machines

MDPI

Article

Trajectory Planning of Dual-Robot Cooperative Assembly

Xuyang Chen, Xiaojun You, Jinchao Jiang, Jinhua Ye and Haibin Wu *

School of Mechanical Engineering and Automation, Fuzhou University, Fuzhou 350108, China
* Correspondence: wuhb@fzu.edu.cn; Tel.: +86-1895-049-2523

Abstract: Efficiency can be improved through the cooperation of a dual-robot during assembly. However, how to effectively plan a simple and smooth path in a dynamic environment is a prominent problem in the process of dual-robot cooperative assembly. In this paper, a method based on RRT-Connect algorithm for trajectory planning and post-processing for trajectory optimization is proposed. This method takes full advantage of the excellent solution ability of RRT-Connect algorithm in the complex environment so as to obtain the initial path successfully. Through post-processing, the problem of RRT-Connect non-convergence to target is optimized. We use two 6-DOF industrial robots to build an experimental platform and design a dual-robot cooperative assembly system. According to the given task, the system can generate the original collision-free path based on RRT-Connect algorithm. Then the original path is simplified by Floyd algorithm and smoothed by multi-segment Bezier curve. Finally, the time parameter is sequenced for all the path points based on the iterative method, and the effective trajectory is obtained. The experimental results show that the algorithm proposed in this paper can effectively plan and optimize the trajectory of dual-robot. Compared to other methods, this approach has a higher success rate and less planning time.

Keywords: dual-robot cooperation; assembly; trajectory planning; trajectory optimization

check for updates

Citation: Chen, X.; You, X.; Jiang, J.; Ye, J.; Wu, H. Trajectory Planning of Dual-Robot Cooperative Assembly. *Machines* **2022**, *10*, 689. https://doi.org/10.3390/machines10080689

Academic Editor: Dan Zhang

Received: 6 July 2022
Accepted: 9 August 2022
Published: 13 August 2022

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

Copyright: © 2022 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 the time of high intelligence, the application of robots in industrial production has become more and more extensive [1]. Using robots to achieve assembly automation can improve production efficiency, reduce production costs, and ensure product quality [2,3]. However, a single-arm industrial robot is difficult to complete complex assembly tasks, so additional equipment is usually required to assist the robot [4]. Compared with single-robot assembling system, dual-robot cooperation can complete complex assembly tasks more efficiently. Dual or more industrial robot cooperation is the main development trend in the future [5–8].

At present, there are some dual-robot system products, such as PR2 [9] and YuMi [10], which are the important equipment used for research and experiments. The trajectory planning and optimization are difficult in the process of dual-robot cooperative assembly. Due to the multi degrees of freedom of dual-robot and the coupling relationship between time and space in the movement of different arms, trajectory planning method of dual-robot is very different from that of a single arm robot [11]. How to plan a collision-free trajectory in a dynamic environment, and ensure that the trajectory is accurate, efficient, and smooth, have become the key research content in the dual-robot cooperative assembly.

Some scholars have extended the research on multi-robot motion planning by using vision method [12], grid method [13], artificial potential field method [14,15] and other methods. Chuang et al. [16] proposed a dual-robot alternate priority planning algorithm based on artificial potential field, which uses the iterative method to continuously plan local paths until each arm reaches the target. Sunil et al. [17] studied the trajectory planning of free-floating dual-arm manipulator by using the algorithm based on position kinematics equation and iterative search. Sezgin et al. [18,19] introduced control configuration points (CCPs) to take advantage of redundancy when using multi-plane robots. For obstacle

avoidance, Liu et al. [20] used CCPs to develop a method for planning the trajectory of multiple robot fingers when folding cartons.

However, in the high-dimensional space and complex working environment, the traditional path planning algorithm has the problems of large calculation amount and complex model construction. Instead, motion planning method based on random sampling shows unique advantages in solving the problem, such as Probabilistic Roadmaps (PRM) [21], Rapidly-exploring Random Trees (RRT) [22], and their variants [23–26]. Among them, the reduced query time and memory requirements make RRT become the current main algorithm for dual-robot cooperative motion planning. However, the path generated by the basic RRT is poor quality and lack of repeatability due to randomness. The fast convergence and asymptotic optimization are important exploration fields for RRT. For example, Informed RRT* [27] can reduce the computational burden by defining the sampling domain as a prolate hyper-ellipsoid. Batch Informed Trees (BIT*) [28] which is based on a unified graph and sampling-based planning technology can also reduce path costs. Nevertheless, it remains a challenge to strike a balance between path quality and planning efficiency in high-dimensional space. Indeed, in practical real-time applications, obtaining a high-quality solution quickly is of more interest than achieving an optimal solution. Part of the focus of this paper will be on this issue.

In fact, RRT-Connect [29] has a high success rate in solving dual robots in complex environments. By properly post-processing the sampling path, we can quickly obtain a high-quality path which optimizes the problem that RRT-Connect does not converge to the target. Without loss of generality, and in order to increase operating space and load capability, this paper builds a dual-robot platform based on two independent industrial robots ABB IRB120, and discusses the trajectory planning of dual-robot cooperation for disassembling the wire from terminal. The main research contents and results are summarized as follows:

(1) An overall scheme of the dual-robot cooperative assembly system is designed. For the trajectory planning problem, a trajectory planning hierarchy based on ROS (Robot Operating System) is proposed.
(2) The RRT-Connect algorithm is uesd for the motion planning of the dual-robot, and the original collision-free path is generated.
(3) The planned path is further simplified and smoothed. In addition, then time parameter is matched to the path points. Finally, the optimized joint angle data of the dual-robot with time parameter is obtained.
(4) Through experiments, this paper verified the feasibility of the trajectory planning method in complex environments. The proposed algorithms are used for actual dual-robot to disassemble the wire from terminal.

2. Trajectory planning Frame

2.1. Dual-Robot Cooperative Assembly System

The structure diagram of the dual-robot cooperative assembly system designed in this paper is shown in Figure 1. As the controllers of usual industrial robots are mostly not an open-source system, it is unsuitable to develop the trajectory planning algorithm directly on the robot controllers. Therefore, a main computer is needed, and a system with two sub-controllers of dual-robot is formed.

In the main computer, there are many software development platforms for robot systems, such as MATLAB, RobotStudio (only for ABB robot), Player/stage, ROS, etc. Among them, ROS is widely used worldwide because of its openness and rich functions. In addition, the offline programming interface, scene visualization interface and motion planning module are realized based on ROS. Specifically, the user inputs the task into the offline programming interface through specific instructions, which is implemented through Qt Creator. The scene visualization interface implemented by the visualization tool RViz is used to display the robot and its working environment to the user. The motion planning module which based on the forward and inverse solutions of robot kinematics can calculate the relationship between Cartesian space and joint space.

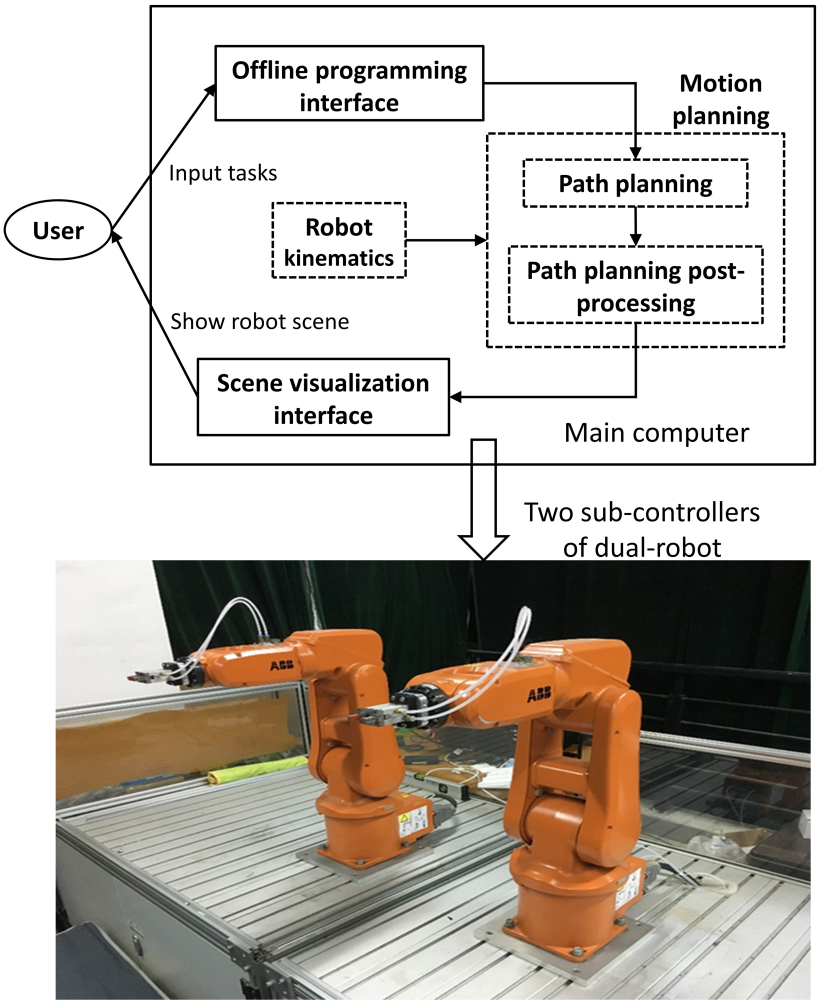


Figure 1. Dual-robot collaborative assembly system.

Finally, the generated joint trajectory can be displayed on the visual interface RViz to simulate the task execution process of the dual-robot. On the other hand, it can be sent to the sub-controllers. In addition, every robot controller will convert the trajectory into actual control instructions so that it can drive the robot arm and end effector to perform specific movements.

2.2. Trajectory Planning Hierarchy

In order to solve the problem of trajectory planning in the process of dual-robot cooperative assembly, a theoretical hierarchical framework for dual-robot cooperative trajectory planning is designed. As shown in Figure 2, it is divided into 5 layers, from top to bottom including user layer, planning layer, transmission layer, communication layer, and execution layer.

The user layer includes the human-computer interface and virtual motion scene. The human-computer interface provides the user with a task input window. Users should first input the robot model, and then set the position of objects and obstacles according to the actual scene. These models can be displayed in virtual motion scene by using RViz.

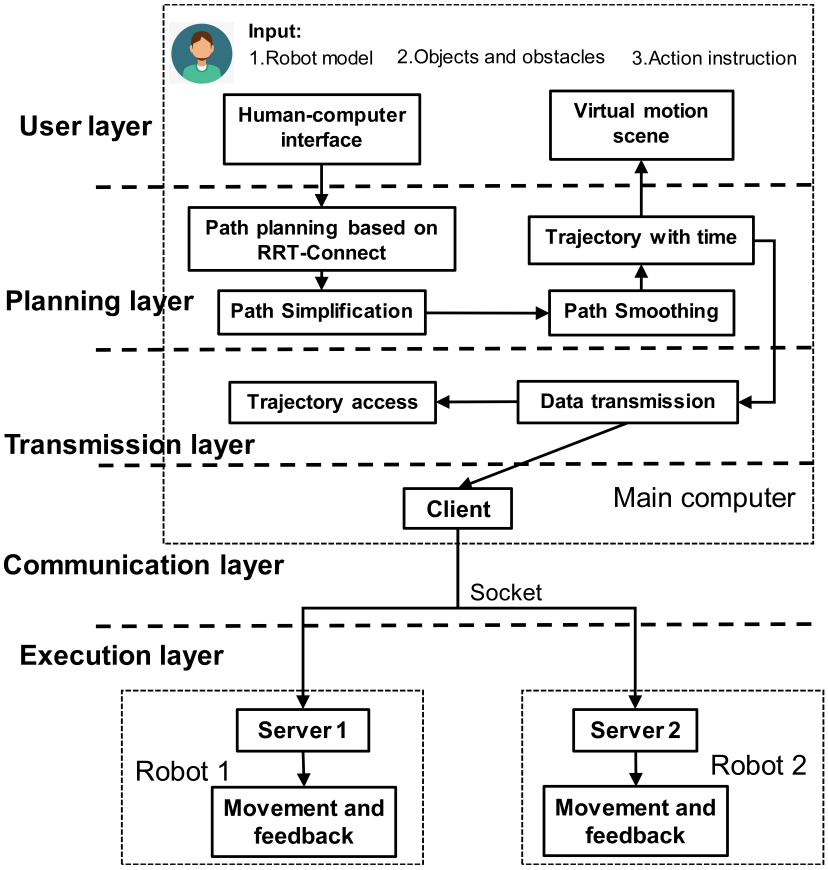


Figure 2. Dual-robot collaborative trajectory planning hierarchy.

This paper also designed several action instructions for users to input. Each action contains the respective action instructions of the two robots. The action instructions are shown in Table 1. Here, p represents the pose of the target point, t represents the pause time, **a** and **r**, respectively, represent the approach vector and exit retreat when picking or placing the object.

Table 1. Action instructions for planning messages.

Instructions	Meaning	Function
Reach	Move to the target pose according to the trajectory	$Reach(p)$
Linear	Move to the target pose according to the linear trajectory	$Linear(p)$
Hold	Maintains current state	$Hold(t)$ or $Hold()$
Pick	Pick the object	$Pick(p, \mathbf{a}, \mathbf{r})$
Place	Place the object	$Place(p, \mathbf{a}, \mathbf{r})$

In order to better understand, as shown in Figure 3, the disassembling wire from terminal is taken as an example. In each of actions, the two robots start to move at the same time. An example of the dual-robot action instructions is shown in Table 2.

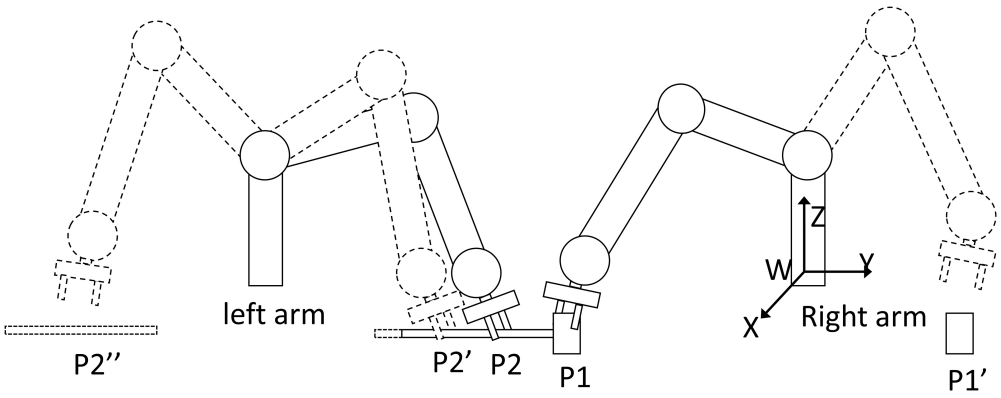


Figure 3. Schematic diagram of dual-robot cooperative disassembling for terminal with wire.

Table 2. Example of action instructions.

Number	Left Robot	Right Robot
1	$Pick(P2, \mathbf{V}, \mathbf{0})$	$Pick(P1, \mathbf{V}, \mathbf{0})$
2	$Linear(P2')$	$Hold()$
3	$Place(P2'', \mathbf{0}, -\mathbf{V})$	$Place(P1', \mathbf{0}, -\mathbf{V})$

In the first set of actions, the left and right robots, respectively, pick the wire and terminal. $\mathbf{V} = (0, 0, 10)$ is the direction vector which means 10mm translation along the $+z$ axis. The left robot adjusts its posture to be the same as $P1$, and then extends along $\mathbf{V}$ to point $P2$. After clamping the wire, the left robot remains motionless, so the retreat vector is set to $\mathbf{0}$. The right robot picks up the terminal with the same action.

In the second set of actions, the left robot moves along a straight trajectory to pull out the wire and move the wire from $P2$ to $P2'$. At the same time, the right robot remains stationary.

In the third set of actions, the left robot places the wire from $P2'$ to $P2''$. There is no need to place the object in a specified direction, so the approach vector is set to $\mathbf{0}$. In addition, then the left robot moves along the $-\mathbf{V}$ vector after placing the object. The action of the right robot is the same, and the whole process of disassembling terminal with wire is finally finished.

After the user inputs tasks, the planning layer will perform motion planning according to these actions. RRT-Connect algorithm is used to get the original path. Then the path need to be simplified and smoothed. At last, the time parameter is sequenced for all the path points to obtain the effective trajectory.

The transmission layer will transmit trajectory data to the communication layer. In addition, it can save the trajectory data as a text file for next use.

The communication layer uses Socket to realize the information interaction between the main computer and the sub-controllers. The main computer acts as the Socket client, and the sub-controllers of dual-robot act as servers.

The execution layer is responsible for receiving and processing information from the main computer. It can also feed back the processing results to communication layer.

3. Motion Planning Algorithm for Dual-Robot

The motion planning of the dual-robot is carried out in high-dimensional space, and the two arms are dynamic obstacles to each other. Therefore, this paper adopts the random sampling motion planning method and develops the motion planning analysis based on the RRT algorithm [22].

In this paper, RRT-Connect algorithm [29] is selected for initial path planning of dual-robot. RRT-Connect algorithm can improve the node expansion mode of RRT algorithm.

The schematic diagram of node expansion is shown in Figure 4 and the pseudo code of RRT-Connect is shown in Algorithm 1. It establishes two random search trees based on the starting point q_{start} and the target point q_{goal}. In each round of expansion, one tree for sampling expansion and the other tree for directional expansion are specified.

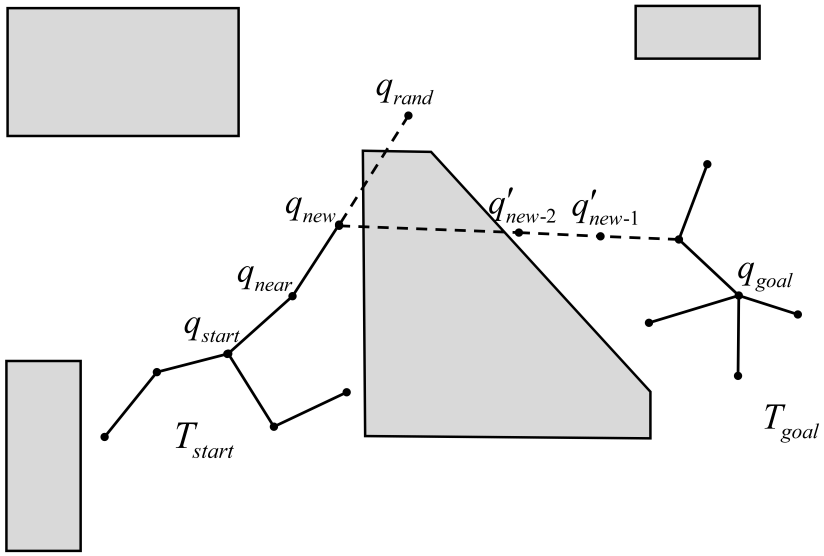


Figure 4. Node expansion of RRT-Connect algorithm.

Algorithm 1 RRT-Connect

Input: $T_{start} \Leftarrow q_{start}$, $T_{goal} \Leftarrow q_{goal}$
Output: T_{path}

while $t < T$ **do**
  $q_{rand} \Leftarrow Sample()$, $q_{near} \Leftarrow Nearest(T_{start}, q_{rand})$, $q_{new} \Leftarrow Generate(q_{rand}, q_{near}, d_m)$
  if q_{new} is in Q_{free} **then**
    $T_{path} \Leftarrow q_{new}$, $q'_{near} \Leftarrow Nearest(T_{goal}, q_{new})$, $q'_{new} \Leftarrow Generate(q_{new}, q'_{near}, d_m)$
    if $q'_{new}$is in Q_{free} **then**
      $T_{path} \Leftarrow q'_{new}$
      if $q'_{new} = q_{new}$ **then**
        return T_{path}
      end if
    else
      $Swap(T_{start}, T_{goal})$
    end if
  else
    $Swap(T_{start}, T_{goal})$
  end if
end while

Within the rated planning time T, firstly, the random tree for sampling expansion samples a point q_{rand} in the free space Q_{free}. Secondly, we need to find the node nearest to q_{rand} in random tree T_{start} and denoted as q_{near}. Thirdly, a new point q_{new} needs to be generated by the following principles. If the distance between q_{rand} and q_{near} is greater than the maximum step d_m, d_m is used as the step to interpolate point q_{new} in the direction from q_{near} to q_{rand}. On the contrary, if the distance between q_{rand} and q_{near} is less than d_m, q_{rand} is directly set as the point q_{new}. After that, if q_{new} is located in the obstacle space Q_{obs}, the expansion of the current round will be ended. Otherwise, another random tree T_{goal} will be expanded in a directional way. The process of directional expansion is as follows. Firstly, the node q'_{near} in the tree T_{goal} which is nearest to q_{new} needs to be found, and a new

node q'_{new} with a certain search step in the direction of q_{new} node is generated. If q'_{new} is in free space Q_{free}, directional expansion towards q_{new} should be proceeded. Finally, the two random trees are tried to be connected together. If q'_{new} can reach to q_{new}, the two random trees are connected successfully and T_{path} records the connection points between two trees. In order to make the two random trees have equal opportunities to explore in the state space, the expansion modes of the two random trees are exchanged after each round of expansion.

After planning the path of the robot, the trajectory points on the path need to be sampled and recorded. In this paper, sampling in joint space is selected to record the joint angular displacement of the robot. In this way, we can use forward kinematics to determine the unique attitude point, which can ensure a high success rate of planning. The joint space of the dual-robot system composed of two six degree of freedom robots is a 12-dimensional linear space. The calculation of sampling points in space is as follows:

$$\mathbf{q} = ({}^{L1}\theta, {}^{L2}\theta, \cdots, {}^{L6}\theta, {}^{R1}\theta, {}^{R2}\theta, \cdots, {}^{R6}\theta)^T \tag{1}$$

where ${}^{L1}\theta - {}^{L6}\theta$ represents the angular displacement of joint $1-6$ of the left robot, and ${}^{R1}\theta - {}^{R6}\theta$ represents the angular displacement of joint $1-6$ of the right robot.

The RRT-Connection algorithm needs a distance function to solve the distance between two state points. For multi-joint space R^{12}, the path cost during the movement between two state points can be expressed by Manhattan distance. Manhattan distance only needs addition and subtraction, so the cost of computing in the computer is low, and this advantage is more obvious in multi-dimensional space. The specific formula is in Equation (2):

$$d(\mathbf{q}_A, \mathbf{q}_B) = \sum_{n=1}^{12} |{}^n\theta_A - {}^n\theta_B| \tag{2}$$

where $\mathbf{q}_A = ({}^1\theta_A, {}^2\theta_A, \cdots, {}^{12}\theta_A)$ and $\mathbf{q}_B = ({}^1\theta_B, {}^2\theta_B, \cdots, {}^{12}\theta_B)$ represent any two points in the joint space.

The RRT-Connect algorithm also needs a collision detection mechanism to determine whether the leaf node and geometric models are in the collision space [30]. This paper adopts the Bounding Volume Hierarchy (BVH) algorithm [31], which is a tree data structure. It does not need to traverse all bounding box pairs, but recursively detect the bounding boxes of each layer. When several pairs of bounding boxes that may collide are detected, BVH algorithm will further detect the subsets of these bounding boxes. In this way, the obvious disjoint parts in the space can be quickly excluded, so the number of bounding box pairs for intersection test are reduced as much as possible, and it improves the detection efficiency.

The process of collision detection for each point is as follows. First, the points are transformed into Cartesian space by forward kinematics. Then we use the BVH algorithm to perform collision detection in pairs for each manipulator of the dual-robot itself. If the two manipulators do not collide themselves, then perform the collision detection between the dual-robot and obstacles in the environment.

In the actual trajectory planning process, in order to avoid collision when the robot moves between path points, collision detection should be applied to every movement between path points. In other words, for a continuous path connected by two state points, it is necessary to discretize the path so as to interpolate enough intermediate points to represent the state of each movement time. The interpolation function is defined as Equation (3):

$$\mathbf{q}_x = \mathbf{q}_i + (\mathbf{q}_j - \mathbf{q}_i) \cdot \frac{x \cdot s}{d(\mathbf{q}_i, \mathbf{q}_j)} \tag{3}$$

where $\mathbf{q}_i = ({}^1\theta_i, {}^2\theta_i, \cdots, {}^{12}\theta_i)$ and $\mathbf{q}_j = ({}^1\theta_j, {}^2\theta_j, \cdots, {}^{12}\theta_j)$, respectively, represent the start point and end point of the path, $\mathbf{q}_x = ({}^1\theta_x, {}^2\theta_x, \cdots, {}^{12}\theta_x)$ is the x_{th} interpolation point in the path, and s is the interpolation step length.

In the system, the Flexible Collision Library (FCL) [32] is used to realize the collision detection of each point. The flow of collision detection is shown in Figure 5.

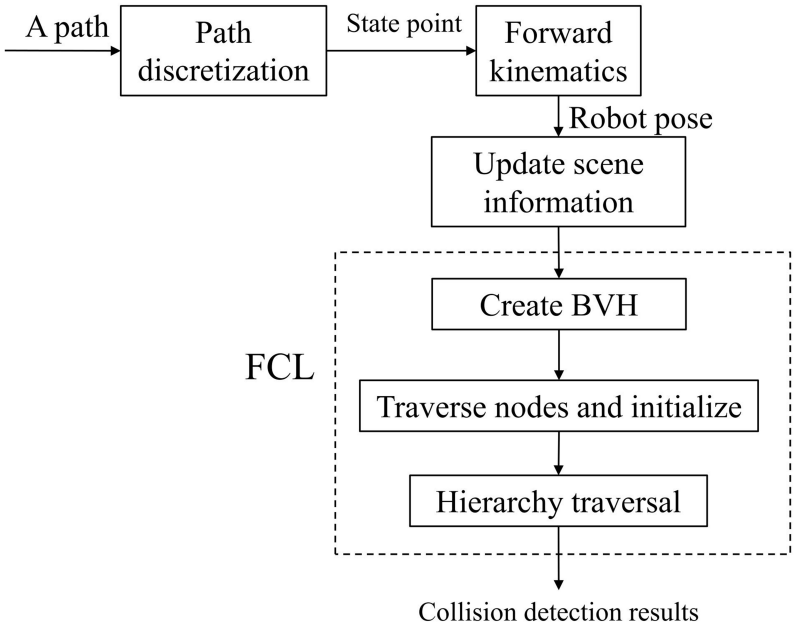


Figure 5. Collision detection process based on FCL.

4. Path Post-Processing

Using the path planning algorithm of RRT-Connect, we obtain an effective original path in the joint space. However, this path is a polyline segment generated by random sampling, which has disadvantages such as large turning angle, spiral path and lack of time parameters. Therefore, it is necessary to optimize the original path and solve these problems.

4.1. Path Simplification

In order to simplify the original path, we need to connect two original path points that are not adjacent to each other. There is usually more than one simplified path between original path points, this paper uses Floyd algorithm to simplify the original path.

Floyd algorithm is a shortest path algorithm [33], which is easy to implement and suitable for finding the shortest path between any two points. As shown in the Algorithm 2, we should first input n original path points. $\mathbf{A}^r$ and $\mathbf{P}^r$, respectively, represent the distance matrix and the path matrix after r traversals, which are both r-order matrices. The element $\mathbf{a}_{ij}^r$ of matrix $\mathbf{A}^r$ stores the shortest path length between the path points q_i and q_j after the r traversals. The initial value $\mathbf{a}_{ij}^0$ is the linear distance from q_i to q_j. If the connection between q_i and q_j is in the collision space Q_{obs}, set $\mathbf{a}_{ij}^0 = \infty$. The element $\mathbf{p}_{ij}^r$ of matrix $\mathbf{P}^r$ stores the vertex number of the shortest path from q_i to q_j. The initial value $\mathbf{p}_{ij}^0 = j$.

The specific calculation method for each traversal is as follows. It is supposed the point q_i passes through the intermediate point q_r to q_j, the shortest distance q_i to point q_r is $\mathbf{a}_{ir}^r$. In addition, the shortest distance q_r passes point q_j is $\mathbf{a}_{rj}^r$. Then compute the path length $\mathbf{a}_{ir}^r + \mathbf{a}_{rj}^r$ of point q_i to q_j which passes through the intermediate point q_r.

In the r traversals of Algorithm 2, if the element $\mathbf{a}_{ij}^r$ obtained by the r_{th} traversal is greater than the element $\mathbf{a}_{ij}^{r-1}$ obtained last time, the value will not change. Otherwise, $\mathbf{a}_{ij}^r$ will update and the element $\mathbf{p}_{ij}^r$ will record the serial number r of the interpolation point.

Algorithm 2 Floyd

Input: $q_1 \sim q_n$
Output: $\mathbf{A}^r, \mathbf{P}^r$

```
Initialize A^0, P^0
for i ← 1 to n do
  for j ← i to n do
    for r ← i to j do
      if a_ij^{r-1} < a_ir^r + a_rj^r then
        a_ij^r = a_ij^{r-1},   p_ij^r = p_ij^{r-1}
      else
        a_ij^r = a_ir^r + a_rj^r,   p_ij^r = r
      end if
    end for
  end for
end for
```

4.2. Path Smoothing

The simplified path may still have problems such as large turning angle, which can be improved by smoothing the path. In this paper, path smoothing is based on Bezier curve, which is one of the most commonly used curve approximation methods in engineering. However, the higher-order Bezier curve will bring about computational difficulties, this paper uses a multi-segment quadratic Bezier curve to approximate the simplified path. The quadratic Bezier curve requires three control points. Through calculation, we get its parameter in Equation (4):

$$C(t) = (P_2 - 2P_1 \cdot + P_0) \cdot t^2 + 2(P_1 - P_0) \cdot t + P_0 \tag{4}$$

where $P_i (i = 0, 1, 2)$ is the control point, $t (t \in [0, 1])$ is the proportional coefficient.

Equation (4) can be further deduced in Equation (5):

$$C(t) = (1 - t) \cdot P_{01} + t \cdot P_{12} \tag{5}$$

where $P_{01} = (1 - t) \cdot P_0 + t \cdot P_1$, $P_{12} = (1 - t) \cdot P_1 + t \cdot P_2$, respectively, represent the point on line segment P_0P_1 and P_1P_2.

Equation (5) shows that the point $C(t)$ of quadratic Bezier curve is a point on line segment $P_{01}P_{12}$, and satisfies the following proportional relationship [34]:

$$\frac{P_{01}C(t)}{P_{01}P_{12}} = \frac{P_0P_{01}}{P_0P_1} = \frac{P_1P_{12}}{P_1P_2} = t \tag{6}$$

In addition, when two quadratic Bezier curves $C_P(t)$ and $C_Q(t)$ with their control point sequences $P_i (i = 0, 1, \cdots, n)$ and $Q_j (j = 0, 1, \cdots, n)$ are given, we need to splice these two curves. Firstly, the two curves need to meet the zero-order parameter continuity. Therefore, the end point of the first curve should coincide with the start point of the second curve $(P_n = Q_0)$. Secondly, in order to make the splice smooth, the two curves must meet at least the first-order parameter continuity, so the splice point should be located at the midpoint of line segment $P_{n-1} = Q_1$ [35]. Finally, the collision detection is carried out again. If the movement of these curve points is feasible, the path smoothing can be realized by splicing multiple quadratic Bezier curves. Otherwise, the original part of the path is retained.

Taking Figure 6 as an example: The midpoints $P_0^{'}$, $P_1^{'}$, $P_2^{'}$, and $P_3^{'}$ of the original line segment are used as the splicing points of each Bezier curve. Take $t = 0.5$ on each segment of the curve to interpolate points $C_1(0.5)$, $C_2(0.5)$, and $C_3(0.5)$. Therefore, the smoothed path is $P_0 \to P_0^{'} \to C_1(0.5) \to P_1^{'} \to C_2(0.5) \to P_2^{'} \to C_3(0.5) \to P_3^{'} \to P_4$.

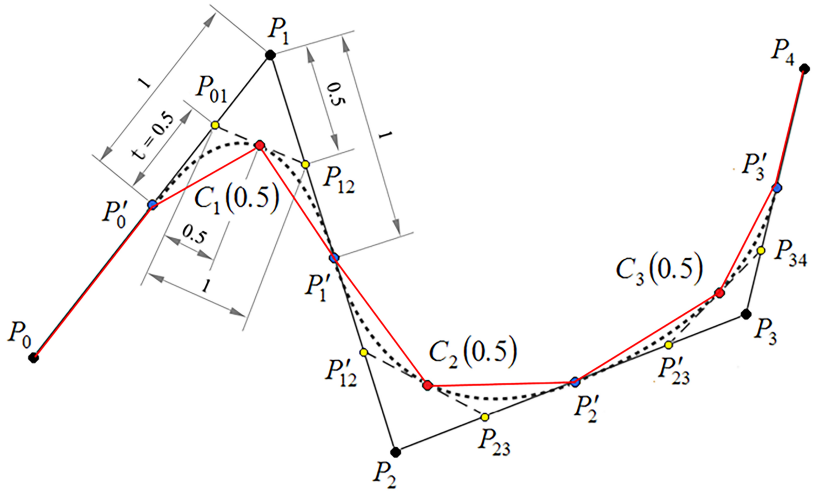


Figure 6. Schematic diagram of path smoothing.

4.3. Path Time Parameterization

It is necessary to specify the motion time parameters of each path on the basis of meeting the speed and acceleration limits of each joint. Because the path planned by sampling is complex and changeable, it is difficult to obtain the analytical formula of the path, so this paper uses the iterative method to calculate the motion time. We use Equation (3) to interpolate the smoothed points of each segment, and these subdivided path points will be used as joint position parameters for controlling the motion of the robot.

Assume that the planned path P has n path points, which are recorded as $P = \mathbf{q}_i (i = 1, 2, \cdots, n)$. Each path point $\mathbf{q}_i$ contains the position information of 12 joints, and the position of each joint is recorded as ${}^j\theta_i (j = 1, 2, \cdots, 12)$. Accordingly, the instantaneous velocity and acceleration of this position are recorded as ${}^jv_i$ and ${}^ja_i$. It assumed that each joint moves from $\mathbf{q}_i$ to $\mathbf{q}_{i+1}$ at the maximum joint speed ${}^jv_{max}$, the motion time T_i shall meet the constraints of each joint and it can be calculated in Equation (7):

$$T_i = max\left\{ \frac{{}^j\theta_{i+1} - {}^j\theta_i}{{}^jv_{max}} \right\} \tag{7}$$

The motion time also need to meet the constraint of the acceleration of the joint. It can be seen from Equation (3) that when the interpolation length s is very small, the change of the joint angle will also be small. From Equation (7), T_i will also a small value. Therefore, We can regard the speed from ${}^j\theta_i$ to ${}^j\theta_{i+1}$ as a uniform speed change, and according to the definition of acceleration, we get Equation (8):

$${}^ja_i = 2 \cdot \frac{{}^j\bar{v}_i - {}^j\bar{v}_{i-1}}{T_i + T_{i-1}} \tag{8}$$

where $1 < i < n$, ${}^j\bar{v}_i$ represents the average velocity of joint j from ${}^j\theta_i$ to ${}^j\theta_{i+1}$.

According to Equation (8), the shorter the time interval $T_i + T_{i-1}$, the greater the acceleration. To limit the acceleration in a certain range, it is necessary to increase T_{i-1} or T_i. At the same time, the acceleration ${}^ja_i$ of each joint can be recursive, i can be recursive from 1 to n to obtain the acceleration in the path, and can also be recursive from n to 1 to obtain the deceleration in the path.

In the process of forward recursion, the new motion time $T_i^{'}$ between each path point is calculated in Equation (9):

$$T_i^{'} = T_i + \Delta T \tag{9}$$

where T_i is the motion time to be adjusted to meet the joint speed limit, and ΔT is the increment of time. In this paper, $\Delta T = 0.01$. By using the increased new motion time $T_i^{'}$, the corresponding speed and acceleration can be calculated in Equations (10) and (11):

$$^{j}\vec{v}_i^{'} = \frac{^{j}\theta_{i+1} - ^{j}\theta_i}{T_i^{'}} \tag{10}$$

$$^{j}a_i^{'} = 2 \cdot \frac{^{j}\vec{v}_i^{'} - ^{j}\vec{v}_{i-1}^{'}}{T_i^{'} + T_{i-1}} \tag{11}$$

The process of backward recursion is similar to that of forward recursion, which will not be repeated here. In the process of iteration, the processes of forward recursion and reverse recursion will be repeatedly adjusted until $^{j}a_i$ of each joint is less than $^{j}a_{max}$. Finally, we can calculate the time of each trajectory point.

5. Experiment and Discussion

The dual-robot platform is built based on two independent industrial robots ABB IRB120 with end effectors. The end effector is a pneumatic finger gripper. The I/O module of the robot controller can control the gas conduction to realize the opening and clamping of the gripper. Before experiments, we calibrated the base coordinate system of the dual-robot and imported its model into RViz. Simulations were run on a single core of a standard desktop CPU (Intel Core i7, 2.6 GHz) through Ubuntu 18.04 and ROS Melodic.

5.1. Comparison of Path Planning and Post-Processing

In order to test the rationality of the algorithm used in this paper for path planning and post-processing in complex environment, the experimental scene is designed and shown in Figure 7. Four rings are placed as obstacles within the working space of dual-robot. The robot needs to extract from the current ring and extend into another ring of the same color without collision.

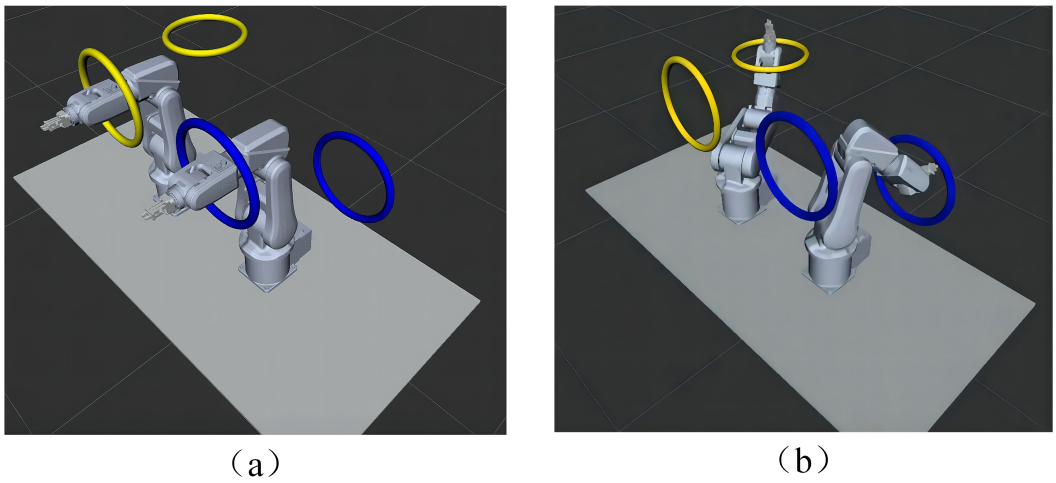


Figure 7. Experimental scenario 1. (**a**) Motion start state. (**b**) Motion end state.

Then Floyd algorithm is used to simplify the path. The original sampled joint path and the simplified joint path are compared in Figure 8. The original path is generated by sampling through the RRT-Connect algorithm, and a total of 17 path points are obtained. After being simplified by the Floyd algorithm, points 1, 3, 6, 7, 13, 14, 16, and 17 are retained: in total, 8 waypoints.

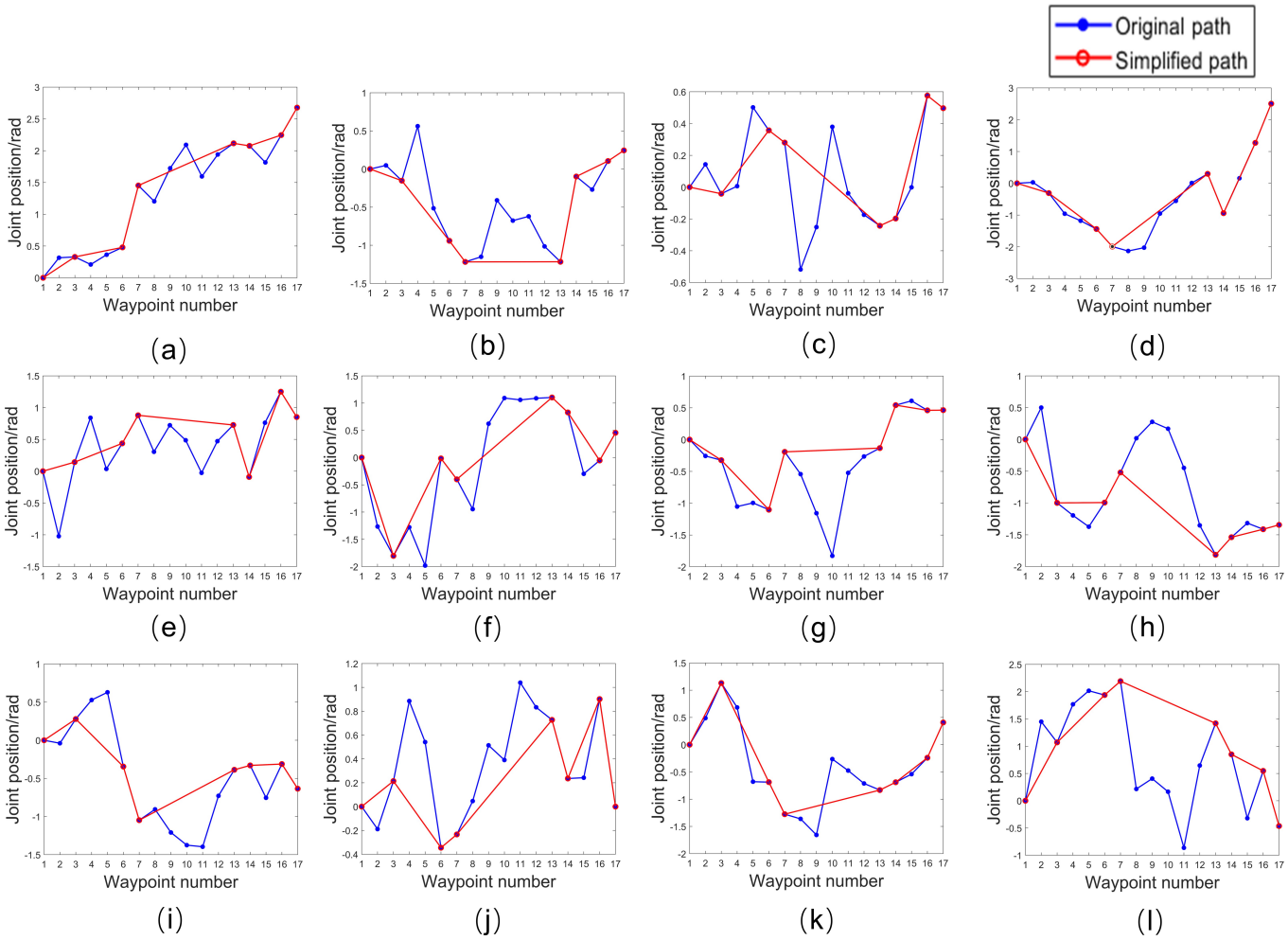


Figure 8. Comparison of original path and simplified path of each joint. (**a**–**f**) Positions of the left robot joint 1–6. (**g**–**l**) Positions of the right robot joint 1–6.

The original joint points and simplified joint points are transformed into path points through forward kinematics and drawn in RViz. The vertical view is shown in Figure 9, in which the cyan polyline is the connection of the original path points and the purple polyline is the connection of the simplified path points. It can be seen that due to the randomness of RRT-Connect, the motion under the original path is very tortuous. After simplification, many redundant actions are omitted and its motion becomes simple and effective.

In order to test the effectiveness of the path smoothing method, the above simplified path is smoothed, and the comparison results are shown in Figure 10. In this experiment, only the points with $t = 0.5$ in the quadratic Bezier curve are taken during smoothing.

It can be seen from the figure that due to the parameter continuity of Bezier curve, each polyline in the simplified path is inserted into a midpoint, so the smooth path obtained by each joint has 15 points. Points 3, 5, 7 and 9 in the simplified path are replaced by points on the Bezier curve, making this part of the path smooth. When smoothing the point 11 and point 13 in the simplified path, collision is detected, so the point 11 and point 13 are not replaced by Bezier curve points. The simplified and smoothed joint points are converted into path points through forward kinematics, and draw them in RViz. The vertical view is as shown in the Figure 11. The purple polyline is the connection of simplified joint points, and the green polyline is the connection of smooth joint points. It can be seen from the figure that the complexity of the two paths is similar, but the smoothed path has a smaller corner.

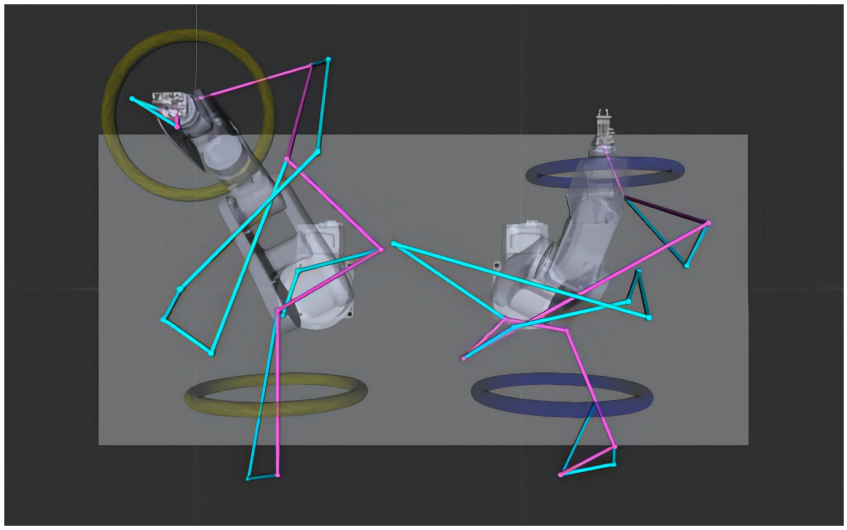


Figure 9. Comparison of original path and simplified path in RViz.

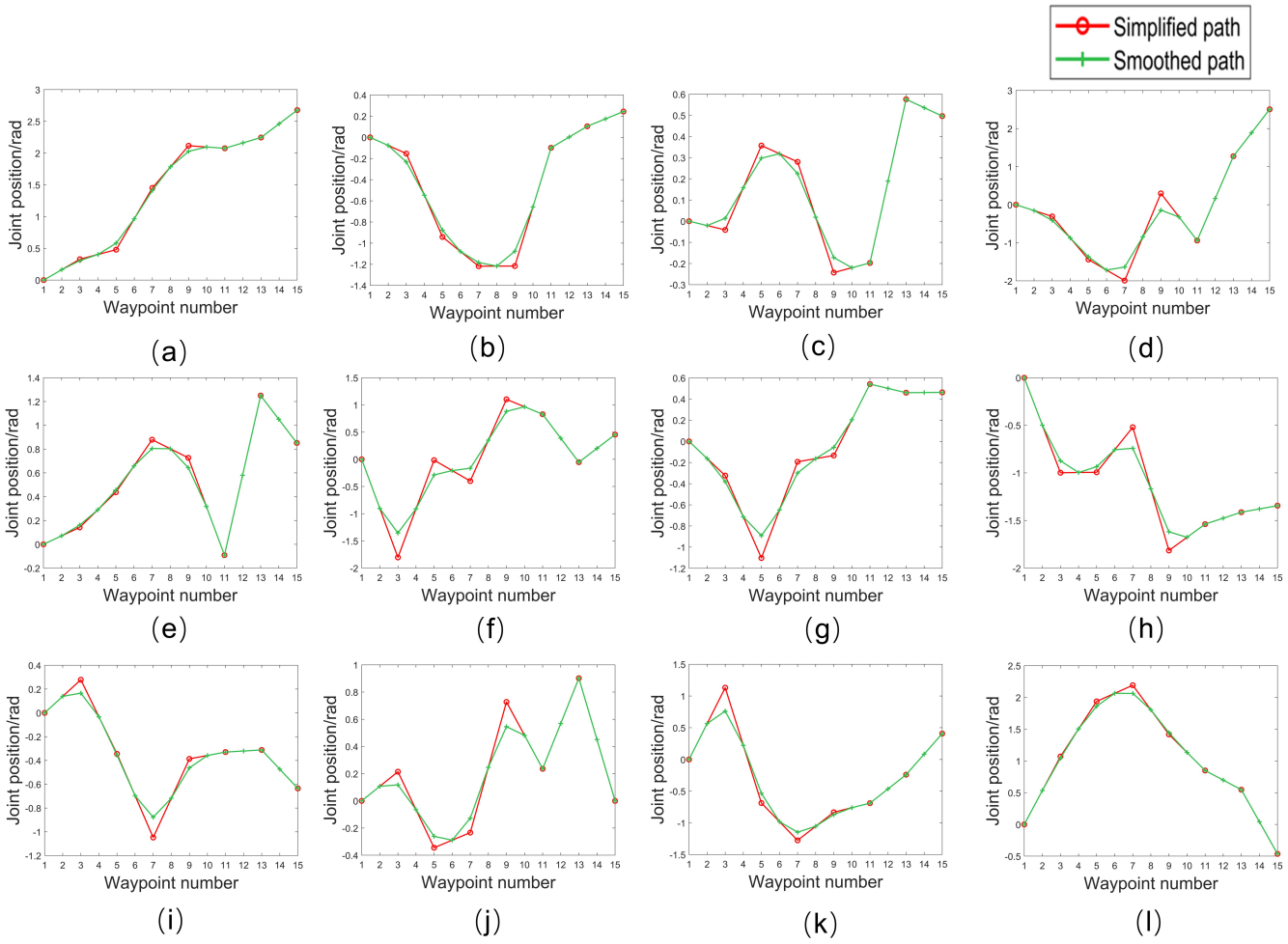


Figure 10. Comparison of simplified path and smoothed path of each joint. (**a**–**f**) Positions of the left robot joint 1–6. (**g**–**l**) Positions of the right robot joint 1–6.

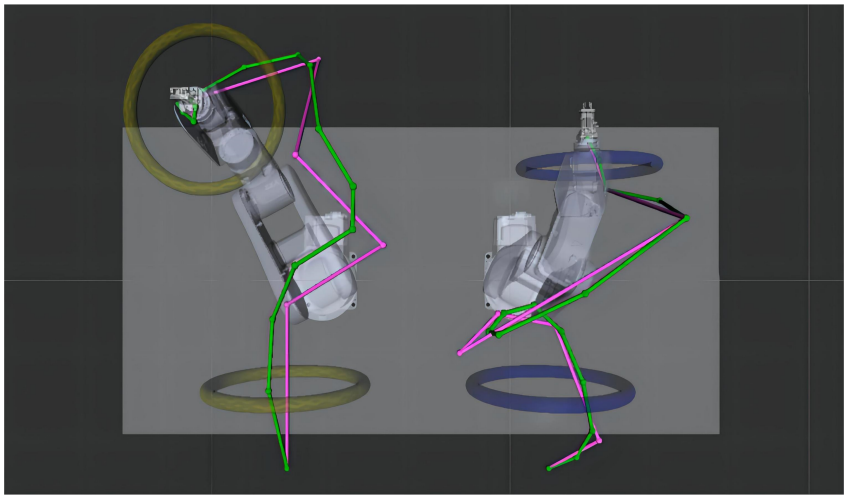


Figure 11. Comparison of simplified path and smoothed path in RViz.

Finally, the time parameterized calculation of each joint path is performed. The obtained position, velocity, and acceleration are recorded, and their curves are shown in Figure 12. It can be seen from the figure that on each segment of the smoothed path, each joint moves at a speed trajectory similar to a parabola. During the movement, the joint speed is continuous, but the acceleration has abrupt changes. In the curve segment where the joint position changes greatly, the speed and acceleration change greatly, but the speed and acceleration will not exceed the limit of robot joint. Because the velocity and acceleration of each joint can meet the requirements, it can be proved that the planned robot trajectory is effective.

5.2. Efficiency Comparison with Other Methods

In order to further test the ability of the method proposed in this paper to carry out the dual-robot motion planning, we compare other algorithms using the publicly available Open Motion Planning Library (OMPL) [36] implementations. In addition, on the basis of Figure 7 (Scenario 1), the scene shown in Figure 13 (Scenario 2) is set up in RViz to conduct experiments. In this figure, the cyan door-shaped frame is an obstacle, and a gap of 200 mm high is left in the middle of the obstacle. The dual-robot will move from the starting position in Figure 14a to the other side of the door-shaped obstacle in Figure 14b. Compared with Figure 7, the two arms must pass through the gap one after the other to achieve collision-free motion in this scenario. The emphasis on trajectory planning in the dynamic interference of two arms increases the difficulty of algorithm solving.

Our method based on RRT-Connect is not a asymptotically optimal planner. Therefore, it was first compared to T-RRT [23], and BIT-RRT [24] which are nonasymptotically optimal planners. All algorithm parameters were chosen in good faith to maximize performance. The three planners ran 100 times in each of the two scenarios from the same start state to end state.

Table 3 compares the time consumption of the planners in detail. The results show that the average planning time of our method is the lowest in two different scenarios. Moreover, the planning time distribution is also more concentrated than that of the other two planners, which means that our method has a more stable performance. In the more complex environment of Figure 13 (Scenario 2), the median time of our method has changed from 4.531 s to 6.698 s which is only an increase of 2.267 s, while the planning time of other methods has increased significantly. It can be seen that our method is more efficient than the other two planners in complex environments.

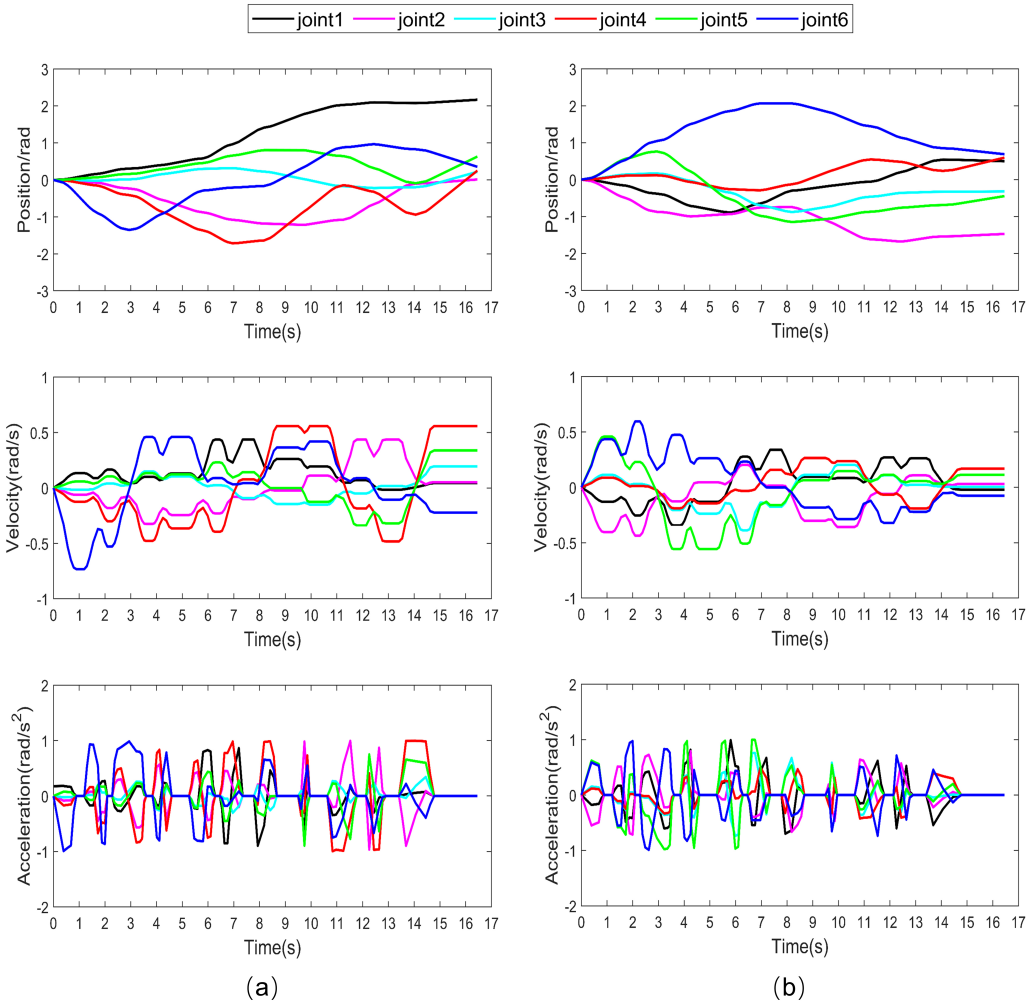


Figure 12. Position-velocity-acceleration curve of each dual-robot joint. (**a**) Joint 1–6 of left robot. (**b**) Joint 1–6 of right robot.

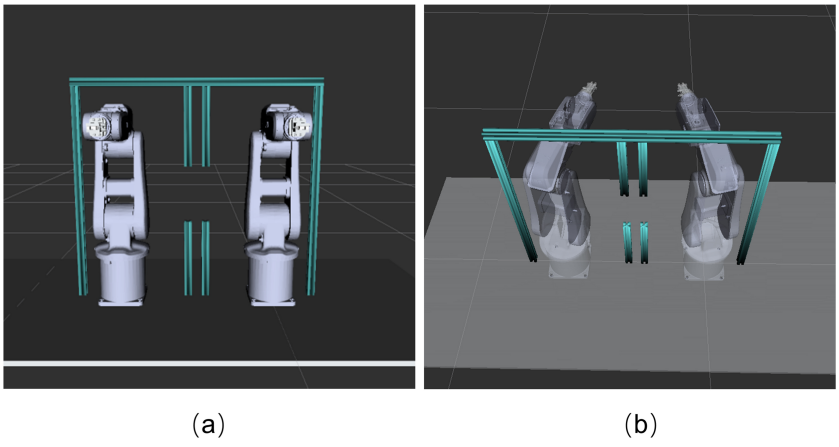
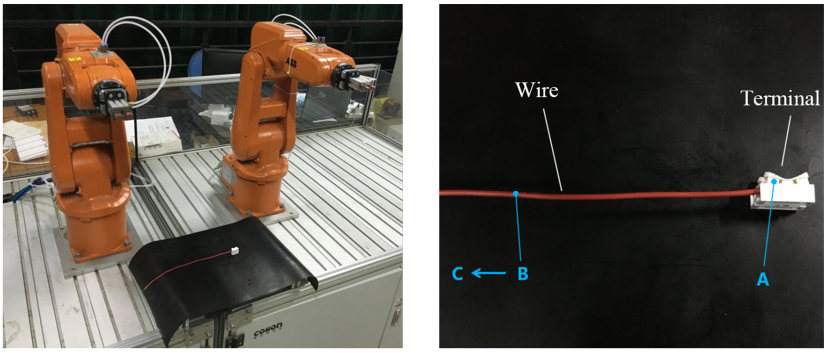


Figure 13. Experimental scenario 2. (**a**) Motion start state. (**b**) Motion end state.



(a) (b)

Figure 14. Experimental scene of disassembling for terminal with Wire. (**a**) Overall layout of the experiment. (**b**) Partial enlarged view.

Table 3. Comparison of the planning time (s) of planners.

	Scenario 1			Scenario 2		
Method	**Median**	**Max**	**Std. Dev.**	**Median**	**Max**	**Std. Dev.**
T-RRT	30.103	46.584	9.645	108.209	141.601	22.984
BIT-RRT	16.859	31.211	8.763	56.354	78.649	12.344
Ours	4.531	13.838	3.475	6.698	18.159	5.784

We further compare the planning efficiency with Informed RRT* [27], and BIT* [28] which are asymptotically optimal planners. As the true optima for these algorithms are different and unknown, there is no meaningful way to compare path quality across algorithms. In addition, the experiments in Section 5.1 show that our method can obtain high-quality trajectory, so we no longer evaluate and compare the trajectory quality. Instead, performance of three planners was evaluated for the success rate and solution cost within the specified planning time.

Table 4 illustrates the comparison of the success rate and solution cost in 100 repeated experiments. Given one minute of planning time in Scenario 1, RRT-Connect was 100% successful though the median solution cost is 20.4. BIT* has a lower median solution cost, but it does not guarantee success within the planning time. Informed RRT* has the lowest success rate, and its median solution cost is also slightly higher than our method. Given 2. 5 min of planning time in the more complex Scenario 2, the experimental results are similar to scenario 1. It is worth mentioning that the success rate of our method is still 100%.

Table 4. Comparison of the success rate and solution cost of planners in specified planning time.

	Scenario 1 (In 1 min)		Scenario 2 (In 2.5 min)	
Method	**Median Solution Cost**	**Success Rate**	**Median Solution Cost**	**Success Rate**
Informed RRT*	21.7	26%	46.8	9%
BIT*	15.6	86%	24.6	71%
Ours	20.4	100%	38.5	100%

5.3. Experiments of Dual-Robot Cooperative Disassembling Wire from Terminal

In order to test whether the dual-robot can complete a task using the planned trajectory, an experiment was designed for the dual-robot to cooperate to disassemble terminal with Wire. The experimental scene is shown in Figure 14. The red wire is inserted in the terminal. The part of the wire buried in the socket is clamped inside the terminal. In order to separate the wire from the terminal, the button on the terminal needs to be pressed to loosen the socket, while the wire should be pulled out at the same time.

Decomposing the task, the specific actions performed by the dual-robot are:

1. The right robot moves to point A to clamp the terminal $S1$ and presses its button to loosen the socket. At the same time, the left robot moves to point B to clamp the wire $S2$.
2. The right robot keeps pressing while the left robot draws out the wire in a straight line. When this action is completed, the left robot reaches to the point C.
3. The right robot puts down the terminal at point A, and the left robot puts down the wire at point C.
4. The two robots return to their respective initial points $O1$ and $O2$.

The object model of terminal, wire and main point information are input into the human-computer interface. In addition, according to the task decomposition, dual-robot action instructions are arranged in the human-computer interface as shown in Table 5.

Table 5. Action instructions of dual-robot cooperative disassembling for terminal with wire.

Number	Left Robot	Right Robot
1	$Pick(S2, \mathbf{V}, \mathbf{0})$	$Pick(S1, \mathbf{V}, \mathbf{0})$
2	$Linear(C)$	$Hold()$
3	$Place(C, \mathbf{0}, -\mathbf{V})$	$Place(A, \mathbf{0}, -\mathbf{V})$
4	$Reach(O1)$	$Reach(O2)$

Finally, the task is simulated in RViz and the trajectory file is generated. Figure 15 shows the task execution process of dual-robot in RViz. First, as shown in Figure 15a, the two robots move at the same time and approach their respective grasping points. After reaching the above grasping point, the two arms move vertically downward, the right robot grabs the terminal, and the left robot grabs the wire as shown in Figure 15b. Then, the left robot moves along the wire away from the terminal and pulls out the wire as shown in Figure 15c. After that, the two robots put down the object, respectively, as shown in Figure 15d. After releasing the object, the robots return to the initial point as shown in Figure 15e. The trajectory generated by the simulation can be further sent to the sub-controllers for execution.

The generated trajectory file is sent to the sub-controllers, and the trajectory reproduction is completed in the actual experimental scene. The moving process of the dual-robot is shown in the Figure 16. This process is completely consistent with the motion process in the simulation environment, and actual dual-robot successfully performed the task of removing the wire. The experimental results show that the trajectory planning method and the dual-robot cooperation system are feasible.

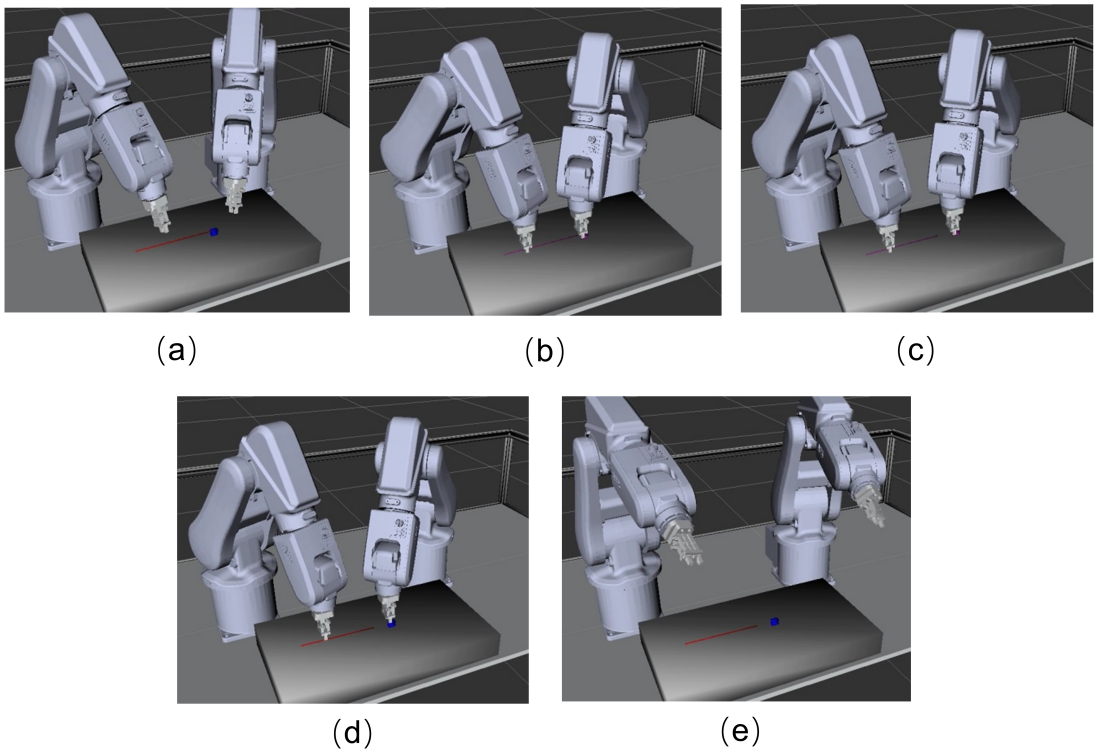


Figure 15. Task simulation in RViz. (**a**) Dual-robot approaches grasping points. (**b**) Dual-robot grabs object. (**c**) Left robot pulls out the wire. (**d**) Dual-robot puts down the object. (**e**) Dual-robot returns to the origin.

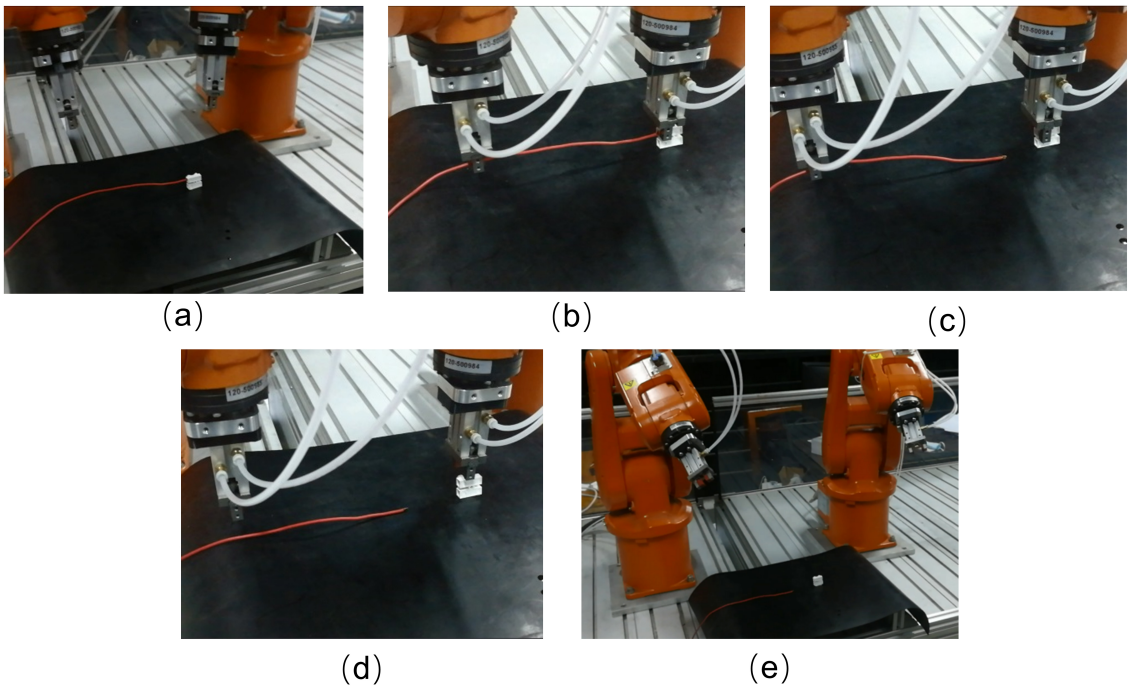


Figure 16. Perform tasks in real robot scene. (**a**) Dual-robot approaches grasping points. (**b**) Dual-robot grabs object. (**c**) Left robot pulls out the wire. (**d**) Dual-robot puts down the object. (**e**) Dual-robot returns to the origin.

6. Conclusions

This paper is motivated to provide a solution to the trajectory planning for dual-robot cooperative assembly. Due to the high-dimensional space of dual-robot and the complex environment of dynamic changes, the method of random sampling based motion planning is adopted. The random path is generated in the working space based on RRT-Connect algorithm. The original path planning of dual-robot is carried out combined with BVH algorithm. In order to solve problems of existing many broken lines and circling paths of the original path, the obtained path is simplified by Floyd algorithm. Aiming at the problem of large turning angles of path, the path is smoothed by Bezier curve. In order to obtain the trajectory point information with time parameters, the time information corresponding to each joint angle is derived by iterative method. Finally, the effective joint trajectory of the dual-robot is obtained.

Experiments show that the proposed path planning method can effectively plan a simple, smooth and fast path for dual-robot to complete the assembly task. It provides a whole method for trajectory planning of dual-robot cooperative assembly.

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

Funding: This work was supported by the National Key Research and Development Project under Grant 2018YFB1308603 and by Intelligent Manufacturing Comprehensive Standardization Project under Grant GXSP20181001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data generated or analyzed during the study are included in the article, and the data that support the findings of this study are openly available.

Conflicts of Interest: All authors confirm that there are no known conflict of interest associated with this publication.

References

1. Wang, T.; Tao, Y. Research status and industrialization development strategy of Chinese industrial robot. *J. Mech. Eng.* **2014**, *50*, 1–13. [CrossRef]
2. Sun, D.; Mills, J. Adaptive synchronized control for coordination of multirobot assembly tasks. *IEEE Trans. Robot. Autom.* **2002**, *18*, 498–510. [CrossRef]
3. Koditschek, D. E. An approach to autonomous robot assembly. *Robotica* **1994**, *12*, 137–155. [CrossRef]
4. Do, H.M.; Choi, T.Y.; Kyung, J.H. Automation of cell production system for cellular phones using dual-arm robots. *Int. J. Adv. Manuf. Technol.* **2016**, *83*, 1349–1360. [CrossRef]
5. Park, D.I.; Park, C.; Do, H.; Choi, T.; Kyung, J. Development of dual arm robot platform for automatic assembly. In Proceedings of the 2014 14th International Conference on Control, Automation and Systems (ICCAS 2014), Gyeonggi-do, Korea, 22–25 October 2014; pp. 319–321. [CrossRef]
6. Smith, C.; Karayiannidis, Y.; Nalpantidis, L.; Gratal, X.; Qi, P.; Dimarogonas, D.V.; Kragic, D. Dual arm manipulation—A survey. *Robot. Auton. Syst.* **2012**, *60*, 1340–1353. [CrossRef]
7. Yamada, Y.; Nagamatsu, S.; Sato, Y. Development of multi-arm robots for automobile assembly. In Proceedings of the 1995 IEEE International Conference on Robotics and Automation, Aichi, Japan, 21–27 May 1995; Volume 3, pp. 2224–2229. [CrossRef]
8. Surdilovic, D.; Yakut, Y.; Nguyen, T.M.; Pham, X.B.; Vick, A.; Martin-Martin, R. Compliance control with dual-arm humanoid robots: Design, planning and programming. In Proceedings of the 2010 10th IEEE-RAS International Conference on Humanoid Robots, Nashville, TN, USA, 6–8 December 2010; pp. 275–281. [CrossRef]
9. Cousins, S. ROS on the PR2 [ROS Topics]. *IEEE Robot. Autom. Mag.* **2010**, *17*, 23–25. [CrossRef]
10. Alebooyeh, M.; Urbanic, R.J. Neural Network Model for Identifying Workspace, Forward and Inverse Kinematics of the 7-DOF YuMi 14000 ABB Collaborative Robot. *IFAC-Papersonline* **2019**, *52*, 176–181. [CrossRef]
11. Park, C.; Park, K.; Park, D.I.; Kyung, J.H. Dual arm robot manipulator and its easy teaching system. In Proceedings of the 2009 IEEE International Symposium on Assembly and Manufacturing, Seoul, Korea, 17–20 November 2009; pp. 242–247.

12. Tran, N.; Nguyen, D.T.; Vu, D.L.; Truong, N.V. Global path planning for autonomous robots using modified visibility-graph. In Proceedings of the International Conference on Control, Automation and Information Sciences (ICCAIS), Ho Chi Minh City, Vietnam, 25–28 November 2013; pp. 317–321. [CrossRef]
13. Wang, K.H.C.; Botea, A. Tractable multi-agent path planning on grid maps. In Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, CA, USA, 11–17 July 2009; pp. 1870–1875.
14. Bounini, F.; Gingras, D.; Pollart, H.; Gruyer, D. Modified artificial potential field method for online path planning applications. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Redondo Beach, CA, USA, 11–14 June 2017; pp. 180–185. [CrossRef]
15. Nazarahari, M.; Khanmirza, E.; Doostie, S. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Syst. Appl.* **2019**, *115*, 106–120. [CrossRef]
16. Chuang, J.; Lin, C.; Chou, T. An Alternate Priority Planning Algorithm for Dual-Arm Systems. In Proceedings of the 2006 IEEE International Symposium on Industrial Electronics, Montreal, QC, Canada, 9–13 July 2006; Volume 4, pp. 3084–3089. [CrossRef]
17. Agrawal, S.K.; Shirumalla, S. Planning motions of a dual-arm free-floating manipulator keeping the base inertially fixed. *Mech. Mach. Theory* **1995**, *30*, 59–70. [CrossRef]
18. Sezgin, U.; Seneviratne, L.D.; Earles, S.W.E. Redundancy utilization for obstacle avoidance of planar robot manipulators. *Proc. Inst. Mech. Eng. Part J. Mech. Eng. Sci.* **1997**, *211*, 463–475. [CrossRef]
19. Sezgin, U.; Seneviratne, L.; Earles, S. Collision Avoidance in Multiple-Redundant Manipulators. *Int. J. Robot. Res.* **1997**, *16*, 714–724. [CrossRef]
20. Liu, H.; Dai, J. Manipulation analysis of carton-folding task by using GGPs and CCPs. In Proceedings of the IEEE Internatinal Symposium on Intelligent Control, Vancouver, BC, Canada, 30–30 October 2002; pp. 637–641.
21. Kavraki, L.; Svestka, P.; Latombe, J.C.; Overmars, M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
22. LaValle, S.M.; Kuffner, J.J. Randomized Kinodynamic Planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
23. Jaillet, L.; Cortes, J.; Simeon, T. Transition-based RRT for path planning in continuous cost spaces. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; pp. 2145–2150. [CrossRef]
24. Devaurs, D.; Siméon, T.; Cortés, J. Enhancing the transition-based RRT to deal with complex cost spaces. In Proceedings of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 4120–4125. [CrossRef]
25. Suh, J.; Gong, J.; Oh, S. Fast Sampling-Based Cost-Aware Path Planning With Nonmyopic Extensions Using Cross Entropy. *IEEE Trans. Robot.* **2017**, *33*, 1313–1326. [CrossRef]
26. Xu, J.; Song, K.; Zhang, D.; Dong, H.; Yan, Y.; Meng, Q. Informed Anytime Fast Marching Tree for Asymptotically Optimal Motion Planning. *IEEE Trans. Ind. Electron.* **2021**, *68*, 5068–5077. [CrossRef]
27. 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 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2997–3004. [CrossRef]
28. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May, 2015; pp. 3067–3074. [CrossRef]
29. Kuffner, J.; LaValle, S. RRT-Connect: An efficient approach to single-query path planning. In Proceedings of the 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 [CrossRef]
30. Bialkowski, J.; Otte, M.; Karaman, S.; Frazzoli, E. Efficient collision checking in sampling-based motion planning via safety certificates. *Int. J. Robot. Res.* **2016**, *35*, 767–796. [CrossRef]
31. Fang, Z.; Xu, J.; Jiang, J.; Wang, X. Efficient collision detection using bounding volume hierarchies of OBB-AABBs and its application. In Proceedings of the International Conference On Computer Design and Applications, Washington, DC, USA, 21–25 June 2010; Volume 5, pp. 242–246. [CrossRef]
32. Pan, J.; Chitta, S.; Manocha, D. FCL: A general purpose library for collision and proximity queries. In Proceedings of the IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 3859–3866.
33. Shi, W.; Wang, K. Floyd algorithm for the shortest path planning of mobile robot. *Chin. J. Sci. Instrum.* **2009**, *30*, 2088–2092. [CrossRef]
34. Boehm, W.; Müller, A. On de Casteljau's algorithm. *Comput. Aided Geom. Des.* **1999**, *16*, 587–605. [CrossRef]
35. Song, Y.; Fan, C.; Chen, J. On Smooth Connection between Two Bézier Spline Curves in Gn Continuity. *J. Xuchang Univ.* **2008**, *5*, 34–36.
36. Sucan, I.A.; Moll, M.; Kavraki, L.E. The Open Motion Planning Library. *IEEE Robot. Autom. Mag.* **2012**, *19*, 72–82. [CrossRef]