $x_{\text{nearest2}} \leftarrow \text{Nearest}\ (V_2, x_{prand2})$;
  $\sigma_1 \leftarrow \text{steer}\ (x_{\text{nearest1}}, x_{prand1})$; $\sigma_2 \leftarrow \text{steer}\ (x_{\text{nearest2}}, x_{prand2})$;
  if CollisionFree$(\sigma_{1,2})$ **then**
    $X_{\text{near1}} \leftarrow \text{Near}(V_1, x_{\text{prand1}}, r)$; $X_{\text{near2}} \leftarrow \text{Near}\ (V_2, x_{\text{prand2}}, r)$;
    $X_{\text{parent1}} \leftarrow \text{Ancestry}\ (G_1, X_{\text{near1}})$; $X_{\text{parent2}} \leftarrow \text{Ancestry}\ (G_2, X_{\text{near2}})$;
    $(x_{\text{parent1}}, \sigma_{\text{parent1}}) \leftarrow \text{ChooseParent}\ (X_{\text{near1}} \cup X_{\text{sparent1}}, x_{\text{nearest1}}, \sigma_1)$;
    $(x_{\text{parent2}}, \sigma_{\text{parent2}}) \leftarrow \text{ChooseParent}\ (X_{\text{near2}} \cup X_{\text{sparent2}}, x_{\text{nearest2}}, \sigma_2)$;
    $V_1 \leftarrow V_1 \cup \{x_{l1}\}$; $V_2 \leftarrow V_2 \cup \{x_{l2}\}$;
    $E_1 \leftarrow E_1 \cup \{x_{\text{parent1}}, x_{l1}\}$; $E_2 \leftarrow E_2 \cup \{x_{\text{parent2}}, x_{l2}\}$;
    $G_1 \leftarrow \text{Rewire} - \text{IPQ} - \text{RRT}^*\text{Connect}\ (G_1, x_{\text{prand1}}, X_{\text{near1}})$;
    $G_2 \leftarrow \text{Rewire} - \text{IPQ} - \text{RRT}^*\text{Connect}\ (G_2, x_{\text{prand2}}, X_{\text{near2}})$;
    $(\text{T1}, \text{T2}) \leftarrow \text{ConnectTwoTree}\ (G1, G2)$;
  end if
end for
return G_1, G_2

Algorithm 4: Greedynodes function.

Greedynodes (x_{rand1}, x_{rand2})
1 : **for** n = 1 *to* n **do**
2 : **if** Objudge $(x_{rand1}, x_{rand2}, d_{ob})$ **then**
3 : $x_{rand1} \leftarrow x_{goal}$;
4 : $x_{rand2} \leftarrow x_{init}$;
5 : **else**
6 : $x_{rand1} \leftarrow x_{rand2}$;
7 : $x_{rand2} \leftarrow x_{rand1}$;
9 : **end if**
10 : $x_{l1} \leftarrow x_{rand1}$;
11 : $x_{l2} \leftarrow x_{rand2}$;
10 : **end for**
11 : **return** x_{l1}, x_{l2};

3.3. Model and Path Optimization

In the assembly environment, the assembly robotic arm operation process not only needs to ensure that the end-effector of the robotic arm does not collide with the obstacle but also needs to ensure that the connecting rod of the robotic arm does not collide with the obstacle; this paper uses the hierarchical wraparound box method to optimize the model of the robotic arm and the obstacle. In addition, when the IPQ-RRT* connect algorithm is used to complete the path planning, inflection points may appear in the path; these inflection points may lead to some unwanted jitter in the robotic arm during the operation process; in order to solve this problem, this paper adopts the Bezier curve method to optimize the path so that the path remains smooth.

3.3.1. Geometric Modeling of Six-Axis Robotic Arm

The collision detection between the robotic arm and the obstacle is mainly detected between each joint linkage of the robotic arm and the obstacle. In order to simplify the robotic arm model, we use cylinders and spheres to represent the robotic arm model; spheres represent some of the connecting rods, and cylinders represent other connecting rods. The simplified model of the robotic arm is shown in Figure 3. In a real-world environment, a robotic arm may collide with the fixture table, the assembly table, and the portion of a conveyor. This is represented using a cylindrical wraparound box that matches the actual dimensions of these three obstacles. The following fixture table is chosen as an example, as shown in Figure 4.

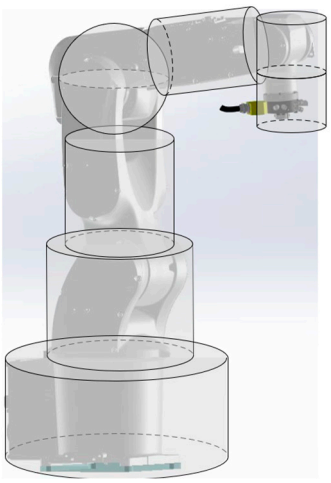


Figure 3. Example diagram of robotic arm modeling.

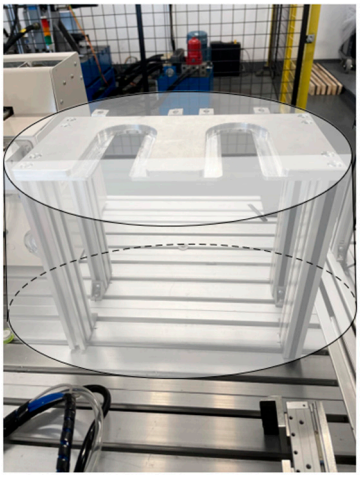


Figure 4. Wraparound box example.

As illustrated in Figure 4, obstacle detection in a robotic arm can be viewed as a distance problem between a sphere and a cylinder, as well as between two cylinders. If the distance between these geometric structures exceeds the sum of their radii, collision can be avoided. Since the collision of the robotic arm primarily occurs between the connecting rod and an obstacle, this paper examines the collision detection algorithm of the robotic arm and obstacle using the calculation of the shortest distance as an illustration. Figure 5 shows an example.

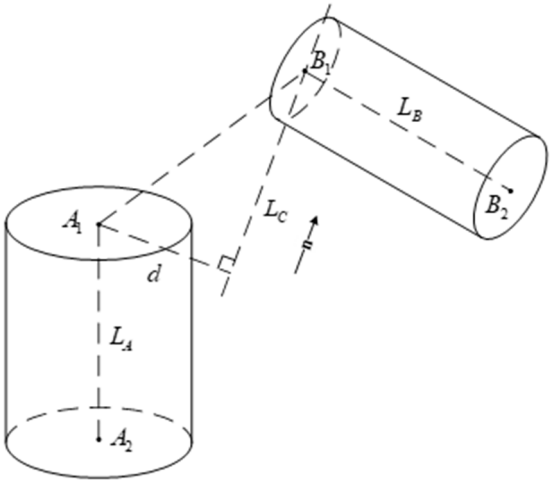


Figure 5. Collision detection diagram.

When the robotic arm is not on the same plane as the obstacle, to determine the distance between the robotic arm's connecting rod and the obstacle, use the neutral lines of the two cylinders, labeled as a and b. This is calculated as follows:

$$d = \frac{\left|\overrightarrow{A_1B_1} \cdot \vec{n}\right|}{\left|\vec{n}\right|} \tag{1}$$

where A_1 and B_1 are the center points of the linkage and obstacle, respectively, which can be calculated using positive kinematics; $\vec{n}$ is the normal vector of the plane formed by L_B and L_C. From the above analysis, it can be seen that the problem of collision detection between the robotic arm linkage and the obstacle can be transformed into the problem of distance calculation from the centerline of the cylinder. The d_{ob} parameter in the Objudge function of the node-greedy bidirectional scaling strategy is used as the distance function.

When $d_{ob} > r_A + r_B$, this indicates that there is no collision between the two cylinders. This ensures that the sampling points are at a safe distance from obstacles during the sampling process. If there is no collision relationship between all of the robotic arm's connecting rod's basic geometric structures and the obstacle, then it can be assumed that the robotic arm will not collide during operation. This method is computationally less intensive, making it suitable for detecting obstacles that may lead to a collision and it has the advantage of using the hierarchical wraparound box algorithm.

3.3.2. Trajectory Optimization

When the completion of the IPQ-RRT* connect algorithm is finished, some inflection points will appear, and the appearance of these inflection points may lead to the jitter problem of the robotic arm during operation, and the appearance of the jitter may lead to the damage of the equipment. In addition, the high degree of freedom joint constraints of the robotic arm may not be satisfied at some inflection points. Therefore, this paper proposes a trajectory optimization scheme by adding the Bezier curve.

The Bezier curve is a parametric curve applied to graphics applications [24]. This is a smooth curve plotted based on the coordinates of four consecutive points on the path. Where P_i is the four curvature control points, $B_{i,n}(t)$ is a Bernstein polynomial and t is a parameter that takes values from 0 to 1. n denotes the order of the curve and i denotes the control point.

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t), t \in [0,1] \tag{2}$$

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, t \in [0,1] \tag{3}$$

Since the IPQ-RRT* connect algorithm runs with a large number of path nodes, the Bezier curve optimized smooth path order is greater than or equal to 3. Therefore, the Bezier curve fitted smooth slip path must have continuous curvature. In addition, since the higher the order of the Bezier curve, the greater the deviation of the optimized smooth curve from the original path, it may happen that the original path, which is originally safe, collides with the obstacle after the Bezier curve optimization. In this paper, by dividing the three control points in the original path into a small segment and performing segmented Bezier curve optimization on the original path, a path that meets the needs of assembly robotic arm motion planning is generated. It is shown in Figure 6.

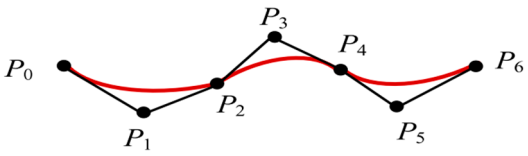


Figure 6. Example diagram of a Bezier curve.

This method of Bezier curve optimization ensures that the optimized path is not shifted too much compared to the original path while ensuring the smoothness of the planned path. It meets the application requirements for assembly robotic arms in assembly environments.

4. Case Applications

In this study, MATLAB 2023b was used as the simulation software. The simulation results of the IPQ-RRT* connect algorithm are compared with the existing RRT*, Informed RRT*, Improved Bi-RRT*, and PQ-RRT* in two two-dimensional maps with different characteristics. It is proved that the IPQ-RRT* connect algorithm has the advantages of fast search speed, shorter search path, and stability.

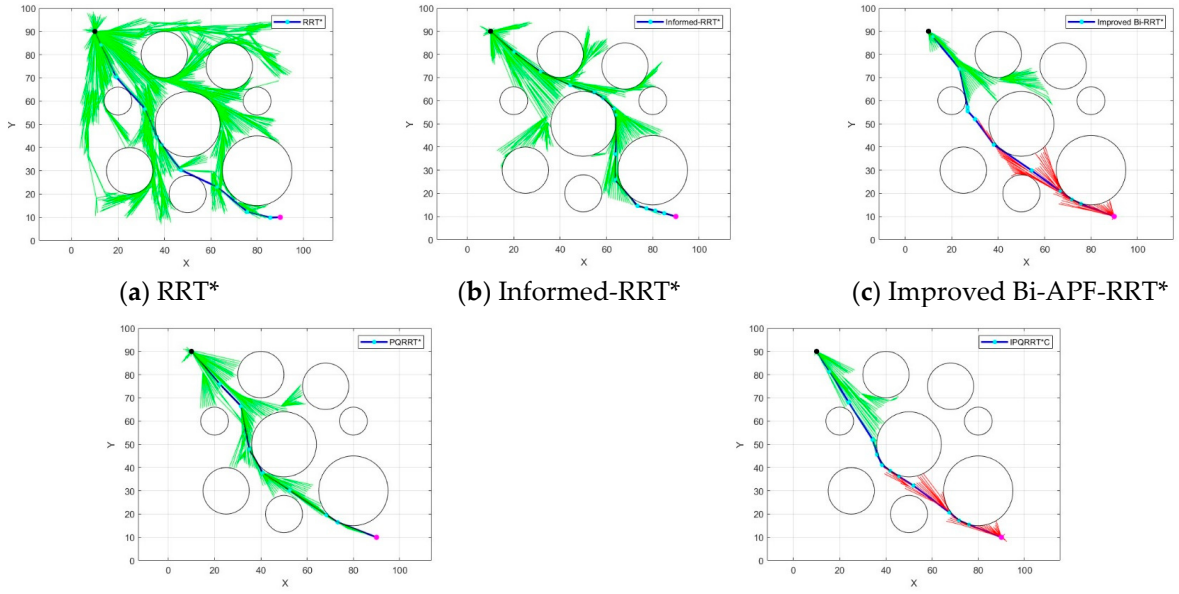
In the three-dimensional environment, the target points to be passed through are specified based on the actual assembly environment. These paths are planned using the IPQ-RRT* connect algorithm. Optimize the planned paths using Bezier curves after planning is complete. After ensuring the smoothness of the paths, the model of the assembled robotic arm is optimized using the hierarchical wraparound box method. Finally, the paths are completed in the simulation environment using the simulated robotic arm.

Finally, the effectiveness of the IPQ-RRT* connect algorithm is verified by completing the assembly task in the real assembly environment.

4.1. Two-Dimensional Simulation

Due to the random nature of the sampling-based path planning algorithms. Each algorithm was run 30 times individually and the algorithm run time, trajectory length, number of iterations and number of failures were recorded. The simulations were implemented on an Intel I5 12500H CPU with 16GRAM. The test environment was the same for all three algorithms, and if a solution could not be found within 200 s, the experiment was directly recognized as a failure.

Two different two-dimensional environments were chosen, both with a map range of 100 mm × 100 mm, and the first map was set up as a unidirectional map with starting coordinates of (10, 90) and target coordinates of (90, 10). The second type of map was set up as a more complex folding line map. The starting point coordinates are (20, 10), and the target coordinates are (80, 90). In the figure, the black point and the rose red point serve as the start and end point, respectively. The blue line is the generated path after the algorithm is run. The green and red lines are the branches generated by the algorithm as it runs from the start and end point, respectively. In the first simulation environment, we set up eight black circular obstacles. The first simulation environment is shown in Figure 7.



(**a**) RRT* (**b**) Informed-RRT* (**c**) Improved Bi-APF-RRT*

(**d**) PQ-RRT* (**e**) IPQ-RRT* connect

Figure 7. Simulation environment (1).

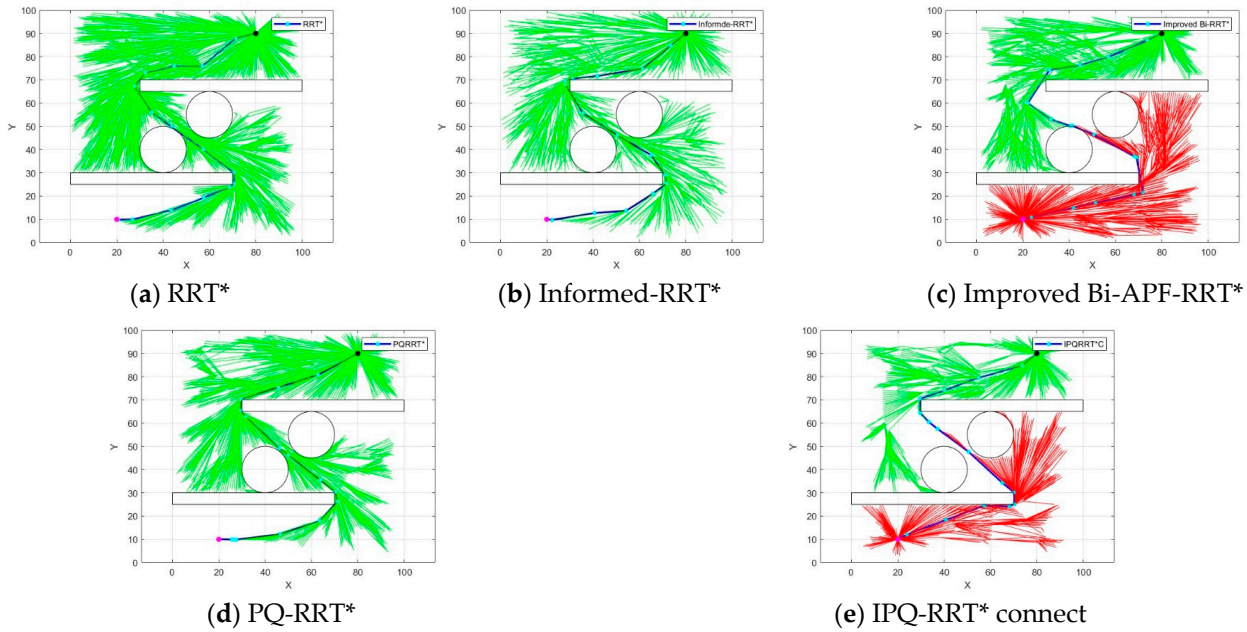
The results of the simulation environment (1) are shown in Figure 7. In Figure 7a, it can be clearly seen that due to the large number of point picking in the environment by RRT*, its final trajectory had more inflection points and longer paths. In Figure 7b, the path length of Informed-RRT* is longer compared to that of RRT*, and the overall path is skewed towards searching above the middle circular obstacle, but its random trees grew faster, the search range was more concentrated, and the time used was shorter. Compared to Informed-RRT*, the Improved Bi-APF-RRT* algorithm in Figure 7c added the obstacle

exclusion effect and reduced some of the redundant points so that the algorithm's search efficiency was improved, but there were still more trees in the algorithm. In Figure 7d, the PQ-RRT* algorithm had a faster search speed compared to the Improved Bi-APF-RRT* algorithm, and the generation of redundant random trees is less. In Figure 7e, the IPQ-RRT* connect algorithm given in this paper is shown, which had a clearer search direction, generated fewer branches and shorter paths, and met the working requirements of the robotic arm. The simulated data in the environment (1) are shown in Table 1.

Table 1. Simulation environment (1) algorithm comparison table.

Algorithm	Average Path Length (mm)	Average Running Time (s)	Average Iterations	Number of Failures
RRT*	132.74	17.91	296	0
Informed-RRT*	140.22	12.22	272	0
Improved Bi-APF-RRT*	130.01	9.26	232	0
PQ-RRT*	124.26	7.34	191	0
IPQ-RRT* connect	117.76	0.28	101	0

Analysis of Table 1 shows that the IPQ-RRT* connect algorithm trajectory length was reduced by 5.24% compared to the pre-improved PQ-RRT* algorithm trajectory length. Compared to the Improved Bi-APF-RRT* algorithm, the average running time and the number of iterations were reduced by 96.97% and 56.46%, respectively. The results show that the algorithm in this paper improved the search speed and significantly reduced the generation of redundant tree branches in unidirectional search maps, also proving the stability of the algorithm. In the second simulation environment, we set up two rectangular black obstacles and two circular black obstacles. The second simulation environment is shown in Figure 8.



(**a**) RRT* (**b**) Informed-RRT* (**c**) Improved Bi-APF-RRT*

(**d**) PQ-RRT* (**e**) IPQ-RRT* connect

Figure 8. Simulation environment (2).

Facing a more complex environment (2), the Informed-RRT* algorithm and Improved Bi-APF-RRT* experienced search failures. The IPQ-RRT* connect algorithm in this paper guarantees a shorter search time and also guarantees a shorter path length. The simulated data in the environment (2) are shown in Table 2.

Table 2. Simulation environment (2) algorithm comparison table.

Algorithm	Average Path Length (mm)	Average Running Time (s)	Average Iterations	Number of Failures
RRT*	194.26	17.09	963	0
Informed-RRT*	187.76	20.22	821	5
Improved Bi-APF-RRT*	192.56	16.27	643	2
PQ-RRT*	186.72	11.29	598	0
IPQ-RRT* connect	174.88	2.33	442	0

Analyzing the above table shows that the IPQ-RRT* connect algorithm trajectory length was reduced by 6.34% compared to the PQ-RRT* algorithm trajectory length before improvement. The average running time and number of iterations were reduced by 85.67% and 31.25%, respectively, compared to the Improved Bi-APF-RRT* algorithm.

In summary, it can be seen that the IPQ-RRT* connect algorithm can maintain a short search time and good path quality, whether facing the simpler unidirectional environment or the more complex folded line environment.

4.2. Three-Dimensional Simulation

Based on the analysis in Section 3.1, the four goal points and the paths that need to be path planned can be identified. We assume that the robotic arm end-effector position is P0, the fixture table position is set to P1, the conveyor belt position is set to P2, and the assembly table position is set to P3. We determine the coordinates of each target point in the simulation environment based on the actual position coordinates of the real assembly platform. The real assembly environment is shown in Figure 9.

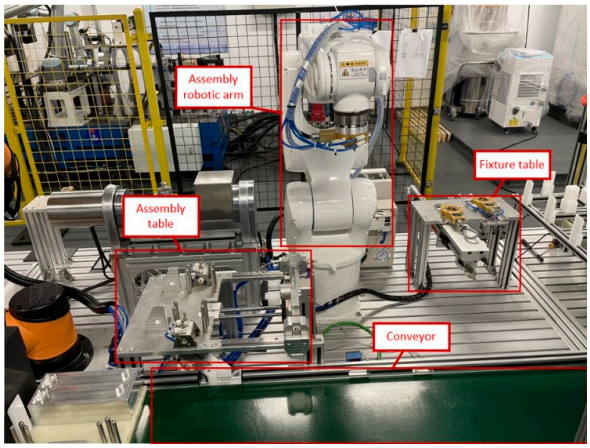


Figure 9. Realistic assembly experiment platform.

We set up a 1000 mm × 1000 mm × 1000 mm map, and in order to test the obstacle avoidance ability of the IPQ-RRT* connect algorithm in a three-dimensional environment, we set up six virtual spherical obstacles with different radii in the three-dimensional map. In addition, we set the exact position coordinates according to the target point P0 of the end-effector of the assembly robotic arm, the target point P1 of the fixture table, the target point P2 of the conveyor, and the target point P3 of the assembly table. The spatial coordinates of P0 are (500, 0, 600), the spatial coordinates of P1 are (0, 789.2, 100), and the spatial coordinates of P2 are (907.6, 612.5, 100), and the P3 space coordinates are (702.4, −200.6, 100). In the map, black dot denote the target point P1, blue dot denote the target point P2, rose-red dot denote the target point P3, and yellow dot denote the target point P4. Blue line denotes the path after the algorithm is run. The red and green lines indicate the branches generated during the running of the algorithm. Yellow spheres indicate obstacles.

The path planned by the IPQ-RRT* connect algorithm is shown in Figure 10a. It can be found that there are many inflection points in the planning path. We used the Bezier curve method for optimization, and the optimized path is shown in Figure 10b. It can be found that the optimized path is guaranteed to be smooth.

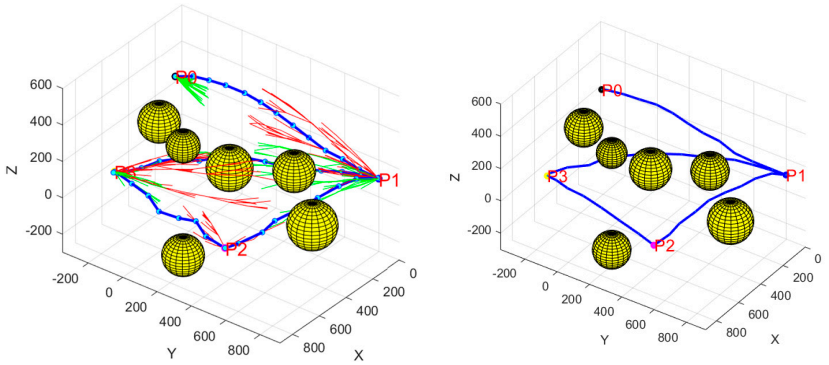


Figure 10. IPQ-RRT* connect algorithm in three-dimensional operation.

In the simulation robotic arm experiment, to ensure the realism of the simulation environment, the robotic arm simulation model is modeled according to the DH parameters of the real robotic arm. Inverse kinematic solution paths for each intermediate path point of the robotic arm running path using the ikine function in the Robotic Toolbox tool of MATLAB 2023b software. In the simulation map, several spherical obstacles were set to verify the obstacle avoidance ability of the robotic arm end-effector. In addition, to verify the obstacle avoidance of the linkage of the robotic arm, the assembly table, the fixture table, and the conveyor belt were set up as obstacles using the hierarchical wraparound box method in Section 3.2.1 with cylindrical wraparound boxes. This visualized the collision and accuracy of the robotic arm. The joint angles were also recorded to check whether the robotic arm jerked during operation. In the map, red dots indicate target points. The blue line indicates the robotic arm end-effector run path. Yellow spheres and green cylinders indicate obstacles.The simulation results of the robotic arm are shown in Figure 11.

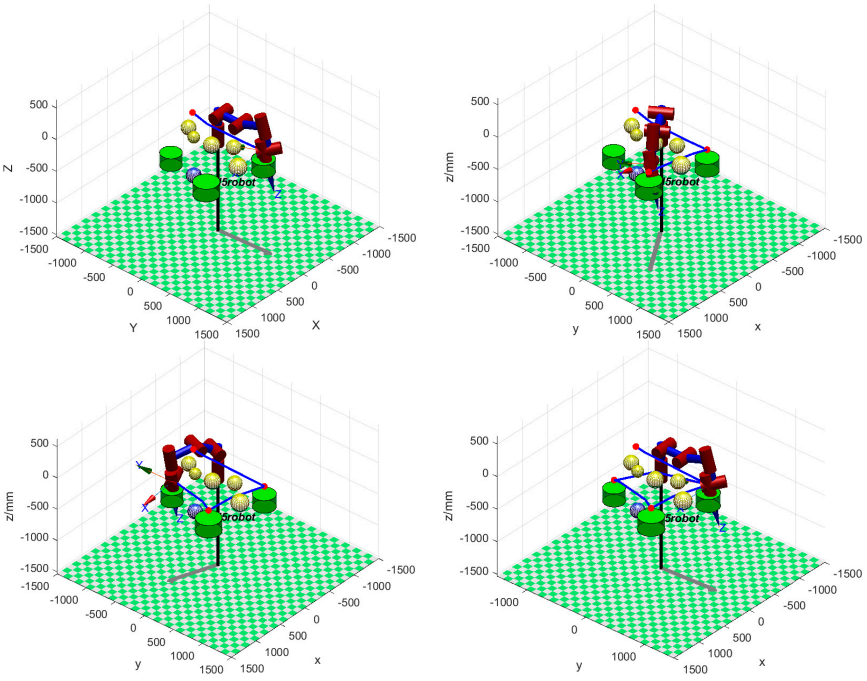


Figure 11. Robotic arm simulation.

It can be seen from Figure 11 that the simulated robotic arm yields four collision-free and smoother paths after running the simulation. In the actual assembly process, the robotic arm end-effector only needs to follow these four collision-free paths. Under the premise of ensuring safe and fast paths, the overall efficiency of the assembly task is improved.

The joint angles of the robotic arm in the four-segment path are shown in Figure 12.

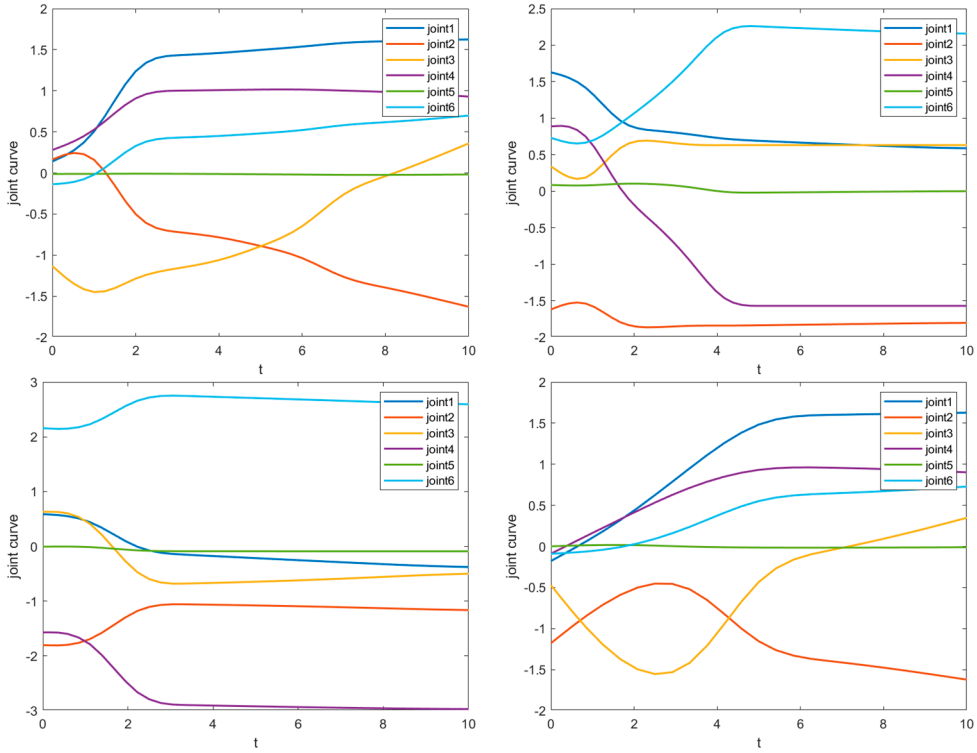


Figure 12. Angle diagram of each joint.

From the analysis of the robotic arm simulation diagram, it can be seen that the end-effector and linkage did not collide with the obstacles during the operation of the robotic arm. From the analysis of the angle diagram of each joint, it can be seen that no inflection point was found in the curve of any joint. It can be demonstrated that the hierarchical wraparound box method used in the robotic arm model effectively avoided the collision of the robotic arm linkage with obstacles. The application of the Bezier curve method ensures that the generated paths are sufficiently smooth. It also avoided jittering of the robotic arm and allowed the robotic arm to run smoothly.

4.3. Real Environment Experiment

In this study, the I5 robotic arm was used to conduct experiments in a real environment. The end-effector of the robotic arm is located at a certain position in space, so it can have many possible attitudes. Since the end-attitude constraint of the robotic arm is not the focus of this paper, we chose two reasonable attitudes as the working attitudes of the robotic arm, i.e., vertical and horizontal. The robotic arm operated in a real environment, as shown in Figure 13. The process of each assembly is explained in the red boxes in the figure.

As shown in Figure 13a, when the workpiece reaches the specified position, the robotic arm reaches the fixture table to change the gripper; As shown in Figure 13b, the robotic arm clamps the unassembled workpiece; As shown in Figure 13c, the robotic arm moves to the assembly table and places the clamped workpiece to the assembly table; As shown in Figure 13d, the robotic arm moves to the fixture table, places the gripper to the fixture table and replaces the screwdriver machine; As shown in Figure 13e, the robotic arm moves to the assembly table and uses the screwdriver machine to assemble the workpiece; As shown

in Figure 13f, after replacing the gripper, the robotic arm places the already assembled workpiece onto the conveyor. The task of assembling the workpiece is finally completed.

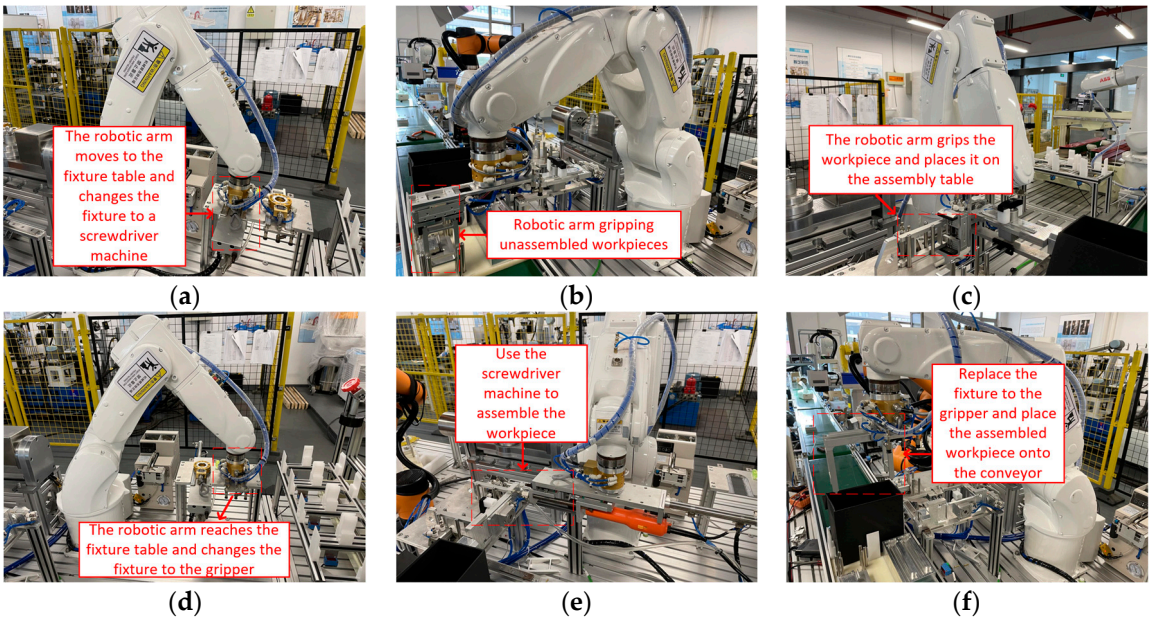


Figure 13. Robotic arm operating in real environment.

During the operation of the robotic arm, the robotic arm operated smoothly without sudden changes. Finally, the assembly task was completed.

5. Conclusions

This study presents an IPQ-RRT* connect motion planning algorithm for assembly environments. The algorithm improves the original PQ-RRT* algorithm applied to two-dimensional UAVs. A node-greedy bidirectional scaling strategy is added to address the inefficiency caused by unidirectional random sampling in the PQ-RRT* algorithm. The node-greedy bidirectional scaling strategy sets an obstacle detection range under the premise of bidirectional sampling, and different sampling strategies are used according to the results of the detection range. The IPQ-RRT* connect algorithm has the advantages of fast searching speed, short searching path, and good stability. In addition, considering the linkage collision and jitter problem that may occur during the operation of the robotic arm, the hierarchical wraparound box method and the Bezier curve method are added. The superiority of the IPQ-RRT* connect algorithm is verified by the results in two-dimensional simulation and three-dimensional simulation. In the actual experiment, the robotic arm passes through multiple target points, moves quickly and smoothly between these target points, and finally completes the assembly task.

Author Contributions: Conceptualization, Q.Z. and H.L.; methodology, H.L.; software, Q.Z. and J.D.; validation, Q.Z., J.Q. and Y.Z.; formal analysis, H.L. and J.D.; investigation, H.L.; resources, H.L.; data curation, Q.Z. and H.L.; writing—original draft preparation, H.L.; writing—review and editing, J.Q.; supervision, J.D.; project administration, Q.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data will be made available upon request.

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

References

1. Vestermark, G.L.; Bhowmik-Stoker, M.; Springer, B.D. Cognitive Training for Robotic Arm-Assisted Unicompartmental Knee Arthroplasty through a Surgical Simulation Mobile Application. *J. Knee Surg.* **2019**, *32*, 984–988. [CrossRef] [PubMed]
2. 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]
3. Sathish Kumar, A.; Naveen, S.; Vijayakumar, R.; Suresh, V.; Asary, A.R.; Madhu, S.; Palani, K. An Intelligent Fuzzy-Particle Swarm Optimization Supervisory-Based Control of Robot Manipulator for Industrial Welding Applications. *Sci. Rep.* **2023**, *13*, 8253. [CrossRef] [PubMed]
4. Lei, T.; Rong, Y.; Wang, H.; Huang, Y.; Li, M. A Review of Vision-Aided Robotic Welding. *Comput. Ind.* **2020**, *123*, 103326. [CrossRef]
5. Lei, T.; Wang, W.; Rong, Y.; Xiong, P.; Huang, Y. Cross-Lines Laser Aided Machine Vision in Tube-to-Tubesheet Welding for Welding Height Control. *Opt. Laser Technol.* **2020**, *121*, 105796. [CrossRef]
6. Wu, C.; Hu, J.; Lei, T.; Yang, P.; Gu, S. Research on Robust Laser Vision Feature Extraction Method for Fillet Welds with Different Reflective Materials under Uncertain Interference. *Opt. Laser Technol.* **2023**, *158*, 108866. [CrossRef]
7. Zammit, C.; van Kampen, E.-J. Comparison Between A* and RRT Algorithms for 3D UAV Path Planning. *Unmanned Syst.* **2022**, *10*, 129–146. [CrossRef]
8. Rapidly-Exploring Random Trees: A New Tool for Path Planning | CiNii Research. Available online: https://cir.nii.ac.jp/crid/1573950399665672960 (accessed on 4 September 2023).
9. Wang, C.; Meng, M.Q.-H. Variant Step Size RRT: An Efficient Path Planner for UAV in Complex Environments. In Proceedings of the 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR), Angkor Wat, Cambodia, 6–10 June 2016; pp. 555–560.
10. Kuffner, J.J.; LaValle, S.M. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In Proceedings of the Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; Volume 2, pp. 995–1001.
11. Xu, J.; Tian, Z.; He, W.; Huang, Y. A Fast Path Planning Algorithm Fusing PRM and P-Bi-RRT. In Proceedings of the 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), Jinan, China, 23–25 October 2020; pp. 503–508.
12. Sampling-Based Algorithms for Optimal Motion Planning—Sertac Karaman, Emilio Frazzoli. 2011. Available online: https://journals.sagepub.com/doi/abs/10.1177/0278364911406761 (accessed on 4 September 2023).
13. Islam, F.; Nasir, J.; Malik, U.; Ayaz, Y.; Hasan, O. RRT*-Smart: Rapid Convergence Implementation of RRT* towards Optimal Solution. In Proceedings of the 2012 IEEE International Conference on Mechatronics and Automation, Chengdu, China, 5–8 August 2012; pp. 1651–1656.
14. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Informed RRT*: Optimal Sampling-Based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2997–3004.
15. Mandava, R.K.; Bondada, S.; Vundavilli, P.R. An Optimized Path Planning for the Mobile Robot Using Potential Field Method and PSO Algorithm. In *Soft Computing for Problem Solving*; Bansal, J.C., Das, K.N., Nagar, A., Deep, K., Ojha, A.K., Eds.; Springer: Singapore, 2019; pp. 139–150.
16. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots—Oussama Khatib. 1986. Available online: https://journals.sagepub.com/doi/abs/10.1177/027836498600500106?journalCode=ijra (accessed on 4 September 2023).
17. Qureshi, A.H.; Ayaz, Y. Potential Functions Based Sampling Heuristic for Optimal Path Planning. *Auton. Robot.* **2016**, *40*, 1079–1093. [CrossRef]
18. 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]
19. 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]
20. Yu, Z.; Xiang, L. NPQ-RRT*: An Improved RRT* Approach to Hybrid Path Planning. *Complexity* **2021**, *2021*, 6633878. [CrossRef]
21. Song, B.; Wang, Z.; Zou, L. An Improved PSO Algorithm for Smooth Path Planning of Mobile Robots Using Continuous High-Degree Bezier Curve. *Appl. Soft Comput.* **2021**, *100*, 106960. [CrossRef]
22. Celestini, D.; Primatesta, S.; Capello, E. Trajectory Planning for UAVs Based on Interfered Fluid Dynamical System and Bézier Curves. *IEEE Robot. Autom. Lett.* **2022**, *7*, 9620–9626. [CrossRef]
23. Kuo, Y.-L.; Lin, C.-C.; Lin, Z.-T. Dual-Optimization Trajectory Planning Based on Parametric Curves for a Robot Manipulator. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 172988142092004. [CrossRef]
24. FAST RRT* 3D-Sliced Planner for Autonomous Exploration Using MAVs | Unmanned Systems. Available online: https://www.worldscientific.com/doi/abs/10.1142/S2301385022500108 (accessed on 4 September 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.