machines

MDPI

Article

Bidirectional RRT*-Based Path Planning for Tight Coordination of Dual Redundant Manipulators

Jun Dai [1,2], Yi Zhang [1,2] and Hua Deng [1,2,*]

[1] School of Mechanical and Electrical Engineering, Central South University, Changsha 410083, China
[2] The State Key Laboratory of High Performance Complex Manufacturing, Central South University, Changsha 410083, China
* Correspondence: hdeng@csu.edu.cn

Abstract: There are closed-chain constraints between the left manipulator and the right manipulator in tight coordination of the dual redundant manipulator. The existing planning algorithms suitable for loose coordination cannot be directly applied to tight coordination, as the planned path cannot satisfy the closed-chain constraints. To solve the above problem, a master-slave planning method based on bidirectional RRT* is proposed for dual redundant manipulators. Bidirectional RRT* is adopted to plan the path of the master manipulator. The path of the slave manipulator is calculated by terminal generalized velocity constraints instead of terminal position and posture constraints. Moreover, a local path replanning strategy is proposed to solve the problem that the planned path is actually not feasible due to the discontinuous joint path of the slave manipulator. The joint self-motion in the null space is utilized to keep the terminal position and posture of the slave manipulator unchanged. The proposed method is verified by simulations and experiments and the results show that it can solve the discontinuity problem, increase the success rate, shorten the planning time and satisfy closed-chain constraints. Therefore, the proposed method can be feasibly and effectively applied to the tight coordination of dual redundant manipulators.

Keywords: dual redundant manipulator; closed-chain constraint; bidirectional RRT*; master-slave planning; local path replanning

Citation: Dai, J.; Zhang, Y.; Deng, H. Bidirectional RRT*-Based Path Planning for Tight Coordination of Dual Redundant Manipulators. *Machines* **2023**, *11*, 209. https://doi.org/10.3390/machines11020209

Received: 14 December 2022
Revised: 28 January 2023
Accepted: 30 January 2023
Published: 1 February 2023

1. Introduction

Redundant manipulators, compared with nonredundant manipulators [1,2], can be more flexible to perform a variety of delicate and complex operations due to their increased degrees of freedom [3–5]. Hence, they are widely applied to various industrial fields. For many operations such as transportation, assembly, maintenance and processing of complex parts, single redundant manipulators may not be competent due to the limitations of their load-bearing capacity, workspace and application range. Therefore, coordination operations of dual redundant manipulators have attracted extensive attention [6–8].

Path planning is one of the key technologies for safe operations of dual redundant manipulators [9–11]. Sampling-based algorithms are the most popular algorithms for path planning. Rapidly exploring random tree (RRT) and its derivative algorithms are the mainstream among various sampling-based algorithms [12–17]. Yu et al. [18] proposed a spline RRT* for coordinated motion planning of dual-arm space manipulators. RRT* was adopted to plan the desired path and then the quartic spline function was adopted to smooth it. Ying et al. [19] proposed a bidirectional RRT with deep learning for the cooperative assembly of dual manipulators. The historical training data of deep learning were adopted to generate new nodes that were close to the goal node and away from the obstacles. Chen et al. [20] proposed a bidirectional RRT with post-processing for the cooperative assembly of dual manipulators. After the planning of bidirectional RRT was completed, the path was optimized by post-processing. Compared with RRT, bidirectional

RRT has a faster convergence speed, and RRT* can plan an optimized path. To shorten the planning time and path length, bidirectional RRT* (B-RRT*) is proposed by combining bidirectional RRT with RRT* [21,22]. Therefore, B-RRT* is adopted in this study for path planning of dual redundant manipulators.

Existing algorithms based on RRT are generally applied to plan the path for loose coordination of dual redundant manipulators [18–20,23–25]. Compared with loose coordination, the terminal positions and postures of the left manipulator and the right manipulator are mutually restricted by the kinematic closed-chain constraints in tight coordination [26,27]. Therefore, planning algorithms suitable for loose coordination cannot be directly applied to tight coordination. To solve the above problem, a master-slave planning method based on B-RRT* is proposed for tight coordination of dual redundant manipulators in this study.

For general master-slave planning methods, the path of the slave manipulator in the task space is obtained by the terminal position and posture constraints, and then its path in the joint space is calculated by solving the inverse kinematics [28,29]. For redundant manipulators, the inverse kinematics is usually solved by numerical algorithms. Numerical algorithms generally obtain the joint angular velocity of the redundant manipulator through its terminal generalized velocity and then calculate the joint angles accordingly [30,31]. According to the above characteristics, the terminal generalized velocity constraints are directly adopted to obtain the joint angular velocity of the slave manipulator in our study. Therefore, it can avoid the calculation of inverse kinematics and improve the execution efficiency.

When the master-slave planning method based on B-RRT* is applied to dual redundant manipulators, it cannot guarantee the continuity of the path at the connection nodes of two trees in the joint space. The reasons for this problem are described as follows. The path of the master manipulator is planned by the planning algorithm. Hence, it is continuous at the two connection nodes in both the task space and the joint space. Nevertheless, the path of the slave manipulator is calculated by the closed-chain constraints that are just acted on the terminal states. Therefore, it can only ensure that its path is continuous at the two connection nodes in the task space. Since there are infinite sets of inverse kinematics solutions for redundant manipulators, there is a high probability that the two sets of joint angles at the two connection nodes are inconsistent and usually quite different. That is, the path of the slave manipulator may be discontinuous at the two connection nodes in the joint space. As a result, there is an additional motion between the two connection nodes in the actual motion due to the above discontinuity problem. This additional motion is not included in the planned path. Therefore, the dual redundant manipulator may collide with obstacles or fail to satisfy the closed-chain constraints in the actual motion. The planned path is actually not feasible in this case. To solve the discontinuity problem, it is necessary to modify the master-slave planning method based on B-RRT* so that the planned path is continuous in both the task space and the joint space.

The main contributions of this paper are highlighted as follows:

(1) For tight coordination of dual redundant manipulators, a master-slave planning method based on B-RRT* is proposed. One manipulator is regarded as the master manipulator and the other is regarded as the slave manipulator. The path of the master manipulator is planned by B-RRT* and then the path of the slave manipulator is calculated according to the kinematic closed-chain constraints.

(2) Considering the characteristics of dual redundant manipulators, the path of the slave manipulator is calculated by the terminal generalized velocity constraints instead of the terminal position and posture constraints. In this way, the calculation of the inverse kinematics solutions can be avoided, and the planning efficiency can be improved.

(3) To solve the discontinuity problem in the joint path of the slave manipulator at the connection nodes when the master-slave planning method based on B-RRT* is applied to dual redundant manipulators, a local path replanning strategy is designed on the basis of the joint self-motion in the null space. On the premise that the terminal position and posture of the slave manipulator remain unchanged, the local path of

the slave manipulator between the two connection nodes is replanned to guarantee its continuity and satisfy the closed-chain constraints.

The subsequent contents of this paper are organized as follows. In Section 2, the closed-chain constraints of dual redundant manipulators are described. The master-slave planning method based on B-RRT* for dual redundant manipulators is introduced. The local path replanning strategy is designed. In Section 3, comparative simulations and experiments are carried out, and their results are analyzed. In Section 4, innovations, extended applications and limitations of our study are discussed. In Section 5, conclusions are given.

2. Materials and Methods

2.1. Closed-Chain Constraints for Dual Redundant Manipulators

The general situation of tight coordination operation of dual redundant manipulators is that the left manipulator and the right manipulator cooperate with each other to grip a workpiece and then move it along a desired path. In this situation, a closed-chain structure is formed between the left manipulator, the right manipulator and the workpiece. There are strict constraints between them, which are called the closed-chain constraints. The transportation of a workpiece is taken as an example to describe the closed-chain constraints in the tight coordination operation of dual redundant manipulators, as shown in Figure 1. The left manipulator is taken as the benchmark in the following description.

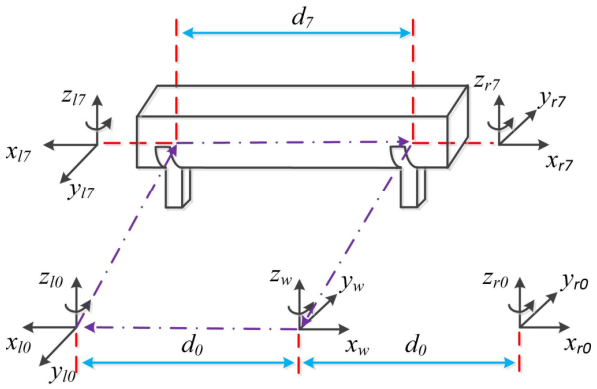

Figure 1. Closed-chain constraints for dual redundant manipulators.

2.1.1. Terminal Position and Posture Constraints

It can be seen from the purple dotted line box in Figure 1 that there is a closed-chain relationship between the terminal position of the left manipulator and the terminal position of the right manipulator. Therefore, the closed-chain constraint between them can be obtained according to the addition rule of vectors [32,33], as described below:

$$d_0 + R_w^{l0} p_{l0}^{l7}\left(q^l\right) + R_w^{l0} R_{l0}^{l7}\left(q^l\right) d_7 = p_w^{r7}(q^r) \tag{1}$$

where d_0 is the position vector of the base coordinate frame of the left manipulator in the world coordinate frame, R_w^{l0} is the posture transformation matrix of the base coordinate frame of the left manipulator with respect to the world coordinate frame, $p_{l0}^{l7}\left(q^l\right)$ is the position vector of the terminal coordinate frame of the left manipulator in its base coordinate frame, $R_{l0}^{l7}\left(q^l\right)$ is the posture transformation matrix of the terminal coordinate frame of the left manipulator with respect to its base coordinate frame, d_7 is the distance vector between the terminal coordinate frames of the left manipulator and the right manipulator in the terminal coordinate frame of the left manipulator and $p_w^{r7}(q^r)$ is the position vector of the terminal coordinate frame of the right manipulator in the world coordinate frame.

According to the transformation relation between the coordinate frames at the four vertices of the purple dotted line box in Figure 1, the constraint between the terminal

posture of the left manipulator and the terminal posture of the right manipulator can be obtained [32,33], as described below:

$$R_w^{r0} R_{r0}^{r7}(q^r) = R_w^{l0} R_{l0}^{l7}\left(q^l\right) R_{l7}^{r7} \tag{2}$$

where R_w^{r0} is the posture transformation matrix of the base coordinate frame of the right manipulator with respect to the world coordinate frame, $R_{r0}^{r7}(q^r)$ is the posture transformation matrix of the terminal coordinate frame of the right manipulator with respect to its base coordinate frame and R_{l7}^{r7} is the posture transformation matrix of the terminal coordinate frame of the right manipulator with respect to the terminal coordinate frame of the left manipulator.

2.1.2. Terminal Linear and Angular Velocity Constraints

The constraint between the terminal linear velocity of the left manipulator and the terminal linear velocity of the right manipulator can be obtained through the derivation of Equation (1) [32,33], as described below:

$$\left[R_w^{l0} J_l\left(q^l\right) + L\left(q^l\right)\right]\dot{q}^l = R_w^{r0} J_l(q^r)\dot{q}^r \tag{3}$$

where $L\left(q^l\right) = \frac{\partial\left[R_w^{l7}(q^l)d_7\right]}{\partial q^l}$, $J_l\left(q^l\right)$ is the first three rows of the Jacobian matrix of the left manipulator, $J_l(q^r)$ is the first three rows of the Jacobian matrix of the right manipulator, $\dot{q}^l$ is the joint angular velocity of the left manipulator and $\dot{q}^r$ is the joint angular velocity of the right manipulator.

For the transportation of a workpiece, the posture transformation matrix R_{l7}^{r7} is a constant matrix. Therefore, the angular velocity of the terminal coordinate frame of the right manipulator with respect to the terminal coordinate frame of the left manipulator is 0. That is, the terminal angular velocity of the right manipulator is equal to the terminal angular velocity of the left manipulator. The constraint between the terminal angular velocity of the left manipulator and the terminal angular velocity of the right manipulator is described below [32,33]:

$$R_w^{l0}\omega_l = R_w^{r0}\omega_r \tag{4}$$

where $\omega_l = J_a\left(q^l\right)\dot{q}^l$, $\omega_r = J_a(q^r)\dot{q}^r$, ω_l is the terminal angular velocity of the left manipulator in its base coordinate frame, ω_r is the terminal angular velocity of the right manipulator in its base coordinate frame, $J_a\left(q^l\right)$ is the last three rows of the Jacobian matrix of the left manipulator and $J_a(q^r)$ is the last three rows of the Jacobian matrix of the right manipulator.

The constraint between the terminal generalized velocity of the left manipulator and the terminal generalized velocity of the right manipulator can be obtained by combining constraint (3) with constraint (4) [32], as described below:

$$\dot{q}^r = J^+(q^r)\left(R_w^0\right)^{-1} J_n\left(q^l\right)\dot{q}^l \tag{5}$$

where $R_w^0 = \begin{bmatrix} R_w^{r0} & O \\ O & R_w^{r0} \end{bmatrix}$, $J_n\left(q^l\right) = \begin{bmatrix} R_w^{l0} J_l\left(q^l\right) + L\left(q^l\right) \\ R_w^{l0} J_a\left(q^l\right) \end{bmatrix}$, $J^+(q^r)$ is the pseudo inverse of the Jacobian matrix of the right manipulator.

2.2. Master-Slave Planning Based on B-RRT for Dual Redundant Manipulators*

For dual redundant manipulators, B-RRT* establishes a tree T_1 at the initial node q_{init} and another tree T_2 at the goal node q_{goal}, respectively. The root node of the tree T_1 is q_{init} and the root node of the tree T_2 is q_{goal}. Then, two trees are extended successively until they are connected to each other, as shown in Algorithm 1.

Algorithm 1. Bidirectional RRT*

Input: the initial node q_{init} and the goal node q_{goal}
Output: the desired path σ
1. T_1.init(q_{init}); T_2.init(q_{goal});
2. **For** i = 1 **To** N
3. $(T_1, flag_1, q_{e1}, q_{e2}) \leftarrow$ Extend(T_1, T_2);
4. **If** $flag_1 = true$
5. **Break**;
6. **End If**
7. $(T_2, flag_2, q_{e1}, q_{e2}) \leftarrow$ Extend(T_2, T_1);
8. **If** $flag_2 = true$
9. **Break**;
10. **End If**
11. **End For**
12. **If** $flag_1 = true$ **or** $flag_2 = true$
13. $\sigma \leftarrow$ ExtractPath (T_1, T_2, q_{init}, q_{goal}, q_{e1}, q_{e2});
14. **Else**
15. $\sigma \leftarrow \emptyset$;
16. **End If**
17. **Return** σ;

Assuming that both the left manipulator and the right manipulator have n degrees of freedom, the dual redundant manipulator can be regarded as a whole, which is equivalent to a single manipulator with $2n$ degrees of freedom. In this case, the nodes of two trees can be expressed as below:

$$q = \begin{bmatrix} q^l & q^r \end{bmatrix} \tag{6}$$

where $q^l = \begin{bmatrix} q_1^l & \cdots & q_i^l & \cdots & q_n^l \end{bmatrix}$, $q^r = \begin{bmatrix} q_1^r & \cdots & q_i^r & \cdots & q_n^r \end{bmatrix}$, q is the node of one tree of the dual redundant manipulator, q^l is the set of n joint angles of the left manipulator, q^r is the set of n joint angles of the right manipulator, q_i^l is the ith joint angle of the left manipulator and q_i^r is the ith joint angle of the right manipulator.

The extension processes of two trees are identical. Hence, the extension process of the tree T_1 is taken as an example to elaborate the details, as shown in Algorithm 2.

Algorithm 2 $(T_1, flag, q_{e1}, q_{e2}) \leftarrow$ Extend(T_1, T_2)

1. $flag \leftarrow false$;
2. $q_{rand} \leftarrow$ Sample(q_{limit});
3. $q_{near1} \leftarrow$ Near(q_{rand}, T_1);
4. $q_{new1} \leftarrow$ Steer(q_{rand}, q_{near1});
5. $feasibility \leftarrow$ DetectCollision(q_{new1});
6. **If** $feasibility = true$
7. $T_1 \leftarrow$ Insert(q_{new1}, q_{near1});
8. $T_1 \leftarrow$ ReselectParentNode(q_{new1}, q_{near1}, T_1);
9. $T_1 \leftarrow$ Rewire(q_{new1}, T_1);
10. $(flag, q_{e1}, q_{e2}) \leftarrow$ JudgeConnectivity(q_{new1}, T_2);
11. **While** $feasibility = true$ **And** $flag = false$
12. $q_{near2} \leftarrow$ Near(q_{new1}, T_2);
13. $q_{new2} \leftarrow$ Steer(q_{new1}, q_{near2});
14. $feasibility \leftarrow$ DetectCollision(q_{new2});
15. **If** $feasibility = true$
16. $T_2 \leftarrow$ Insert(q_{new2}, q_{near2});
17. $T_2 \leftarrow$ ReselectParentNode(q_{new2}, q_{near2}, T_2);
18. $T_2 \leftarrow$ Rewire(q_{new2}, T_2);
19. $(flag, q_{e1}, q_{e2}) \leftarrow$ JudgeConnectivity(q_{new2}, T_1);
20. **End If**
21. **End While**
22. **End If**

Firstly, a sampling point q_{rand} can be obtained by random sampling within the motion range of the dual redundant manipulator. Secondly, the node q_{near1} that is nearest to the sampling point q_{rand} is searched in the tree T_1. Thirdly, a new node q_{new1}, whose parent node is q_{near1}, can be obtained by expanding, as shown in Algorithm 3. Fourthly, the collision between the dual redundant manipulator and obstacles at the new node q_{new1} is detected. If a collision occurs, the new node q_{new1} will be abandoned and then the tree T_2 is switched to extend. Conversely, if no collision occurs, the new node q_{new1} will be added to the tree T_1. Fifthly, the parent node of the new node q_{new1} is reselected in its neighborhood. Its new parent node is the node with the lowest path cost. The path cost of a node is the sum of the path length from the root node q_{init} to this node and the path length from this node to the new node q_{new1}. Sixthly, all nodes in the neighborhood of the new node q_{new1} are rewired. The new node q_{new1} is taken as the parent node of these nodes and then their new path costs are calculated. If the new path costs are less than the old path costs, the parent nodes of these nodes are changed to the new node q_{new1}. On the contrary, the parent nodes of these nodes remain unchanged. Seventhly, the connectivity of the two trees is checked, as shown in Algorithm 4. If the distance between the new node q_{new1} and some node of the tree T_2 is less than the connection threshold, the two trees have been connected to each other and then the extensions of two trees are stopped. If the distance between the new node q_{new1} and any node of the tree T_2 is greater than the connection threshold, the tree T_2 is extended towards the new node q_{new1} until there is a collision or the two trees are connected to each other. In the case that there is a collision, the tree T_2 is switched to perform the above steps. Finally, in the case that the extensions of two trees are completed, if the two trees are connected to each other, the desired path can be extracted from the two trees and it will be the output of the algorithm. Otherwise, the empty set is directly returned as the output of the algorithm.

Algorithm 3 $q_{new} \leftarrow \text{Steer}(q_{target}, q_{near})$

1. $(q^l_{target}, q^r_{target}) \leftarrow \text{Decompose}(q_{target})$;
2. $(q^l_{near}, q^r_{near}) \leftarrow \text{Decompose}(q_{near})$;
3. $q^l_{new} \leftarrow \text{Expand}(q^l_{target}, q^l_{near})$;
4. $\dot{q}^l \leftarrow (q^l_{new} - q^l_{near})/t$;
5. $\dot{q}^r \leftarrow \text{Constraint}(\dot{q}^l, q^l_{near}, q^r_{near})$;
6. $q^r_{new} \leftarrow q^r_{near} + t \cdot \dot{q}^r$;
7. $q_{new} \leftarrow \text{Merge}(q^l_{new}, q^r_{new})$;

Algorithm 4 $(flag, q_{e1}, q_{e2}) \leftarrow \text{JudgeConnectivity}(q_{new}, T)$

1. $q_{e1} \leftarrow q_{new}$;
2. **For** $i = 1$ **To** k
3. $q_i \leftarrow \text{ExtractNode}(T, i)$;
4. **If** $\text{Distance}(q_{new}, q_i) < threshold$
5. $flag \leftarrow true$;
6. $q_{e2} \leftarrow q_i$;
7. **Break**;
8. **Else**
9. $flag \leftarrow false$;
10. **End If**
11. **End For**

In the expansion phase, for single redundant manipulators, a new node q_{new} can be obtained directly by extending a step distance δ forward in the direction from the node q_{near} to the sampling point q_{rand}. However, for dual redundant manipulators, considering the closed-chain constraints (1)–(5) between the left manipulator and the right manipulator in tight coordination, a new node q_{new} cannot be obtained by the expansion mode of single

redundant manipulators. Therefore, the master-slave planning method is adopted to extend the trees of dual redundant manipulators. It is assumed that the left manipulator is the master manipulator and the right manipulator is the slave manipulator. A new set of joint angles q_{new}^{l} of the left manipulator is generated by the expansion mode of single redundant manipulators. Then, the corresponding set of joint angles q_{new}^{r} of the right manipulator can be calculated according to the closed-chain constraints. Lastly, a new node q_{new} of the dual redundant manipulator can be obtained by merging the two sets of joint angles q_{new}^{l} and q_{new}^{r}.

The closed-chain constraints (1)–(5) of the dual redundant manipulator are the constraints between the terminal states of the left manipulator and the right manipulator. Therefore, the general execution process of the master-slave planning method is described as follows. After obtaining the set of joint angles of the left manipulator, the terminal position and posture of the left manipulator are calculated through forward kinematics. Then, the terminal position and posture of the right manipulator can be calculated through the closed-chain constraints (1) and (2). Finally, the set of joint angles of the right manipulator is solved by inverse kinematics [28,29]. The numbers of degrees of freedom of the left manipulator and the right manipulator are both greater than 6. It results in the calculation of the analytic solution of inverse kinematics being very tedious. Hence, the inverse kinematics is usually solved by numerical algorithms. Numerical algorithms are generally based on the relationship (7) between the terminal generalized velocity of the redundant manipulator and its joint angular velocity. The joint angular velocity of the redundant manipulator is calculated by the numerical algorithm, and then its corresponding set of joint angles can be calculated accordingly [30,31].

$$\dot{q} = J^{+}(q)V + \left[I_{n\times n} - J^{+}(q)J(q)\right]\varepsilon \tag{7}$$

where V is the terminal generalized velocity vector of the redundant manipulator, $J^{+}(q)$ is the pseudo inverse of the Jacobian matrix and ε is a vector in the null space.

The closed-chain constraints (3)–(5) of the dual redundant manipulator are essentially the constraints between the terminal generalized velocities of the left manipulator and the right manipulator, but they have been transformed into the constraints between the joint angular velocities of the left manipulator and the right manipulator through the Jacobian matrix. To simplify the calculation of the set of joint angles of the right manipulator, the joint angular velocity of the left manipulator is calculated after obtaining the set of joint angles of the left manipulator. Then, the joint angular velocity of the right manipulator can be calculated through the closed-chain constraints (3)–(5), and the set of joint angles of the right manipulator can be calculated accordingly, as shown in Algorithm 3. This method can avoid the calculation of the inverse kinematics solution and can effectively improve the planning efficiency and shorten the planning time. The specific steps are described as follows:

(I) In the direction from the node $q_{near}^{l}$of the left manipulator to its temporary target point q_{target}^{l}, the new node q_{new}^{l} can be obtained by extending a step distance δ forward, as described below:

$$q_{new}^{l} = q_{near}^{l} + \delta \tag{8}$$

where q_{new}^{l} is the new node of the left manipulator and q_{near}^{l} is the node closest to the temporary target point q_{target}^{l} of the left manipulator.

(II) The motion time t between two adjacent nodes of the dual redundant manipulator is given. Assuming that the left manipulator moves at a uniform speed between the two adjacent nodes in the joint space, the joint angular velocity $\dot{q}^{l}$ of the left manipulator can be calculated by Equation (9).

$$\dot{q}^{l} = \frac{q_{new}^{l} - q_{near}^{l}}{t} \tag{9}$$

where $\dot{q}^{l}$ is the joint angular velocity of the left manipulator and t is the motion time between two adjacent nodes of the dual redundant manipulator.

(III) According to the closed-chain constraint (5), the joint angular velocity $\dot{q}^r$ of the right manipulator is calculated.

(IV) Similarly, assuming that the right manipulator moves at a uniform speed between the two adjacent nodes in the joint space, the new node q^r_{new} of the right manipulator can be calculated by Equation (10).

$$q^r_{new} = q^r_{near} + t \cdot \dot{q}^r \tag{10}$$

where q^r_{new} is the new node of the right manipulator, q^r_{near} is the node closest to the temporary target point q^r_{target} of the right manipulator and $\dot{q}^r$ is the joint angular velocity of the right manipulator.

(V) The new node q^l_{new} of the left manipulator and the new node q^r_{new} of the right manipulator are merged to obtain the new node $q_{new} = [q^l_{new}\ q^r_{new}]$ of the dual redundant manipulator.

To ensure that the new node q_{new} can satisfy the closed-chain constraints (1) and (2), the step distance δ and the motion time t should not be too large. That is, the motion between the two adjacent nodes should be a differential motion.

2.3. Local Path Replanning

The connection condition of two trees is that the distance between some node q_{1i} of the tree T_1 and some node q_{2j} of the tree T_2 is less than the given threshold, as shown in Algorithm 4. In the case that B-RRT* is applied to dual redundant manipulators, its node q contains both the set of joint angles q^l of the left manipulator and the set of joint angles q^r of the right manipulator. In the actual planning, when the distance between two sets of joint angles of the left manipulator (or the right manipulator) in different trees is less than the given threshold, the distance between two sets of joint angles of the right manipulator (or the left manipulator) in different trees may be not less than the given threshold in a high probability. That is, it is difficult to meet the connection condition of two trees. Therefore, it is necessary to modify the connection condition of two trees for solving the above problem.

Since it is difficult for the left manipulator and the right manipulator to meet the connection condition at the same time, it can first ensure that the parts corresponding to the left manipulator in two nodes of different trees are connected with each other, as shown in Algorithm 5. Then, the local path between the parts corresponding to the right manipulator in two nodes of different trees is replanned. That is, while keeping the left manipulator stationary, the right manipulator moves from the set of joint angles corresponding to the connection node of the tree T_1 (or T_2) to the set of joint angles corresponding to the connection node of the tree T_2 (or T_1). Moreover, to satisfy the closed-chain constraints (1) and (2), the terminal position and posture of the right manipulator must always remain unchanged during the movement.

Algorithm 5 ($flag$, q_{e1}, q_{e2}) ← NewJudgeConnectivity(q_{new}, T)

```
1.  q_e1 ← q_new;
2.  (q^l_new, q^r_new) ← Decompose(q_new);
3.  For i = 1 To k
4.      q_i ← ExtractNode(T, i);
5.      (q^l_i, q^r_i) ← Decompose(q_i);
6.      If  Distance(q^l_new, q^l_i) < threshold
7.          flag ← true;
8.          q_e2 ← q_i;
9.          Break;
10.     Else
11.         flag ← false;
12.     End For
13. End For
```

To ensure that the terminal position and posture of the right manipulator remains unchanged during the local path replanning, it can be realized through the joint self-motion in the null space [34,35] on the basis of the characteristics of redundant manipulators. That is, the joint angular velocity of the right manipulator is calculated by setting the terminal generalized velocity V in Equation (7) to zero, as described below:

$$\dot{q}^r = \left(I_{n\times n} - J^+(q^r)J(q^r)\right)\varepsilon^r \tag{11}$$

where ε^r is a vector in the null space of the right manipulator.

To ensure that the right manipulator can start from the starting point $q^r_{localstart}$ of the local path and move towards the end point $q^r_{localgoal}$ of the local path, the value of vector ε^r can be calculated by Equation (12).

$$\varepsilon^r = \frac{\left(q^r_{localgoal} - q^r_k\right)}{(m-k+1)t} \tag{12}$$

where $m = \frac{\left(q^r_{localgoal} - q^r_{localstart}\right)}{\delta}$, $k = 1 \sim m$, $q^r_{localstart}$ is the starting point of the local path of the right manipulator, $q^r_{localgoal}$ is the end point of the local path of the right manipulator, m is the upper limit of the number of discrete points in the local path of the right manipulator and k is the serial number of the current discrete point in the local path of the right manipulator.

The process of local path replanning is shown in Algorithm 6. In addition, the extension process of bidirectional RRT* with local path replanning (B-RRT*-LPR) is shown in Algorithm 7.

Algorithm 6 $(T, success, q_{e1}) \leftarrow$ Localpath(q_{e1}, q_{e2}, T)

```
1.  success ← false;
2.  (q^l_localstart, q^r_localstart) ← Decompose(q_e1);
3.  (q^l_localgoal, q^r_localgoal) ← Decompose(q_e2);
4.  q_1 ← q_e1; q^r_1 ← q^r_localstart;
5.  For k = 1 To m − 1
6.      J^r_k ← Jacobian(q^r_k);
7.      ε^r_k ← NullSpaceVector(q^r_k, q^r_localgoal, k);
8.      q̇^r_k ← Velocity(J^r_k, ε^r_k);
9.      q^r_{k+1} ← q^r_k + t q̇^r_k
10.     q_{k+1} ← Merge(q^l_localstart, q^r_{k+1} r k;
11.     feasibility ← DetectCollision(q_{k+1});
12.     If feasibility = true
13.         T ← Insert(q_{k+1}, q_k);
14.         q_k ← q_{k+1};
15.         If Distance(q^r_{k+1}, q^r_localgoal) < threshold
16.             q_e1 ← q_{k+1};
17.             success ← true;
18.             Break;
19.         End If
20.     End If
21. End For
```

The specific steps of local path replanning are described as follows:

(a) The parts corresponding to the right manipulator in the connection node of the tree T_1 (or T_2) are selected as the starting point $q^r_{localstart}$ of the local path. The parts corresponding to the right manipulator in the connection node of the tree T_2 (or T_1) are selected as the end point $q^r_{localgoal}$ of the local path. The first discrete point q^r_1 of the local path is $q^r_{rlocalstart}$.

(b) The Jacobian matrix J_k^r and the vector ε_k^r of the current discrete point q_k^r are calculated. Then, the joint angular velocity $\dot{q}_k^r$of the right manipulator is calculated according to Equation (11).

(c) The next discrete point q_{k+1}^r is calculated by Equation (13).

$$q_{k+1}^r = q_k^r + t \cdot \dot{q}_k^r \tag{13}$$

where q_{k+1}^r is the next discrete point of the local path of the right manipulator, q_k^r is the current discrete point of the local path of the right manipulator and $\dot{q}_k^r$ is the joint angular velocity of the right manipulator at the current discrete point of the local path.

Algorithm 7 $(T_1, flag, q_{e1}, q_{e2}) \leftarrow$ NewExtend(T_1, T_2)

```
flag ← false;
q_rand ← Sample(q_limit);
q_near1 ← Near(q_rand, T_1);
q_new1 ← Steer(q_rand, q_near1);
feasibility ← DetectCollision(q_new1);
If feasibility = true
   T_1 ← Insert(q_new1, q_near1);
   T_1 ← ReselectParentNode(q_new1, q_near1, T_1);
   T_1 ← Rewire(q_new1, T_1);
   (flag, q_e1, q_e2) ← NewJudgeConnectivity(q_new1, T_2);
   If flag = true
      (T_1, flag, q_e1) ← Localpath(q_e1, q_e2, T_1);
   End If
   While feasibility = true And flag = false
         q_near2 ← Near(q_new1, T_2);
         q_new2 ← Steer(q_new1, q_near2);
         feasibility ← DetectCollision(q_new2);
         If feasibility = true
            T_2 ← Insert(q_new2, q_near2);
            T_2 ← ReselectParentNode(q_new2, q_near2, T_2);
            T_2 ← Rewire(q_new2, T_2);
            (flag, q_e2, q_e1) ← NewJudgeConnectivity(q_new2, T_1);
            If flag = true
               (T_2, flag, q_e2) ← Localpath(q_e2, q_e1, T_2);
            End If
         End If
   End While
End If
```

(d) The set of joint angles $q_{localstart}^l$ of the left manipulator corresponding to the starting point $q_{localstart}^r$ of the local path of the right manipulator is merged with the next discrete point q_{k+1}^r of the local path of the right manipulator to obtain the next discrete point $q_{k+1} = [\, q_{llocalstart}^l \; q_{k+1}^r]$ of the local path of the dual redundant manipulator.

(e) The collision detection of the next discrete point q_{k+1} of the local path of the dual redundant manipulator is carried out. If the dual redundant manipulator does not collide, step (f) is performed. If the dual redundant manipulator collides, the local path replanning is stopped and the result that the node $q_{localstart} = [q \; q_{localstart}^r]$ and the node $q_{localgoal} = [q_{localgoal}^l \; q_{localgoal}^r]$ cannot be connected with each other is returned.

(f) The next discrete point q_{k+1} of the local path of the dual redundant manipulator is added to the tree T_1 (or T_2).

(g) The distance between the next discrete point $q_{k+1}^r$of the local path of the right manipulator and the end point$q_{rlocalgoal}^r$ of the local path of the right manipulator is judged to determine whether it is less than the given threshold. Alternatively, the serial

number k of the current discrete point is judged to determine whether it is greater than the upper limit m of the number of discrete points. If the distance is less than the given threshold or the serial number k is greater than the upper limit m, the local path replanning is stopped and the result that the node $q_{localstart}$ and the node $q_{localgoal}$ can be connected with each other is returned. If the distance is not less than the given threshold value and the serial number k is not greater than the upper limit m, the current discrete point of the local path of the right manipulator is replaced by q_k^r+1, and then step (b) is performed again.

3. Results

To verify the effectiveness of B-RRT*-LPR for dual redundant manipulators, the Baxter dual redundant manipulator is taken as the experimental object to plan its motion path of tight coordination operations. The experimental platform is built for the situation of collaborative transportation, as shown in Figure 2. There are two obstacles in the environment. Obstacle 1 is a dark yellow paper box suspended in the air, and obstacle 2 is a white storage box placed on the desktop. The workpiece is an industrial aluminum profile whose length, width and height are 0.235 m, 0.04 m and 0.04 m, respectively. A simulation platform, whose size is consistent with the actual size, is built accordingly, as shown in Figure 3. In Figure 3, the yellow cuboids are the obstacles. Then, the simulations are carried out by Matlab 2014a, which runs on a computer with an Intel(R) Core(TM) i7-4710HQ quad-core processor, 2.50 GHz main frequency and 8G memory. Finally, the experiment is carried out according to the simulation results.

Figure 2. Experimental platform: (**a**) initial state; (**b**) goal state.

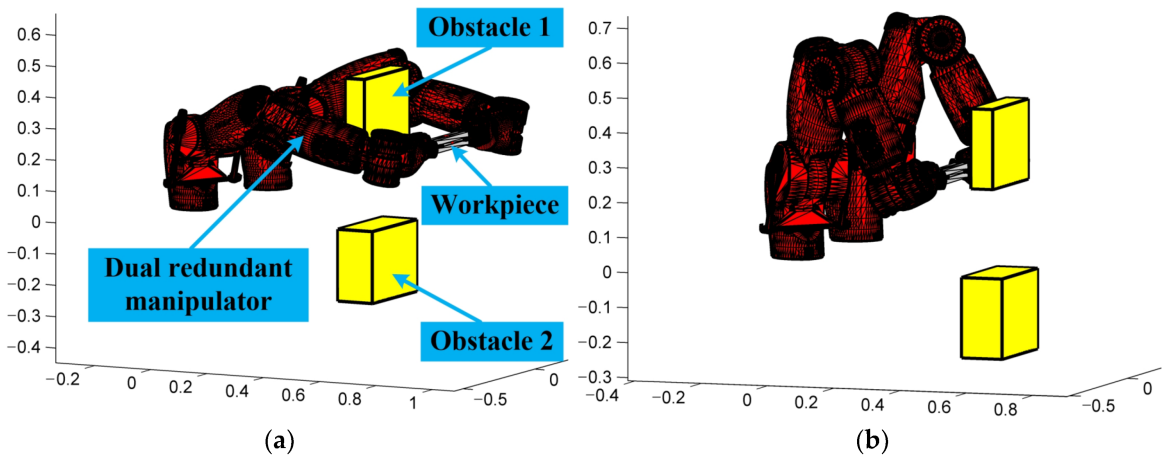

Figure 3. Simulation platform: (**a**) initial state; (**b**) goal state.

3.1. Simulation

Considering the randomness of sampling-based algorithms, B-RRT* and B-RRT*-LPR are adopted to plan 10 groups of paths in the case that the connection threshold is 1°, 5°, 10°, 15°, 20°, 25° and 30°, respectively. The planning results are arranged in ascending order according to the time cost of each group of paths, as shown in Figure 4. The maximum number of iterations of the two algorithms is set to 500. That is, if no feasible path is obtained after 500 iterations, the planning fails.

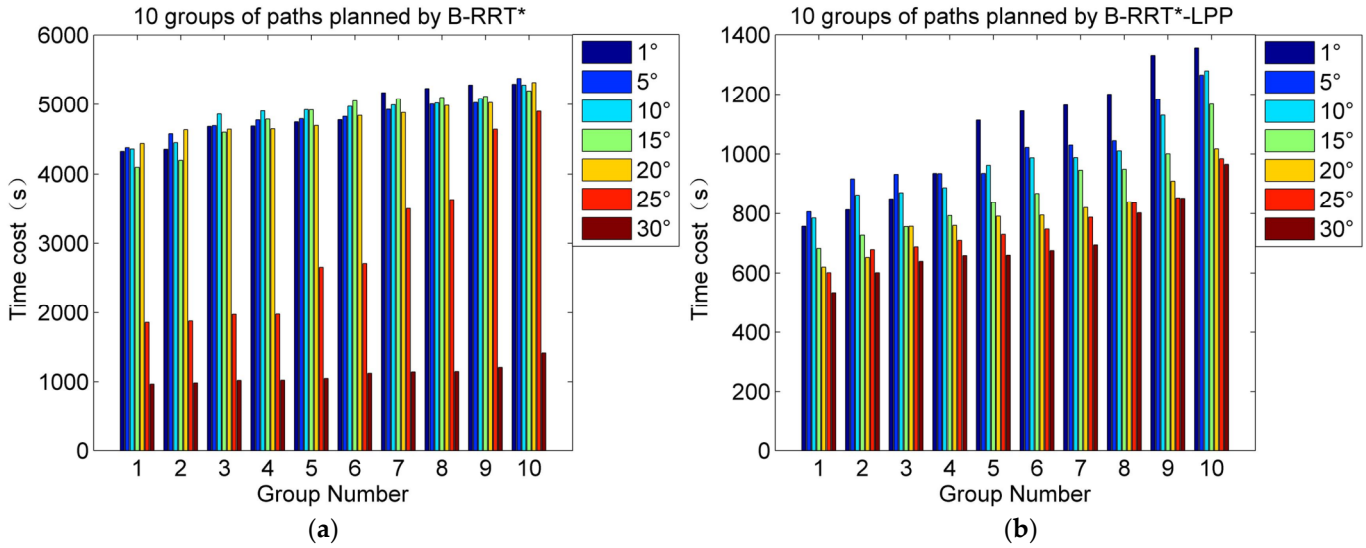

Figure 4. Time cost: (**a**) B-RRT*; (**b**) B-RRT*-LPR.

The maximum time cost, minimum time cost, average time cost and success rate of 10 groups of planning results of the 2 algorithms with different connection thresholds are shown in Tables 1 and 2.

Table 1. Planning results of B-RRT*.

	t_{max} (s)	t_{min} (s)	t_{avg} (s)	sr
Th: 1 (°)	5294.087	4323.168	4855.917	0%
Th: 5 (°)	5376.889	4379.025	4838.732	0%
Th: 10 (°)	5283.419	4359.348	4885.978	0%
Th: 15 (°)	5197.209	4097.994	4815.204	0%
Th: 20 (°)	5316.107	4438.916	4811.795	20%
Th: 25 (°)	4902.081	1856.069	2972.804	80%
Th: 30 (°)	1414.534	963.751	1106.657	100%

where t_{min}, t_{max} and t_{avg} are the minimum, maximum and average time costs of 10 groups of paths, sr is the success rate of 10 groups of paths and Th is the connection threshold that is used to judge the connectivity of two trees.

Table 2. Planning results of B-RRT*-LPR.

	t_{max} (s)	t_{min} (s)	t_{avg} (s)	sr
Th: 1 (°)	1355.623	756.654	1066.326	100%
Th: 5 (°)	1264.872	805.944	1006.879	100%
Th: 10 (°)	1279.273	784.542	976.011	100%
Th: 15 (°)	1166.911	682.488	872.420	100%
Th: 20 (°)	1017.361	619.926	795.953	100%
Th: 25 (°)	984.236	601.268	761.544	100%
Th: 30 (°)	965.477	534.450	708.229	100%

It can be seen from Figure 4 and Tables 1 and 2 that the success rate of B-RRT* is 0% in the case that the connection threshold is less than or equal to 15°. With the increase of the connection threshold, the success rate gradually increases until it reaches 100% when the connection threshold is 30°. However, the success rate of B-RRT*-LPR is 100% in the case that the connection threshold is any one of the seven given values. Therefore, the results show that the local path replanning strategy can effectively solve the discontinuity problem in the joint path of the slave manipulator at the connection nodes in the case that B-RRT* is applied to dual redundant manipulators. It can significantly improve the success rate.

The success rate of B-RRT* is 100% in the case that the connection threshold is 30°, but its average time cost is 1106.657 s. On the contrary, with the same connection threshold, B-RRT*-LPR can ensure that the success rate is 100%, and its average time cost is 708.229 s, which is 36.00% lower than the one of B-RRT*. Therefore, B-RRT*-LPR can effectively shorten the planning time.

In the case that B-RRT* fails to plan, its time cost is always more than 4000 s. That is, for B-RRT*, the time cost of 500 iterations is more than 4000 s. Therefore, in the case that the maximum number of iterations is greater than 500, although it is possible to improve the success rate, the time cost will be further increased. That is, if the number of iterations exceeds 500, B-RRT* may successfully plan the desired path, but the time cost must exceed 4000 s. However, B-RRT*-LPR spends no more than 1400 s successfully planning the desired path with any connection threshold. Therefore, instead of increasing the maximum number of iterations to improve the success rate of B-RRT*, it is better to directly adopt B-RRT*-LPR for path planning.

In addition, although B-RRT* can improve the success rate by increasing the connection threshold, the increase of the connection threshold means that the difference between the connection nodes of two trees will also become larger. When the connection threshold is much larger than the step distance, the local path between the two connection nodes essentially contains some other hidden nodes. These hidden nodes have not been detected whether they would collide with obstacles. Hence, they may collide with obstacles. As a result, the planned path may actually be infeasible. For B-RRT*-LPR, it can ensure that the success rate is 100% with any connection threshold. Therefore, a smaller connection threshold can be selected in the actual planning to ensure that B-RRT*-LPR will not have the above problem.

3.2. Experiment

The most representative path in the 10 groups of planning results of B-RRT*-LPR is chosen in the case that the connection threshold is 1°. Its time cost is closest to the average time cost of 10 paths. That is, the fifth path is chosen. The display of this path in the task space is shown in Figure 5. In Figure 5, the green hollow dot represents the node of two trees, the blue thin solid line represents the local path between two nodes and the red thick solid line represents the final path. The yellow cuboid suspended above is obstacle 1, and the yellow cuboid placed below is obstacle 2.

According to the path shown in Figure 5, the motion process of the dual redundant manipulator can be described as follows. At the initial stage of motion, the dual redundant manipulator is far away from obstacles 1 and 2. Hence, it moves directly towards the goal state. When the dual redundant manipulator approaches obstacle 1, it moves downward to avoid collision with obstacle 1. After moving a certain distance, the dual redundant manipulator moves towards the goal state again on the premise that it does not collide with obstacles 1 and 2. After successfully passing through the gap between obstacle 1 and obstacle 2, the dual redundant manipulator moves upward until its height is close to the height of the goal state. Then, the dual redundant manipulator moves towards the goal state until it reaches the goal state.

In the case that the dual redundant manipulator moves along the path shown in Figure 5, its actual motion process is shown in Figure 6.

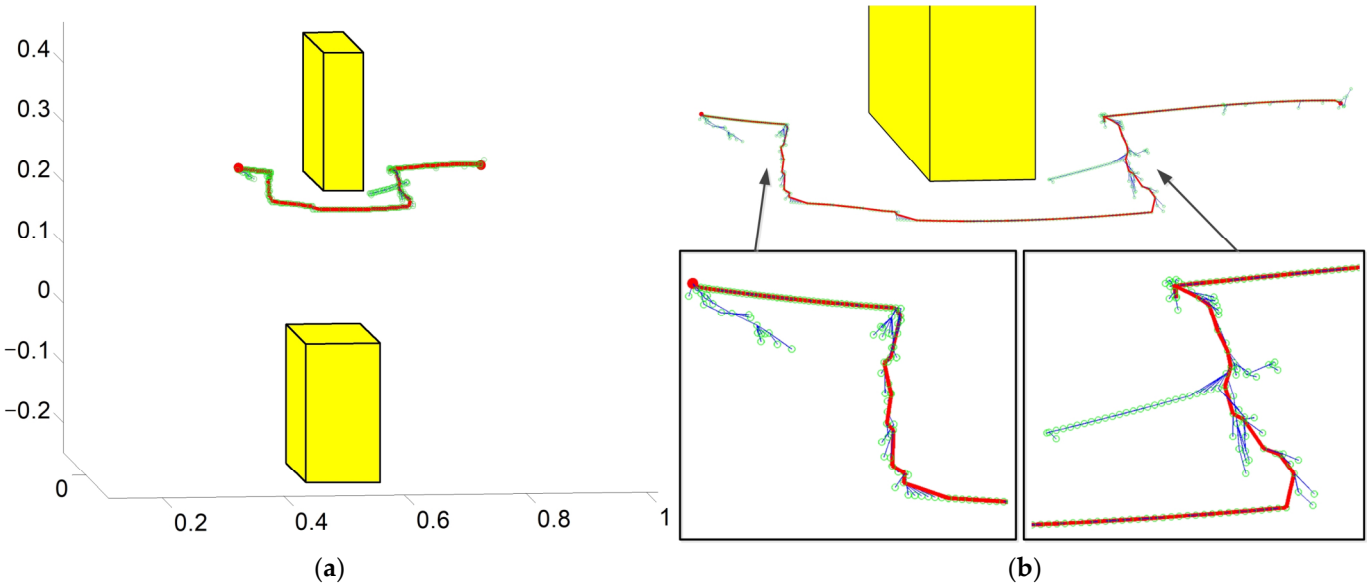

Figure 5. Path of the dual redundant manipulator in the task space: (**a**) global view; (**b**) partial enlarged view.

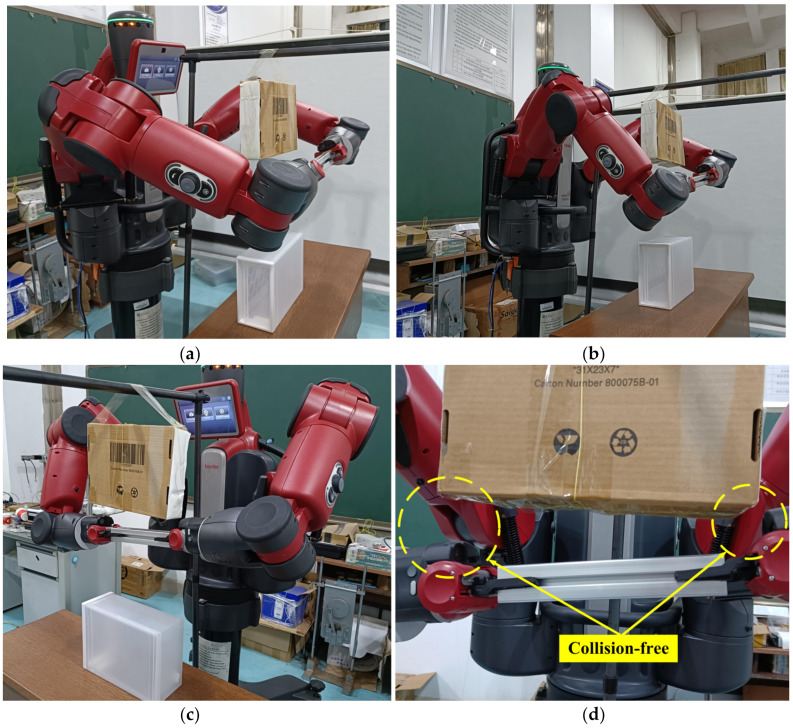

Figure 6. Actual motion process: (**a**) connection node of the tree T_1; (**b**) connection node of the tree T_2; (**c**) global view of a node located between two obstacles; (**d**) partial enlarged view of a node located between two obstacles.

It can be seen from Figure 6 that the dual redundant manipulator starts from the initial state, as shown in Figure 2a. Then, the dual redundant manipulator moves to the vicinity of obstacle 1 in the direction of the goal state, as shown in Figure 6a. The state of the dual redundant manipulator in Figure 6a corresponds to the connection node of the tree T_1. Afterwards, the dual redundant manipulator moves along the local path planned by the local path replanning strategy until it reaches the state corresponding to the connection node of the tree T_2, as shown in Figure 6b. The terminal positions and postures of both the left manipulator and the right manipulator remain unchanged during the local motion

shown in Figure 6a,b. After that, the dual redundant manipulator moves downward for a distance and then passes through the gap between obstacle 1 and obstacle 2 without collision, as shown in Figure 6c,d. Finally, the dual redundant manipulator can reach the goal state successfully, as shown in Figure 2b.

The above results show that the dual redundant manipulator can move according to the desired path and satisfy the closed-chain constraints in the actual motion. The actual path of the dual redundant manipulator is continuous in both the task space and the joint space. Therefore, B-RRT*-LPR can effectively plan a collision-free path for tight coordination operation of the dual redundant manipulator.

4. Discussions

4.1. Innovations

For RRT and its derivative algorithms, they are generally applied to plan the path for loose coordination of dual redundant manipulators, such as assembly [36,37] and grasp [38,39]. In these cases, there are three most common planning strategies, as described as follows: (1) RRT or its derivative algorithm is adopted to plan the paths of the left manipulator and the right manipulator in parallel. After obtaining the paths of the two manipulators, collision detection between the two paths is performed [23,24]. (2) RRT or its derivative algorithm is adopted to plan the path of one manipulator first, and then the path of another manipulator. During the planning of the second manipulator, the first manipulator is treated as a dynamic obstacle [19]. (3) The dual redundant manipulator is treated as a single redundant manipulator with $2n$ degrees of freedom. Then, the path of this single redundant manipulator is planned by RRT or its derivative algorithm. Collision detection between the left manipulator and the right manipulator can be transformed into self-collision detection of the single redundant manipulator [20].

For tight coordination of dual redundant manipulators, there are kinematic closed-chain constraints between the left manipulator and the right manipulator. The terminal positions and postures of the left manipulator and the right manipulator are mutually restricted. However, these kinematic closed-chain constraints are not considered for the above three planning strategies. Therefore, the above three planning strategies are not applicable to tight coordination of dual redundant manipulators. To solve this problem, a master-slave planning method based on B-RRT* is proposed in our study. The path of the master manipulator is directly planned by B-RRT*. Then, the path of the slave manipulator is calculated by the closed-chain constraints. The experimental and simulation results show that the closed-chain constraints can always be satisfied during the motion of the dual redundant manipulator. Therefore, the master-slave planning method based on B-RRT* can be effectively applied to tight coordination of dual redundant manipulators.

Due to the redundant characteristics of dual redundant manipulators, the master-slave planning method based on B-RRT* will cause a discontinuity problem in the joint path of the slave manipulator at the connection nodes. This problem will result in the actual motion path being infeasible. To solve this problem, a local path replanning strategy is designed in our study. The joints of the slave manipulator move only in the null space for satisfying the closed-chain constraints. The experimental and simulation results show that the dual redundant manipulator can move continuously in both the task space and the joint space. Therefore, the local path replanning strategy can effectively solve the discontinuity problem.

Moreover, since the closed-chain constraints act on the terminal state of the dual redundant manipulator, the calculation of the path of the slave manipulator is generally to calculate its terminal position and posture first, and then calculate its joint angles through inverse kinematics [28,29]. For redundant manipulators, the inverse kinematics is usually solved by numerical algorithms [30,31]. However, numerical algorithms generally have the disadvantages of large amounts of calculation and high time cost [40]. To avoid the calculation of inverse kinematics, the terminal generalized velocity constraints are directly adopted to obtain the joint angular velocity of the slave manipulator in our study. Then,

the joint angles of the slave manipulator can be calculated accordingly. Therefore, it can effectively improve the execution efficiency.

4.2. Extended Applications

In our study, the master-slave planning method on B-RRT* is proposed to plan the path for tight coordination of dual redundant manipulators. However, for the path planning of the master manipulator, B-RRT* is not the only choice; other algorithms, such as other derivative algorithms [41,42] of RRT and PRM [43,44], or an artificial potential field method [45,46], can also be chosen.

The local path replanning strategy can also be applied to realize the obstacle avoidance [47,48] of redundant manipulators. In the case that the end-effector does not collide with obstacles, but some joints collide with obstacles, the local path replanning strategy can be adopted to make these joints move away from or around obstacles on the premise that the end-effector remains stationary. In this way, it is not necessary to replan the entire path, but only to replan the local path of joints.

4.3. Limitations

Both the master-slave planning method based on B-RRT* and the local path replanning strategy take advantage of the unique kinematic characteristics of redundant manipulators in the design process. Therefore, the proposed methods are not applicable to dual nonredundant manipulators, but only to dual redundant manipulators.

5. Conclusions

To effectively plan a safe path for tight coordination operations of dual redundant manipulators, a bidirectional RRT* satisfying closed-chain constraints is proposed. B-RRT* is adopted to plan the path of the master manipulator, and the path of the slave manipulator is then obtained based on the closed-chain constraints. When the connectivity between the two nodes of different trees is checked, only the parts corresponding to the master manipulator are checked. The connectivity between the parts corresponding to the slave manipulator is guaranteed by the local path replanning strategy. The simulation results show that, compared with B-RRT*, the proposed method can effectively improve the success rate when the connection threshold is small and can shorten the planning time when the connection threshold is the same. The experimental results show that the dual redundant manipulator can effectively avoid obstacles and satisfy the closed-chain constraints during the motion. Therefore, the proposed method can effectively plan a collision-free path for tight coordination operations of dual redundant manipulators.

Our current study is the path planning for tight coordination of dual redundant manipulators in the static environment. That is, obstacles are stationary. The positions and postures of obstacles are known. However, if obstacles are in motion, their positions and postures will change in real time and are unknown. Hence, to apply the proposed method to the dynamic environment, it is necessary to add some external equipment to obtain the real-time positions and postures of obstacles. In the future, we will use visual sensors to obtain the information of obstacles in real time and then extract the positions and postures of obstacles through image processing technologies. On the basis of the above, the proposed method can be adopted to plan the path for tight coordination of dual redundant manipulators in the dynamic environment.

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

Funding: This research was funded in part by the National Natural Science Foundation of China under Grant 52275297, in part by the Special Key Project for Natural Disaster Prevention Technology and Equipment Engineering, Ministry of Industry and Information Technology of China under Grant TC210H00L/17, and in part by the Project of State Key Laboratory of High Performance Complex Manufacturing, Central South University under Grant ZZYJKT2021-17.

Data Availability Statement: Not applicable.

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

References

1. Guo, D.; Li, Z.; Khan, A.H.; Feng, Q.; Cai, J. Repetitive motion planning of robotic manipulators with guaranteed precision. *IEEE Trans. Ind. Inform.* **2021**, *17*, 356–366. [CrossRef]
2. Hu, C.; Lin, S.; Wang, Z.; Zhu, Y. Task space contouring error estimation and precision iterative control of robotic manipulators. *IEEE Robot. Autom. Let.* **2022**, *7*, 7826–7833. [CrossRef]
3. Patel, R.V.; Shadpey, F. *Control of Redundant Robot Manipulators*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 7–34.
4. Ning, Y.; Li, T.; Du, W.; Yao, C.; Zhang, Y.; Shao, J. Inverse kinematics and planning/control co-design method of redundant manipulator for precision operation: Design and experiments. *Robot. Com. Int. Manuf.* **2023**, *80*, 102457. [CrossRef]
5. Fan, J.; Jin, L.; Xie, Z.; Li, S.; Zheng, Y. Data-driven motion-force control scheme for redundant manipulators: A kinematic perspective. *IEEE Trans. Ind. Inform.* **2022**, *18*, 5338–5347. [CrossRef]
6. Zivanovic, M.D.; Vukobratovic, M.K. *Multi-Arm Cooperating Robots: Dynamics and Control*; Springer: Dordrecht, The Netherland, 2006; pp. 1–4.
7. Zhang, Z.; Zheng, L.; Chen, Z.; Kong, L.; Karimi, H.R. Mutual-collision-avoidance scheme synthesized by neural networks for dual redundant robot manipulators executing cooperative tasks. *IEEE Trans. Neur. Net. Lear.* **2021**, *32*, 1052–1066. [CrossRef] [PubMed]
8. Lu, Z.; Wang, N.; Shi, D. DMPs-based skill learning for redundant dual-arm robotic synchronized cooperative manipulation. *Complex Intell. Syst.* **2022**, *8*, 2873–2882. [CrossRef]
9. Gill, M.A.C.; Zomaya, A.Y. *Obstacle Avoidance in Multi-Robot Systems*; World Scientific: Singapore, 1998; pp. 30–51.
10. Choi, Y.; Kim, D.; Hwang, S.; Kim, H.; Kim, N.; Han, C. Dual-arm robot motion planning for collision avoidance using B-spline curve. *Int. J. Precis. Eng. Man.* **2017**, *18*, 835–843. [CrossRef]
11. Ni, S.; Chen, W.; Ju, H.; Chen, T. Coordinated trajectory planning of a dual-arm space robot with multiple avoidance constraints. *Acta Astronaut.* **2022**, *195*, 379–391. [CrossRef]
12. Meng, B.H.; Godage, I.S.; Kanj, I. RRT*-based path planning for continuum arms. *IEEE Robot. Autom. Let.* **2022**, *7*, 6830–6837. [CrossRef] [PubMed]
13. Becerra, I.; Yervilla-Herrera, H.; Antonio, E.; Murrieta-Cid, R. On the local planners in the RRT* for dynamical systems and their reusability for compound cost functionals. *IEEE Trans. Robot.* **2022**, *38*, 887–905. [CrossRef]
14. Thakar, S.; Rajendran, P.; Kabir, A.M.; Gupta, S.K. Manipulator motion planning for part pickup and transport operations from a moving base. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 191–206. [CrossRef]
15. Wang, J.; Li, B.; Meng, M.Q.H. Kinematic constrained bi-directional RRT with efficient branch pruning for robot path planning. *Expert Syst. Appl.* **2021**, *170*, 114541. [CrossRef]
16. Zhao, W.; Wang, X.; Liu, Y. Path planning for 5-axis CMM inspection considering path reuse. *Machines* **2022**, *10*, 973. [CrossRef]
17. Li, J.; Cui, R.; Su, P.; Ma, L.; Sun, H. A computer-assisted preoperative path planning method for the parallel orthopedic robot. *Machines* **2022**, *10*, 480. [CrossRef]
18. Yu, M.; Luo, J.; Wang, M.; Gao, D. Spline-RRT: Coordinated motion planning of dual-arm space robot. *IFAC PapersOnLine* **2020**, *53*, 9820–9825. [CrossRef]
19. Ying, K.; Pourhejazy, P.; Cheng, C.; Cai, Z. Deep learning-based optimization for motion planning of dual-arm assembly robots. *Comput. Ind. Eng.* **2021**, *160*, 107603. [CrossRef]
20. Chen, X.; You, X.; Jiang, J.; Ye, J.; Wu, H. Trajectory planning of dual-robot cooperative assembly. *Machines* **2022**, *10*, 689. [CrossRef]
21. Mashayekhi, R.; Idris, M.Y.I.; Anisi, M.H.; Ahmedy, I.; Ali, I. Informed RRT*-connect: An asymptotically optimal single-query path planning method. *IEEE Access* **2020**, *8*, 19842–19852. [CrossRef]
22. Wang, X.; Wei, J.; Zhou, X.; Xia, Z.; Gu, X. Dual-objective collision-free path optimization of arc welding robot. *IEEE Robot. Autom. Let.* **2021**, *6*, 6353–6360. [CrossRef]
23. Vahrenkamp, N.; Asfour, T.; Dillmann, R. Simultaneous grasp and motion planning. *IEEE Robot. Autom. Mag.* **2012**, *19*, 43–57. [CrossRef]
24. Li, H.; Liang, Y.; Tang, Q. Parallel algorithm for collision avoidance motion planning of dual arm grasping of humanoid robot. *Comp. Model. New Technol.* **2014**, *18*, 110–116.
25. García, N.; Rosell, J.; Suárez, R. Motion planning by demonstration with human-likeness evaluation for dual-arm robots. *IEEE Trans. Syst. Man. Cy.* **2019**, *49*, 2298–2307. [CrossRef]
26. Xian, Z.; Lertkultanon, P.; Pham, Q.C. Closed-chain manipulation of large objects by multi-arm robotic systems. *IEEE Robot. Autom. Let.* **2017**, *2*, 1832–1839. [CrossRef]

27. Jang, K.; Baek, J.; Park, S.; Park, J. Motion planning for closed-chain constraints based on probabilistic roadmap with improved connectivity. *IEEE/ASME Trans. Mech.* **2022**, *27*, 2035–2043. [CrossRef]
28. Wang, X.; Chen, L. A vision-based coordinated motion scheme for dual-arm robots. *J. Intell. Robot. Syst.* **2020**, *97*, 67–79. [CrossRef]
29. Gao, M.; Zhou, H.; Yang, Y.; Dong, Z.; He, Z. An intelligent master–slave collaborative robot system for cafeteria service. *Robot. Auton. Syst.* **2022**, *154*, 104121. [CrossRef]
30. Tchoń, K.; Janiak, M. Repeatable approximation of the jacobian pseudo-inverse. *Syst. Control Lett.* **2009**, *58*, 849–856. [CrossRef]
31. Reiter, A.; Müller, A.; Gattringer, H. On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators. *IEEE Trans. Ind. Inform.* **2018**, *14*, 1681–1690. [CrossRef]
32. Zhang, F.; Hua, L.; Fu, Y.; Guo, B. Dynamic simulation and analysis for bolt and nut mating of dual arm robot. In Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics, Guangzhou, China, 11–14 December 2012; pp. 660–665.
33. Jiao, C.; Yu, L.; Su, X.; Wen, Y.; Dai, X. Adaptive hybrid impedance control for dual-arm cooperative manipulation with object uncertainties. *Automatica* **2022**, *140*, 110232. [CrossRef]
34. Ren, X.; Zhang, P.; Zhang, Z. Bicriteria velocity minimization approach of self-motion for redundant robot manipulators with varying-gain recurrent neural network. *IEEE Trans. Cogn. Dev. Syst.* **2022**, *14*, 578–587. [CrossRef]
35. Wu, T.; Zhao, J.; Xie, B. A novel method for computing self-motion manifolds. *Mech. Mach. Theory* **2023**, *179*, 105121. [CrossRef]
36. García, N.; Suárez, R.; Rosell, J. Task-dependent synergies for motion planning of an anthropomorphic dual-arm system. *IEEE Trans. Robot.* **2017**, *33*, 756–764. [CrossRef]
37. Wang, Z.; Gan, Y.; Dai, X. Assembly-oriented task sequence planning for a dual-arm robot. *IEEE Robot. Autom. Let.* **2022**, *7*, 8455–8462. [CrossRef]
38. Zimmermann, S.; Hakimifard, G.; Zamora, M.; Poranne, R.; Coros, S. A multi-level optimization framework for simultaneous grasping and motion planning. *IEEE Robot. Autom. Let.* **2020**, *5*, 2966–2972. [CrossRef]
39. Li, Y.; Hao, X.; She, Y.; Li, S.; Yu, M. Constrained motion planning of free-float dual-arm space manipulator via deep reinforcement learning. *Aerosp. Sci. Technol.* **2021**, *109*, 106446. [CrossRef]
40. Lauretti, C.; Grasso, T.; Marchi, E.d.; Grazioso, S.; Gironimo, G.d. A Geometric Approach to Inverse Kinematics of Hyper-Redundant Manipulators for tokamaks maintenance. *Mech. Mach. Theory* **2022**, *176*, 104967. [CrossRef]
41. Gammell, J.D.; Barfoot, T.D.; Srinivasa, S.S. Informed sampling for asymptotically optimal path planning. *IEEE Trans. Robot.* **2018**, *34*, 966–984. [CrossRef]
42. Yuan, C.; Liu, G.; Zhang, W.; Pa, X. An efficient RRT cache method in dynamic environments for path planning. *Robot. Auton. Syst.* **2020**, *131*, 103595. [CrossRef]
43. Xu, Z.; Deng, D.; Shimada, K. Autonomous UAV exploration of dynamic environments via incremental sampling and probabilistic roadmap. *IEEE Robot. Autom. Let.* **2021**, *6*, 2729–2736. [CrossRef]
44. Cao, Y.; Cheng, X.; Mu, J. Concentrated coverage path planning algorithm of UAV formation for aerial photography. *IEEE Sens. J.* **2022**, *22*, 11098–11111. [CrossRef]
45. Tian, Y.; Zhu, X.; Meng, D.; Wang, X.; Liang, B. An overall configuration planning method of continuum hyper-redundant manipulators based on improved artificial potential field method. *IEEE Robot. Autom. Let.* **2021**, *6*, 4867–4874. [CrossRef]
46. Pan, Z.; Zhang, C.; Xia, Y.; Xiong, H.; Shao, X. An improved artificial potential field method for path planning and formation control of the multi-UAV systems. *IEEE Trans. Circuits Syst. II Exp. Briefs* **2022**, *69*, 1129–1133. [CrossRef]
47. Peidró, A.; Reinoso, Ó.; Gil, A.; Marín, J.M.; Payá, L. A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots. *Mech. Mach. Theory* **2018**, *128*, 84–109. [CrossRef]
48. Gong, M.; Li, X.; Zhang, L. Analytical inverse kinematics and self-motion application for 7-DOF redundant manipulator. *IEEE Access* **2019**, *7*, 18662–18674. [CrossRef]