agriculture

Article

Motion Planning of the Citrus-Picking Manipulator Based on the TO-RRT Algorithm

Cheng Liu [1], Qingchun Feng [2,*], Zuoliang Tang [1], Xiangyu Wang [3], Jinping Geng [1] and Lijia Xu [1,*]

1 College of Mechanical and Electrical Engineering, Sichuan Agricultural University, Ya'an 625014, China; 2020317017@stu.sicau.edu.cn (C.L.); zuoliang_tang@stu.sicau.edu.cn (Z.T.); 2020217010@stu.sicau.edu.cn (J.G.)
2 Intelligent Equipment Research Center, Beijing Academy of Agriculture and Forestry Sciences, Beijing 100097, China
3 Institute of System Science and Technology, School of Electrical Engineering, Southwest Jiaotong University, Chengdu 611756, China; wxy1998@my.swjtu.edu.cn
* Correspondence: fengqc@nercita.org.cn (Q.F.); xulijia@sicau.edu.cn (L.X.)

Abstract: The working environment of a picking robot is complex, and the motion-planning algorithm of the picking manipulator will directly affect the obstacle avoidance effect and picking efficiency of the manipulator. In this study, a time-optimal rapidly-exploring random tree (TO-RRT) algorithm is proposed. First, this algorithm controls the target offset probability of the random tree through the potential field and introduces a node-first search strategy to make the random tree quickly escape from the repulsive potential field. Second, an attractive step size and a "step-size dichotomy" are proposed to improve the directional search ability of the random tree outside the repulsive potential field and solve the problem of an excessively large step size in extreme cases. Finally, a regression superposition algorithm is used to enhance the ability of the random tree to explore unknown space in the repulsive potential field. In this paper, independent experiments were carried out in MATLAB, MoveIt!, and real environments. The path-planning speed was increased by 99.73%, the path length was decreased by 17.88%, and the number of collision detections was reduced by 99.08%. The TO-RRT algorithm can be used to provide key technical support for the subsequent design of picking robots.

Keywords: picking manipulator; motion planning; TO-RRT; step-size dichotomy; regression superposition

check for updates

Citation: Liu, C.; Feng, Q.; Tang, Z.; Wang, X.; Geng, J.; Xu, L. Motion Planning of the Citrus-Picking Manipulator Based on the TO-RRT Algorithm. *Agriculture* **2022**, *12*, 581. https://doi.org/10.3390/agriculture12050581

Academic Editor: Wen-Hao Su

Received: 16 March 2022
Accepted: 19 April 2022
Published: 21 April 2022

1. Introduction

Citrus is one of the most economically important crops in the world, and it is also the most cultivated fruit in southwestern China. Currently, citrus fruits are mainly picked manually, which is time-consuming, laborious, and expensive. According to a survey, the labor used in citrus picking operations accounts for $33 \sim 50\%$ of the whole production process. With the sharp decline in the number of rural employees in China, the development of the citrus industry has been severely restricted. To improve the efficiency of picking and enhance the competitiveness of China's citrus industry, both the research and development of citrus-picking robots have become research hotspots at home and abroad, and the path planning of the picking manipulator is one of the most difficult technologies.

In recent years, a series of path-planning methods have been proposed. The artificial potential field (APF) can be used to prevent the manipulator from colliding with obstacles when approaching the target. However, the APF easily falls into a local minimum, and it easily falls into oscillation in a complex environment [1]. Compared to the APF, the rapidly-exploring random tree (RRT) is more adaptable, faster, and more variable, but it is difficult to find the best path when using this approach [2]. Bidirectional RRT and RRT-connect algorithms are used to generate two random trees at the initial node and the target node, respectively, which improves the search speed compared with the RRT algorithm, but the path is still not optimal [3,4]. The RRT-star (RRT*) algorithm is used to make the path gradually converge with the optimum in the search process by reselecting the parent node and rerouting, but its running time is longer than that of the RRT algorithm [5]. Mohammed et al. [6] defined a straight line connecting the initial node and the target node so that the generation probability of the random tree node was normally distributed with the distance from the straight line, preventing excessive searching and avoiding falling into local extreme values. However, the searching ability in a complex environment still needs to be improved. Akgun et al. [7] combined the bidirectional RRT and RRT* algorithms to optimize the search time. Jeong et al. [8] proposed an RRT*-Quick method, which caused the nodes to tend to share the same parent node in a circular (or spherical) neighborhood. While the path generated using this method was smoother than the path generated by the RRT* algorithm, the search time increased slightly. Jeong et al. [9] introduced an informed-RRT algorithm into the RRT*-quick method to limit the sampling space of the random tree and solve the problem of increased search time caused by expanding the search domain in the process of improving the quality of the solution. When the tree nodes reached the maximum, the RRT* Fixed Nodes (RRT*FN) algorithm was used to remove a weak node and add a high-performance node so that the generated tree node was much smaller than the one in the RRT* algorithm. However, this method had little performance gap with the RRT* algorithm before the tree nodes reached the maximum number of nodes [10].

The RRT* algorithm has a strong ability to optimize the path cost, but its search efficiency is low. Cao et al. [11] introduced the target gravity to the RRT algorithm, and the attraction generated by the random node and the attraction generated by the target node were used to jointly guide the generation of new nodes in the random tree. This method improved the search speed of the random tree, but it could not escape the obstacle area quickly when blocked by obstacles. Wang et al. [12] changed the sampling area and assigned node state values so that the random tree could only be expanded through boundary nodes to reduce the generation of invalid nodes, but many redundant nodes were generated near obstacles. Zhang et al. [13] screened new nodes based on a biased-RRT algorithm. If the distance between the new node and the parent node was greater than the distance between the new node and any other nearby node, the new node was discarded. This method can be used to prevent excessive searching of the space and reduce the total number of nodes. Gong et al. [14] made the search direction of the random tree always point to the target node and performed local path planning near the obstacles. Although this method could reduce excessive searching of the space, its escape speed was slow when the random tree was blocked by many large obstacles. Li et al. [15] put forward an adaptive RRT-connect (ARRT-connect), which allowed the random tree to still have good performance in a narrow environment, and path planning could still be completed in a short time. Gao et al. [16] proposed a planning method based on an independent potential field that made the manipulator explore the gradient direction when it was far away from the target and avoided obstacles through the random search. Wang et al. [17] selected tree nodes according to the geometric structure and position of obstacles so that a path with a lower cost could be quickly obtained, but the effect of avoiding obstacles with irregular shapes was poor.

In this paper, based on a citrus tree environment, taking the shortest time as the optimization goal, and taking the Franka manipulator as the experimental platform, the RRT algorithm is improved in multiple dimensions. Its main contributions are as follows:

1. On the basis of the biased-RRT, the potential field function and the adaptive probability threshold are introduced, so that the random tree has corresponding growth strategies in different potential fields. The above strategies improve the directional search ability of random trees in the repulsive potential field and enhance the escape ability of random trees in the repulsive potential field;
2. To solve the problem of "falling into a trap" in the repulsive potential field of random trees, a node-first search strategy is proposed, which makes the selection of extended nodes of random trees more purposeful;

3. Proper step size is crucial to improve search ability. Using an attractive step size is helpful to reduce the number of collision detections and computational complexities outside the repulsive potential field. "Step-size dichotomy" solves the problem of random trees colliding with obstacles many times due to too large of step size in the repulsive potential field;
4. By introducing a regression superposition algorithm, the random tree can avoid over-searching space in the repulsive potential field and enhance the escape ability.

The rest of this article is organized as follows: The basic principles of the RRT algorithm, as well as some improvement methods and the design process of the TO-RRT algorithm, including the adaptive probability threshold, the node-first search strategy, an attractive step size, "step-size dichotomy", and a regression superposition algorithm are introduced in Section 2. In Section 3, the performance of various algorithms in MATLAB, MoveIt!, and the real environment are compared. The main contributions of the article and future work are discussed in Section 4. The full text is summarized in Section 5.

2. Materials and Methods

2.1. RRT Algorithm

The RRT algorithm, which is a spatial search algorithm based on random sampling, aims to generate a collision-free random tree connecting the first and the last positions [18,19].

Each time the *Tree* grows, a random node q_{rand} is generated in the space. Then, the tree node q_{near} closest to q_{rand} is found in the tree, and a new tree node q_{new} is found in the direction of $q_{near} \to q_{rand}$ with a fixed step λ and is connected to q_{near} as $\overline{q_{near}q_{new}}$. If neither q_{new} nor $\overline{q_{near}q_{new}}$ collide with obstacles, q_{new} and $\overline{q_{near}q_{new}}$ are added to the random tree. After several expansions, if the distance between q_{new} and q_{goal} is less than the given threshold, the *Tree* finds a path connecting q_{init} to q_{goal}, as shown in Algorithm 1.

Algorithm 1. RRT Algorithm.

```
1: $Tree \leftarrow q_{init}$
2: for $i = 1$ to $n$ do
3: $q_{rand} \leftarrow RandomSample(C_{free});$
4: $q_{near} \leftarrow NearestPoint(Tree, q_{rand});$
5: $q_{new} \leftarrow Extend(q_{near}, q_{rand}, \lambda);$
6: if $CollisionFree(q_{near}, q_{new})$ then
7 :  $AddNewPoint(Tree, q_{new});$
8: end if
9 :  if $Distance(q_{new}, q_{goal}) < \rho_{min}$ then
10: return $Tree$
11: end if
12: end for
```

2.2. Some Improvement Methods

The RRT algorithm can be used to effectively explore high-dimensional space, but the path cost is high, and the algorithm takes a long time to reach completion. The biased-RRT algorithm can be used to effectively solve the shortcomings of the RRT algorithm [20–23], as shown in Algorithm 2.

Algorithm 2. Biased-RRT Algorithm.

```
1: $Tree \leftarrow q_{init}$
2: for $i = 1$ to $n$ do
3: if $RandomNumber <= m$ then
4: $q_{rand} \leftarrow RandomSample(C_{free});$
5: else
6: $q_{rand} \leftarrow q_{goal};$
7: end condition
8: $q_{near} \leftarrow NearestPoint(Tree, q_{rand});$
9: $q_{new} \leftarrow Extend(q_{near}, q_{rand}, \lambda);$
10: if $CollisionFree(q_{near}, q_{new})$ then
11: $AddNewPoint(Tree, q_{new});$
12: end if
13: if $Distance(q_{new}, q_{goal}) < \rho_{min}$ then
14: return $Tree$
15: end if
16: end for
```

In Algorithm 2, q_{rand} is determined by the size between the random number, $RandomNumber$, and the probability threshold m. If $RandomNumber <= m$, q_{rand} takes any point in the space; otherwise, the target node is taken as the sampling point.

The biased-RRT algorithm is used to guide the growth of the random tree, increase the effectiveness of sampling points, and shorten the time of path planning. However, when obstacles obstruct the growth of random trees, the biased-RRT algorithm cannot escape the obstacles quickly. Therefore, some scholars have put forward corresponding solutions, as shown in Table 1.

Table 1. Comparison of RRT improvement methods.

RRT Type	Algorithm Name	Solutions
Biased-RRT	NC-RRT [12]	The random tree search is guided by gradually changing the sampling area, and it is expanded through the boundary nodes as much as possible through the node control mechanism.
Biased-RRT	RRT-BCR [13]	A regression mechanism is introduced to prevent excessive searching, and an adaptive expansion mechanism is introduced to avoid the repeated search of expansion nodes.
RRT*	MOD-RRT* [24]	An initial path planner and a path replanner are proposed. When encountering obstacles, the path replanner selects alternative paths to avoid collision.
P-RRT	PBG-RRT [25]	By giving weights to the goal and random points, the random tree deviates from obstacles.
RRT*	HSRRT* [26]	The random tree is guided to deviate from an obstacle through the APF, and the heuristic sampling scheme of Gaussian function is used to generate sampling points near the obstacle to improve the search efficiency.

Note: NC-RRT, Node Control-RRT; RRT-BCR, Biased-RRT algorithm with boundary expansion mechanism and regression mechanism; MOD-RRT*, multi-objective RRT*; PBG-RRT, rapidly exploring random tree based on heuristic probability bias-goal; HSRRT*, heuristically sampling-based rapidly exploring random tree.

2.3. TO-RRT Algorithm

2.3.1. Adaptive Probability Threshold

At present, some improved RRT algorithms have been used to add potential field functions to the target node q_{goal}, random nodes q_{rand}, and obstacles. The random tree changes its growth direction under the action of a combined potential field, which makes it expand to the target when avoiding obstacles [27–33]. This kind of algorithm improves the search efficiency of the random tree, but each expansion of the random tree requires several vector operations of the potential field force, which occupies a large amount of the system memory. In addition, if the repulsive potential field of the obstacles is considered, the algorithm may fall into a local minimum problem, as is common in APF, resulting in q_{near} being unable to generate a new node q_{new}, as shown in Figure 1.

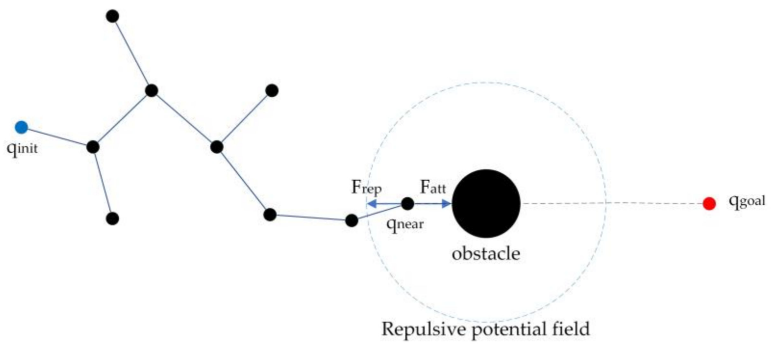


Figure 1. The algorithm falls into a local minimum.

Considering the complexity and uncertainty of the above algorithm in different environments, an attractive potential field and a repulsive potential field for the target node and obstacle, respectively, based on the biased-RRT algorithm were established. Therefore, the probability threshold changed according to the different types of potential fields.

Similar to the APF algorithm, the range of the attractive potential field was the whole operating space, while the range of the repulsive potential field was limited to a certain distance around the obstacle. In the range of the repulsive potential field, if the random tree tended to search for the target node q_{goal}, the random tree had a strong ability to grow biased. At this time, if the obstacle blocked q_{goal}, multiple failed growth near obstacles could occur for the random tree, so it tended to search randomly within the repulsive potential field. When the random tree left the range of the repulsive potential field, it continued to tend to search for q_{goal}, as shown in Algorithm 3.

Algorithm 3. Probability Threshold under the Control of Potential Field.

```
1: if $RandomNumber <= threshold(q_{new}, obstacle)$ then
2: $q_{rand} \leftarrow RandomSample(C_{free})$;
3: else
4: $q_{rand} \leftarrow q_{goal}$;
5: end if
6: return $q_{rand}$
```

The growths of the random tree under both the control of the constant probability threshold and the adaptive probability threshold are shown in Figure 2a,b, respectively. Figure 2a shows that, if the random tree maintained a constant probability threshold during the search process, the obstacle did not affect the goal of random tree expansion. If the adaptive probability threshold was adopted, the random tree chose a better growth direction according to the location tendency of the new node. It was learned through many experiments that the probability threshold outside the scope of the repulsive potential field was 0.3, and the probability threshold inside the range of the repulsive field was 0.7.

2.3.2. Node-First Search Strategy

According to the biased-RRT algorithm, when $RandomNumber > m$, q_{rand} takes the coordinate value of q_{goal} and then selects the q_{near} closest to q_{rand} in the random tree as the parent node of q_{new}. If the random tree only expands to the target in each search round without considering the random search, then the new node in this search round will become the parent node of the new node in the next search round, and the random tree is a straight line segment connecting q_{init} and q_{goal}.

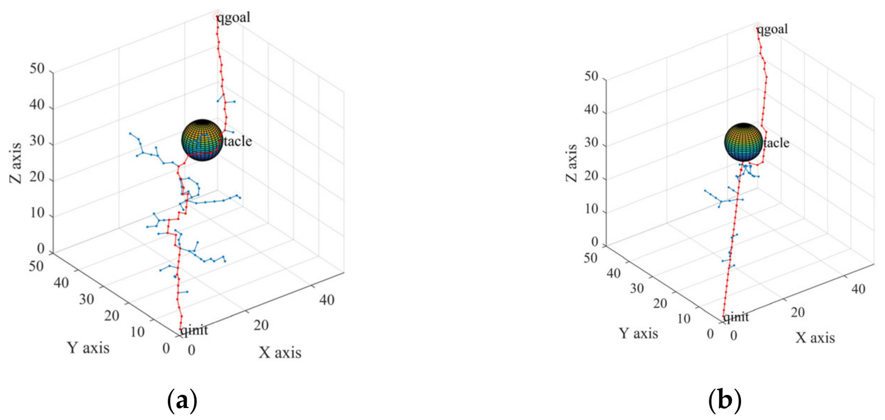


Figure 2. Threshold comparison. (**a**) Constant probability threshold; (**b**) Adaptive probability threshold.

As in Section 2.3.1, when the end node of the random tree expands to the range of the repulsive potential field, the random tree tends to select any node in the space as q_{rand} in the next search selection, so the probability of random expansion of the end node is small. If the next round of search satisfies $RandomNumber > threshold$, since the end node of the random tree is closest to q_{goal}, q_{goal} will be expanded, causing the newly generated path to collide with the obstacle. To summarize, when $RandomNumber > threshold$, the end node collides with the obstacle; when $RandomNumber <= threshold$, any node in the tree will be selected for expansion, which is no different from the traditional RRT algorithm. This phenomenon is called "falling into a trap", as shown in Figure 3.

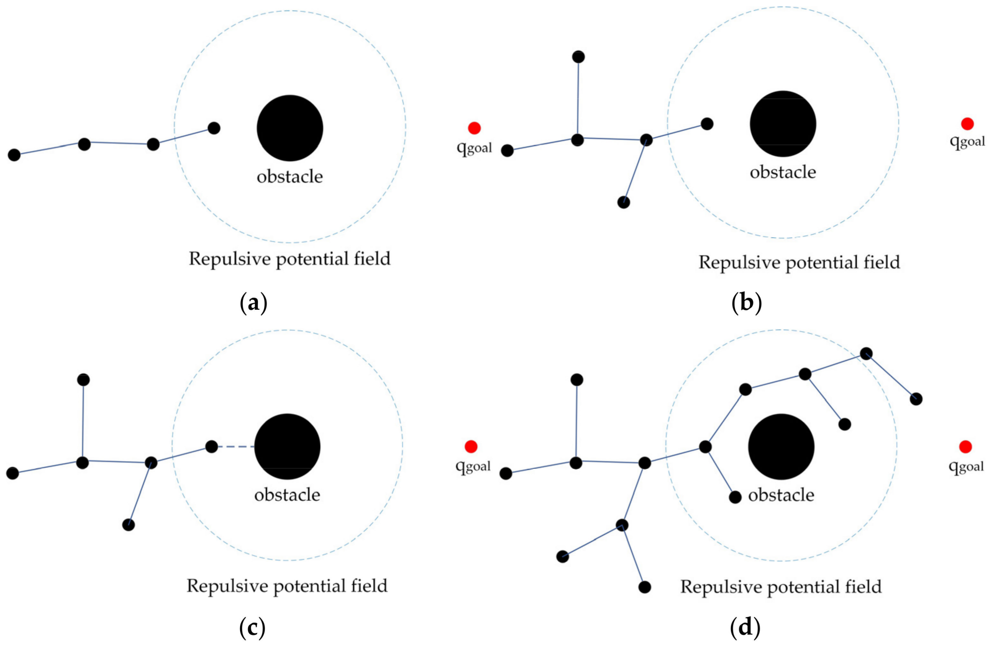


Figure 3. The random tree falls into a trap in the repulsive potential field. (**a**) The random tree entering obstacle potential field; (**b**) The random tree begins to expand randomly; (**c**) The random trees collide with obstacles; (**d**) The random tree is expanded several times.

For this reason, a node-first search strategy was proposed in this paper, as shown in Algorithm 4. When the q_{new} of the random tree grew into the range of the repulsive potential field of obstacles, a virtual spherical surface with a radius r and center q_{new} was generated. If $RandomNumber <= threshold$ was satisfied in the next round of search, point q_{rand} on the virtual spherical surface was preferentially selected, and q_{new} was used as the parent node of the next round of search to generate a new node q_{new2}. If q_{new2} and the line segment $\overline{q_{new}q_{new2}}$ did not collide with obstacles, the path and q_{new2} were kept. A new search round continued until the end node of the random tree was separated from the obstacles, as shown in Figure 4.

Algorithm 4. Node-First Search Algorithm.

```
1: if $RandomNumber <= threshold(q_{new}, obstacle)$ then
2: if $Distance(q_{new}, obstacle) < R_{rpf}$ then
3: $q_{rand} \leftarrow sphere(q_{new}, r_{virtual});$
4: else
5: $q_{rand} \leftarrow RandomSample(C_{free});$
6: end if
7: else
8: $q_{rand} \leftarrow q_{goal};$
9: end if
10: return $q_{rand}$
```

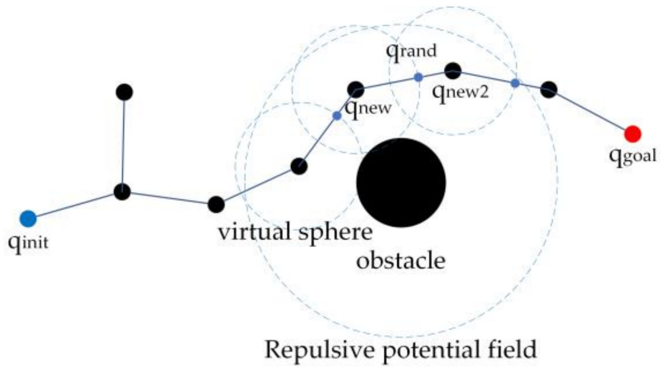


Figure 4. Schematic diagram of the node-first search strategy.

2.3.3. Attractive Step Size and Step-Size Dichotomy

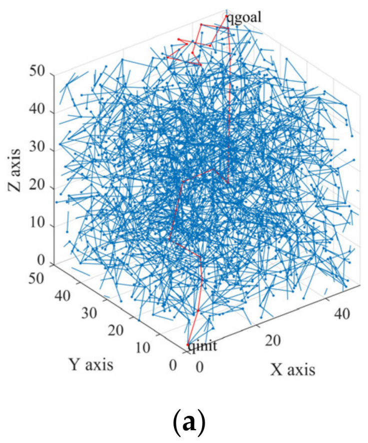
From the above description, the node-first search strategy was used to prevent the random tree "falling into a trap" within the range of the repulsive potential field. Since the random tree has a certain probability of random search outside the range of the repulsive potential field of obstacles, more iterations will be generated. An appropriate step size can effectively reduce the iterations of the random tree. In the case that the length of the path is determined, a small step size will cause more collision detections and distance calculations, and a large step size will often make the random tree collide with obstacles. Therefore, the step size should be expanded as much as possible on the premise of reducing the number of collisions [34,35].

According to the APF algorithm, the attractive force of q_{goal} acts on the whole operating space and is proportional to the distance between the end joints of the manipulator, which is beneficial to control the growth step of the RRT. If obstacles are not considered, the random tree should increase the step size when it is far away from q_{goal} to quickly expand to q_{goal}. When the random tree is closer to q_{goal}, if it continues to maintain a large step size, a large number of redundant nodes will be generated at q_{goal}, as shown in Figure 5a. Therefore, the random tree should gradually approach q_{goal} with small step sizes, as shown in Figure 5b.

For this reason, an attractive step size was proposed, which was defined as:

$$attStepsize = k \times Distance(q_{near}, q_{goal}) \tag{1}$$

where $attStepsize$ represents the attractive step size, and k is the attractive parameter.



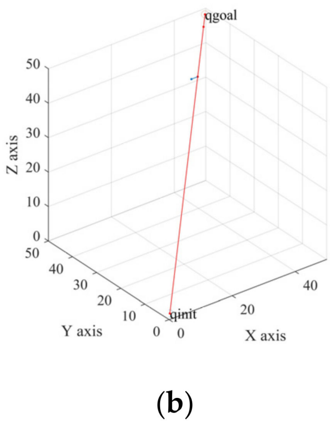


Figure 5. Fixed step size and attractive step size. (**a**) Fixed step size; (**b**) Attractive step size.

If obstacles are considered, the step size of the random tree in the random search is fixed, and an attractive step size is taken when growing toward q_{goal}. This method ensures that the random tree grows toward q_{goal} as soon as possible outside the range of the repulsive potential field and avoids collisions with obstacles due to excessive steps within the range of the repulsive potential field.

The parameters of the potential field function of the manipulator are different in different operating spaces. For example, when the attractive parameter k is too large, $attStepsize$ will increase accordingly. If $Distance(q_{near}, obstacle) < attStepsize$, q_{new} will collide with obstacles. In addition, the end nodes tend to grow toward q_{goal} outside the range of the repulsive potential field. Therefore, the random tree still has a high probability of colliding with obstacles in the next round of search.

For this reason, a "step-size dichotomy" was introduced to solve the problem of excessive step size. When q_{near} grew toward q_{goal} and there were obstacles between them, the distance d_{nob} between q_{near} and the obstacles was calculated. If $d_{nob} <= attStepsize$, the $attStepsize$ was shortened to the original value of 2^{-1}, and the sizes of $attStepsize$ and d_{nob} were compared again until $d_{nob} > attStepsize$; see Algorithm 5.

Algorithm 5. Step-size Dichotomy.

```
1: if $Collision(q_{near}, q_{goal})$ then
2: while $adpStepsize > Distance(q_{near}, obstacle)$ do
3: $adpStepsize = adpStepsize/2$;
4: end while
5: else
6: $attStepsize = k \times Distance(q_{near}, q_{goal})$;
7: end if
8: return $attStepsize$
```

2.3.4. Regression Superposition Algorithm

From Section 2.3.3, if the random tree grows within the range of the obstacle repulsive potential field, a large number of redundant nodes will be generated on the surface of the obstacle due to the high probability of the random search, as shown in Figure 6a. As a result, a regression superposition algorithm is proposed in this section to adaptively select extended nodes and change the step size of the random search, as shown in Figure 6b.

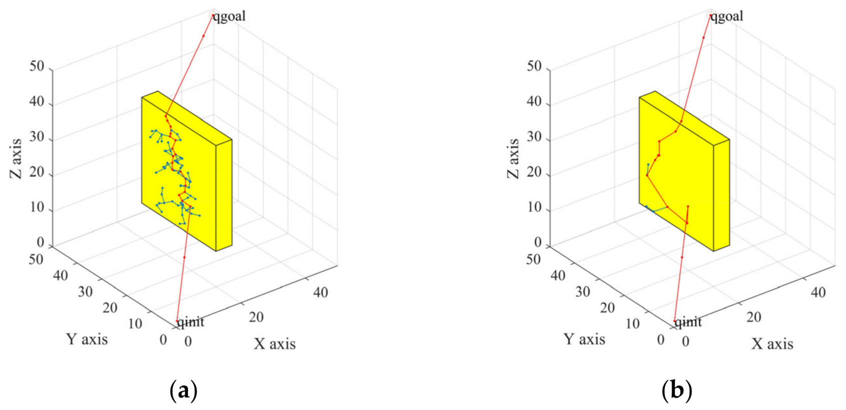


Figure 6. Differences before and after improvement. (**a**) The random trees generated a large number of nodes on the surface of obstacles; (**b**) The random tree had fewer nodes on the obstacle surface.

The regression superposition algorithm consists of a regression algorithm [36] and a step-size superposition algorithm. In the regression algorithm, if the distance between q_{new} and q_{near} was larger than the distance between q_{new} and any node q_i in the random tree except q_{near}, it was considered to meet the regression conditions:

$$\begin{cases} Distance(q_{near}, q_{new}) > Distance(q_{near}, q_i) \\ q_i \in Tree \end{cases} \tag{2}$$

If Formula (2) was satisfied, q_{new} was regarded as a regression node. The regression node would not become the tree node of the random tree, but it was removed until a new node that did not meet the regression condition was found, as shown in Figure 7a.

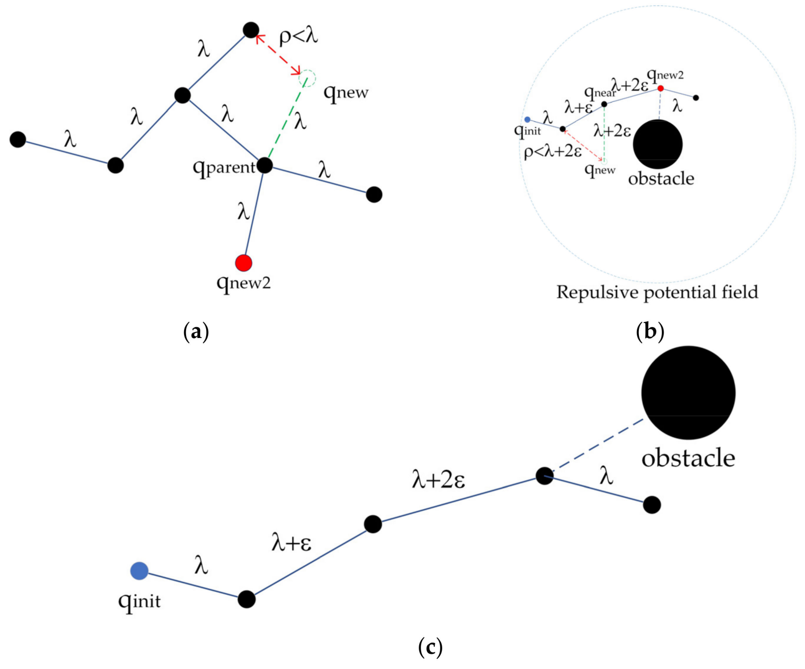


Figure 7. Regression superposition algorithm. (**a**) Regression algorithm; (**b**) Regression superposition algorithm; (**c**) Step-size superposition algorithm. Note: λ, the initial step size; ε, the step size of superposition; ρ, the distance between q_{new} and any node.

To further reduce the number of tree nodes, the step-size superposition algorithm was incorporated based on a regression algorithm. When the random tree was searched randomly, the initial step size was set to λ, and the step size was increased by ε after each round of the random search until the extended branches of the random tree collided with obstacles. Then, the search step size of the next round was returned to the initial step size λ, and the step size was superimposed again until the random tree searched toward q_{goal}, as shown in Figure 7c.

The random tree used an attractive step size when searching toward q_{goal} to reduce the generation of redundant nodes. During the random search of the random tree, the regression superposition algorithm was used to enhance the ability of the random tree to search the unknown space, as shown in Figure 7b.

The TO-RRT algorithm was used to dynamically adjust the growth direction of the random tree by the probability threshold controlled by the potential field and to define two different growth methods according to the different growth directions. Therefore, the random tree could quickly grow to the target outside the range of the repulsive potential field and quickly determine the escape path within the range of the repulsive potential field. The algorithm flow chart is shown in Figure 8.

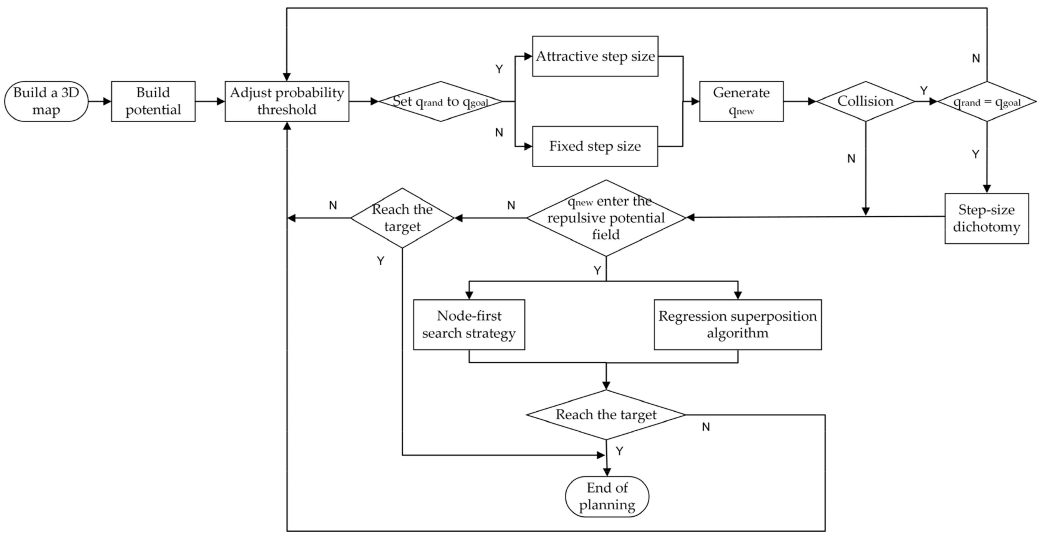


Figure 8. Flowchart of the TO-RRT algorithm. Note: A, yes; N, no.

3. Results

3.1. Comparative Experiment of Path Planning in a Complex Environment

To verify the speed, stability, and low path cost of the TO-RRT algorithm, the RRT algorithm, the biased-RRT algorithm with a target offset probability of 50%, the TO-RRT algorithm, the RRT-BCR algorithm, and the NC-RRT algorithm are compared in this section using complex environments (i.e., a multi-sphere environment, a multi-rectangle environment, a single-channel environment, and a multi-channel environment).

In the simulation experiment, the initial step size was 2, the maximum number of failed growth times was 100,000, the map size was $50 \times 50 \times 50$, the starting point was (1, 1, 1), and the target point was (49, 49, 49). The blank area in the map represented the obstacle-free area, other colors represented the obstacle area, the blue path represented the random tree, the black path represented the collision-free path from the starting point to the target point, and the red path represented the path optimized by the greedy algorithm.

Figure 9a,e,i,m,q show that, although the RRT algorithm can be used to find a collision-free path from the initial point to the target point, the whole space was searched, so that the highest amount path nodes were generated. Compared with the RRT algorithm, the biased-RRT algorithm did not search too much invalid space, so there were fewer path nodes. When using the RRT-BCR algorithm and the NC-RRT algorithm, the sizes of the random trees were reduced through a regression mechanism and an adaptive sampling area, respectively. The TO-RRT algorithm was used to greatly reduce the number of nodes in the space, and its complexity was the lowest. Figure 9b,f,j,n,r show that the RRT algorithm still searched the whole space. Although the biased-RRT algorithm generated fewer nodes than the RRT algorithm, the search tree generated a large number of nodes on the surface of obstacles, which increased the number of iterations. The NC-RRT algorithm made the random tree tend to expand through boundary nodes through the node control mechanism, so it had fewer redundant nodes. It can be seen from Figure 9c,d,g,h,k,l,o,p,s,t that the RRT algorithm and the biased-RRT algorithm could not quickly find the "escape channel". Although the RRT-BCR algorithm limited the expansion of nodes that were prone to collision, it increased the expansion times of other nodes. Due to the regression superposition algorithm and node-first search strategy introduced into the TO-RRT algorithm, the random tree could quickly search the nearby area to find the "escape channel" in the repulsive potential field.

There are certain errors and contingencies in a single experiment. To better reflect a real situation, 10 simulation experiments were carried out in the same environment as described above, shown in Figure 10.

Figure 10 shows that the TO-RRT algorithm maintained strong stability in 10 experiments and did not traverse the whole space due to being blocked by obstacles, while the RRT algorithm and the biased-RRT algorithm both generated a large number of nodes in the space. In addition, the RRT-BCR algorithm had fewer path nodes than the biased-RRT algorithm, and in the NC-RRT algorithm, there was little difference in the path in each search. The comparison of the running times of the three algorithms in different environments is shown in Figure 11. Figure 11 shows that the RRT algorithm had the longest running time and poor running-time stability, especially in a single-channel environment, with the longest running time at 45.6057 s and the shortest running time at 1.2880 s. Compared with the RRT algorithm, the biased-RRT algorithm had a much shorter running time and strong running-time stability, but the search time in a complex environment was longer. The longest running times of the TO-RRT algorithm in the four environments were 0.0225 s, 0.0420 s, 0.0618 s, and 0.0443 s, and the shortest running times were 0.0056 s, 0.0134 s, 0.0101 s, and 0.0115 s. The difference between the longest search time and the shortest search time in a single environment did not exceed 0.06 s, which not only indicated a short search time but also a strong and stable running time. The NC-RRT algorithm performed poorly in a multi-rectangle environment, with a difference of 4.44 times between the longest running time and the shortest running time, while the RRT-BCR algorithm was only 3.82 times.

Table 2 shows the average values of each index of the 3 algorithms over 10 experiments (biased-RRT represents the biased-RRT algorithm with a target offset probability of 50%). In the multi-sphere environment, the TO-RRT algorithm had a running time that was 99.74% less than the RRT algorithm, which was mainly because the number of collision detections and the number of failed node growths of the former were reduced by 99.39% and 97.17%, respectively, compared with the latter. In addition, compared with the RRT algorithm, the number of path nodes in the TO-RRT algorithm was reduced by 82.92%, which shortened the length of its search path by 18.99%. When the random tree encountered a large area of obstacles, the TO-RRT algorithm was used to reflect the advantages in the search time more than the RRT algorithm. For example, the number of tree nodes and the number of failed growths of nodes of the RRT algorithm in the multi-rectangle environment reached 17,358.3 and 3144.8, respectively, resulting in a running time of 7.8822 s, while the running time of the TO-RRT algorithm was only 0.0213 s. In addition, the RRT-BCR algorithm performed better than the NC-RRT algorithm in a multi-rectangle environment, and its running time was shortened by 29.14% compared with the NC-RRT algorithm because the RRT-BCR algorithm removed nodes that collided many times when facing obstacles with large occlusion areas. The biased-RRT algorithm produced too much failure growth when encountering obstacles with large areas. For example, in a multi-channel environment, the node failure growth rate of the biased-RRT algorithm was 62.54%, while the RRT algorithm and TO-RRT algorithm had node failure growth rates of only 36.40% and 15.82%, respectively. Therefore, the biased-RRT algorithm was not ideal in a complex environment. Since the NC-RRT algorithm always took the area between the configuration point and the target as the sampling radius and tended to use boundary nodes for expansion, it could not produce valid nodes when the obstacle was between the configuration point and the target. For example, in multi-channel and multi-rectangle environments, the collision detection times of the NC-RRT algorithm were 21,487 times and 55,077 times. In summary, compared with the other algorithms, the TO-RRT algorithm had significant advantages in searching speed and the number of nodes in the random tree.

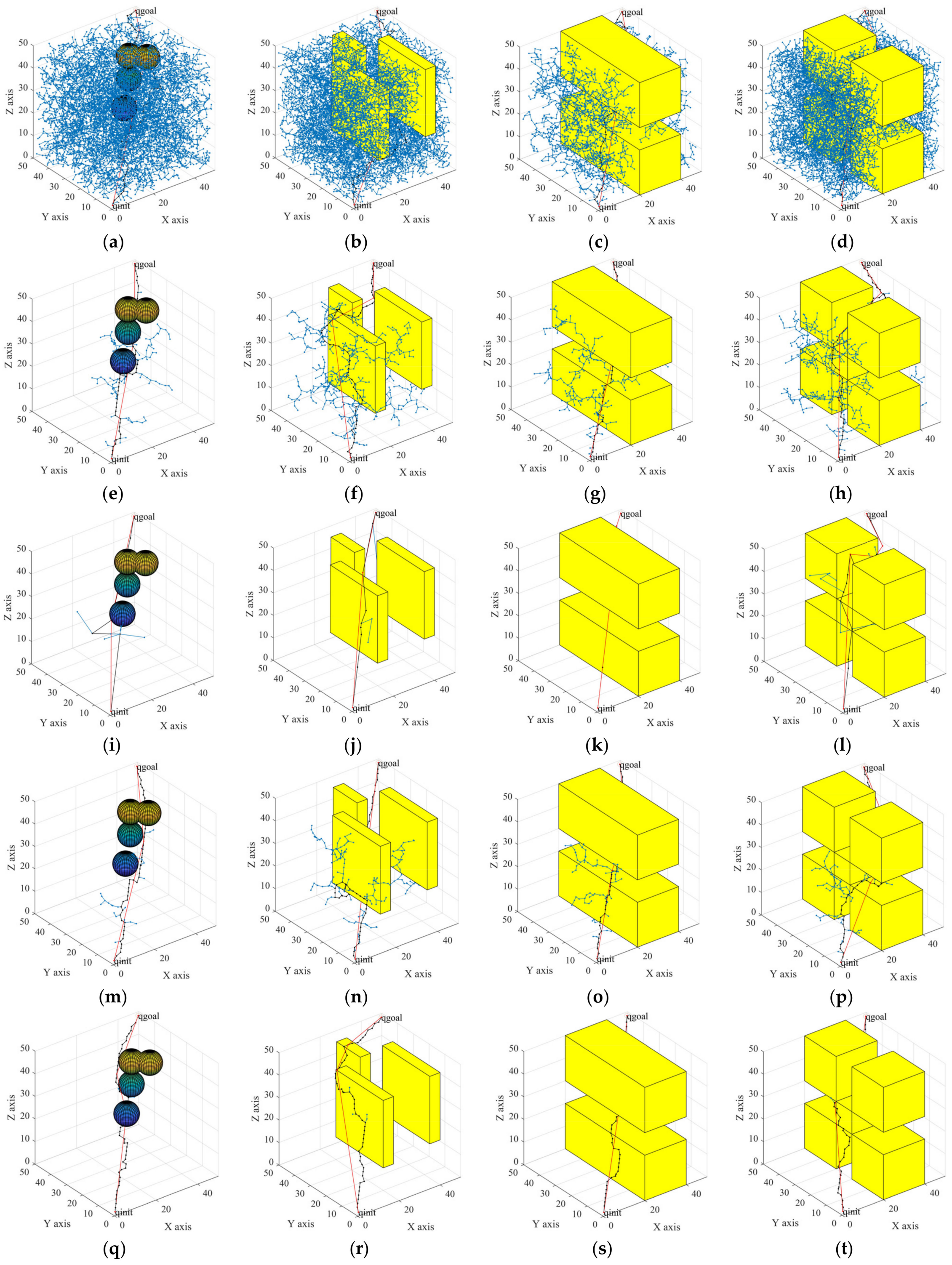


Figure 9. The performances in different environments of: the RRT algorithm (**a**–**d**); the biased-RRT algorithm with a target offset probability of 50% (**e**–**h**); the TO-RRT algorithm (**i**–**l**); the RRT-BCR algorithm (**m**–**p**); and the NC-RRT algorithm (**q**–**t**).

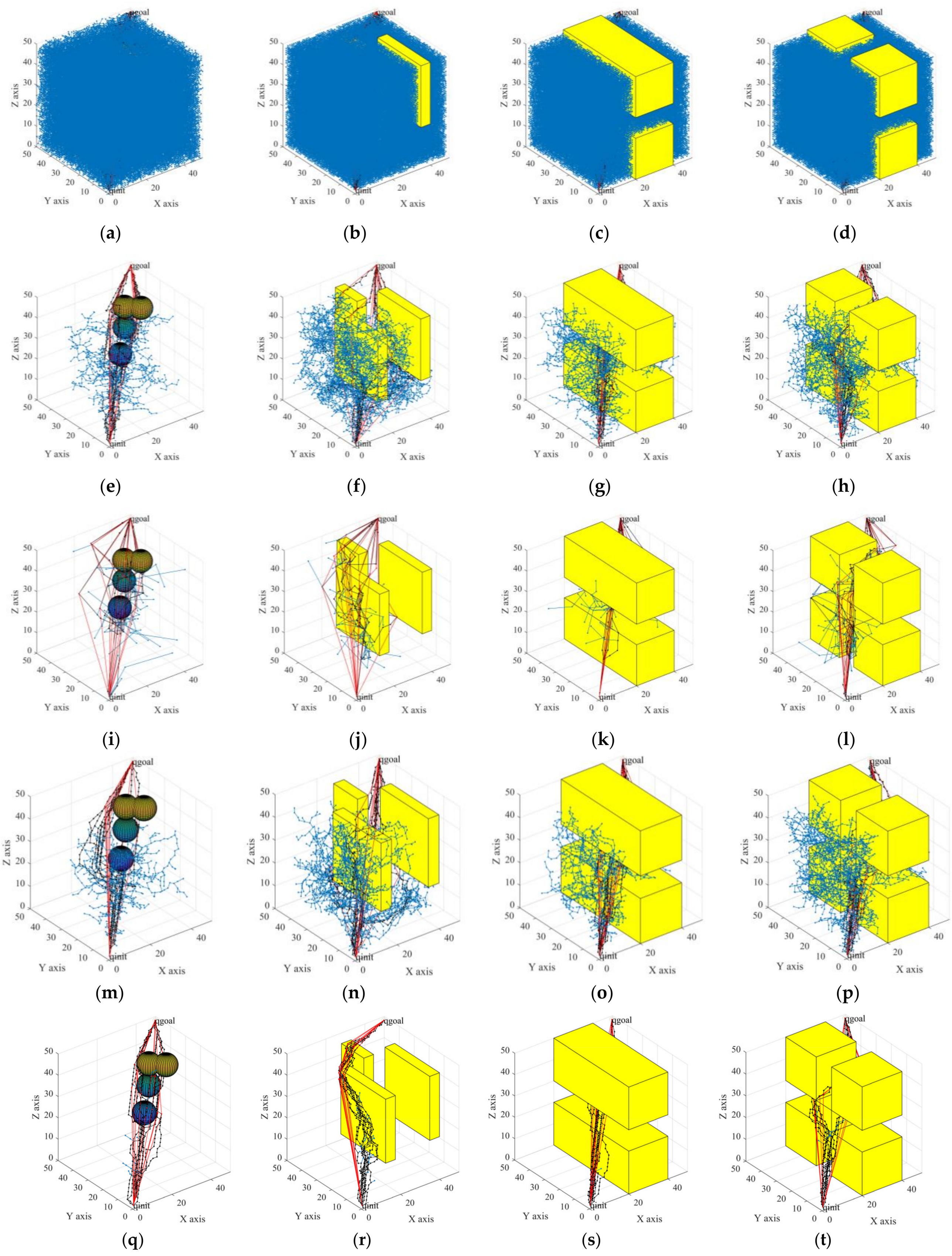
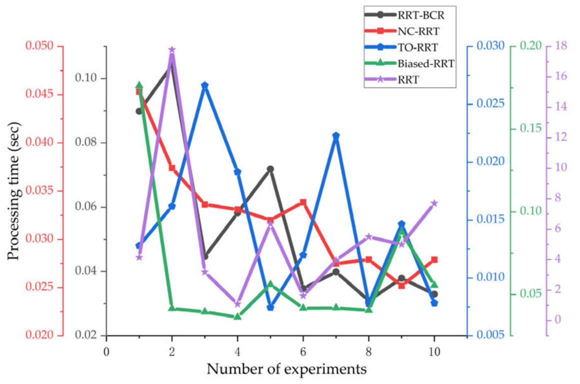
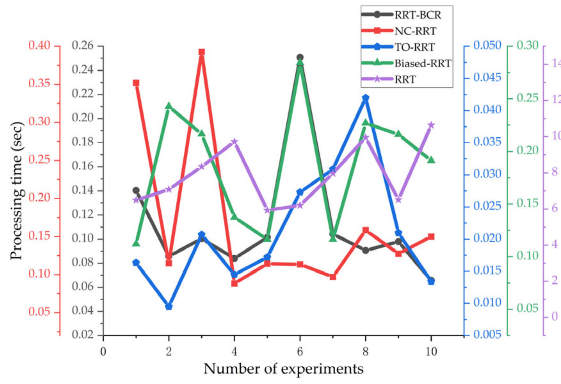


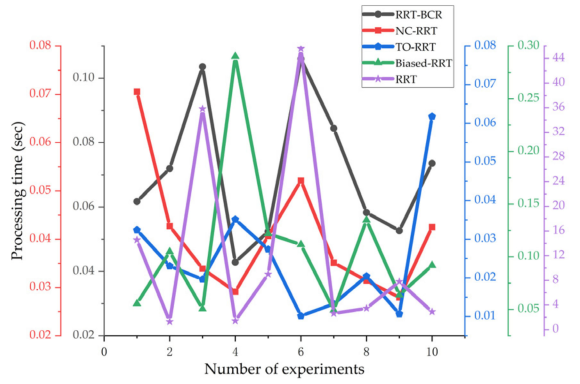
Figure 10. Ten experiments each of: the RRT algorithm (**a**–**d**); the biased-RRT algorithm with a target offset probability of 50% (**e**–**h**); the TO-RRT algorithm (**i**–**l**); the RRT-BCR algorithm (**m**–**p**); and the NC-RRT algorithm (**q**–**t**).



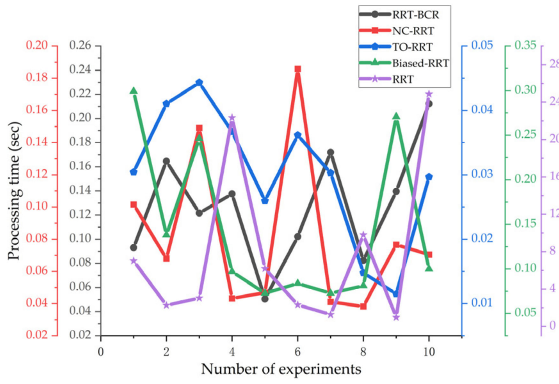
(**a**)



(**b**)



(**c**)



(**d**)

Figure 11. The running times of the RRT algorithm, the biased-RRT algorithm with a target offset probability of 50%, the TO-RRT algorithm, the RRT-BCR algorithm, and the NC-RRT algorithm. (**a**) Multi-sphere environment; (**b**) Multi-rectangle environment; (**c**) Single-channel environment; (**d**) Multi-channel environment.

Table 2. Experimental results of each algorithm in different environments.

	Algorithm Type	Running Time (s)	Path Length (cm)	Tree Nodes (Number)	Path Nodes (Number)	Collision Detection (Number)	Failed Node Growth (Number)	Node Failure Growth Rate (%)
Multi-sphere	RRT	5.6342	124.6008	10,454.3	60.9	10,693.7	229.4	2.15
	Biased-RRT	0.0617	100.1367	140.1	54	228	87.9	38.55
	TO-RRT	0.0147	100.9338	22.9	10.4	65.5	6.5	9.92
	RRT-BCR	0.0545	101.9241	113.4	54.3	123.2	9.8	7.95
	NC-RRT	0.0324	94.3765	50.6	50.2	183.7	133.1	78.46
Multi-rectangle	RRT	7.8822	140.9832	14,213.5	68.7	17,358.3	3144.8	18.12
	Biased-RRT	0.1860	125.8082	414.3	62.9	1033.9	619.6	59.93
	TO-RRT	0.0213	110.1866	32.7	13.2	135.5	17.3	12.77
	RRT-BCR	0.1121	121.8465	243.8	60.4	294.4	50.6	17.19
	NC-RRT	0.1709	107.2454	55.8	53.6	55,077	54,519	99.99
Single-channel	RRT	12.4436	131.1145	8333.9	64.2	13,560.3	5226.4	38.54
	Biased-RRT	0.1074	108.6431	242.2	55.5	607.3	365.1	60.12
	TO-RRT	0.0254	107.4978	32.8	12.7	130.7	20	15.30
	RRT-BCR	0.0707	109.4179	159.3	55.8	203.8	44.5	21.83
	NC-RRT	0.0406	96.7172	49.8	49.8	659	609.2	92.44

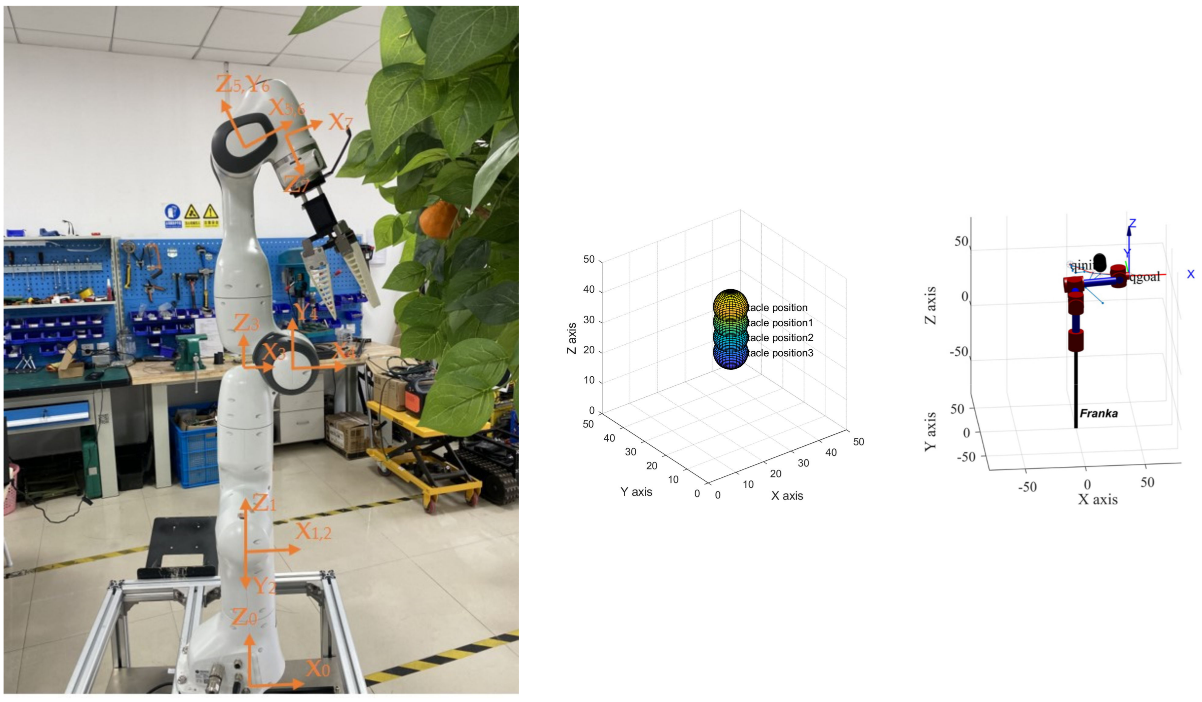
Table 2. *Cont.*

	Algorithm Type	Running Time (s)	Path Length (cm)	Tree Nodes (Number)	Path Nodes (Number)	Collision Detection (Number)	Failed Node Growth (Number)	Node Failure Growth Rate (%)
Multi-channel	RRT	8.0047	134.4688	11,702.5	64.7	18,399.5	6697	36.40
	Biased-RRT	0.1461	114.4721	322.9	56.9	861.9	539	62.54
	TO-RRT	0.0301	117.5516	51.8	16.1	222.5	35.2	15.82
	RRT-BCR	0.1276	120.4389	278.8	61.8	369.3	90.5	24.51
	NC-RRT	0.0821	102.7622	55.3	52.8	21,487	20,934	97.43
Average index	RRT	8.4912	132.7918	11,176.05	64.625	15,002.95	3824.4	23.8025
	Biased-RRT	0.1253	112.2650	279.875	57.325	682.775	402.9	55.285
	TO-RRT	0.0229	109.0425	35.05	13.1	138.55	19.75	13.4525
	RRT-BCR	0.0912	113.4070	198.825	58.075	247.675	48.85	17.87
	NC-RRT	0.0815	100.2753	52.875	51.6	19,351.675	19,061.48	92.08

Note: RRT, rapidly-exploring random tree; Biased-RRT, rapidly-exploring random tree with target Bias; TO-RRT, time-optimal rapidly-exploring random tree; RRT-BCR, Biased-RRT with boundary expansion mechanism and regression mechanism; NC-RRT, Node Control-RRT.

3.2. Obstacle Avoidance Test Based on the Robotics Toolbox

To verify the feasibility of the TO-RRT algorithm on the manipulator, Robotics Toolbox 10.2 in MATLAB was used to model the Franka manipulator. Franka is a 7-DOF robot with high precision and fast response. Its payload is 3 kg, and the maximum contact area is 855 mm. The Franka manipulator can realize two-way communication between itself and the workstation through the Franka Control Interface (FCI) and an Ethernet connection. Therefore, complete real-time control can be achieved with a sampling frequency of 1 kHz. In terms of picking performance, Franka's pose repeatability is within 0.1 mm. Even at the highest speed of 2 m/s, the path deviation can be ignored, which provides good working conditions for fruit picking. The physical object of the Franka manipulator and its D-H parameters are shown in Figure 12a and Table 3, respectively.



(**a**) (**b**) (**c**)

Figure 12. Materials and results of simulation experiments based on using Robotics Toolbox. (**a**) The physical object of the Franka manipulator; (**b**) Trunk model; (**c**) The Franka manipulator avoids obstacles.

Table 3. D-H parameters.

Link i	Link Offset $a_i(\mathrm{m})$	Link Length $d_i(\mathrm{m})$	Link Twist $\alpha_i(\mathrm{rad})$	Link Twist $\theta_i(\mathrm{rad})$
1	0	0.333	$\frac{\pi}{2}$	θ_1
2	0	0	0	θ_2
3	0	0.316	0	θ_3
4	0.0825	0	$\frac{\pi}{2}$	θ_4
5	-0.0825	0.384	$-\frac{\pi}{2}$	θ_5
6	0	0	0	θ_6
7	0.088	0	$\frac{\pi}{2}$	θ_7

To simplify the trunk and improve the operation speed of the TO-RRT algorithm, the trunk was regarded as a combination of spheres [29], as shown in Figure 12b and Table 4. To judge whether the manipulator collided with obstacles, the shortest distance $d_{collision}$ from the center of the sphere to the origin of the coordinate system of adjacent links of the manipulator was used. The three-dimensional coordinates of each joint of the manipulator were obtained through a forward kinematics solution, and if the manipulator did not collide with the tree trunk, the following conditions must be met:

$$d_{collision} > R + r \tag{3}$$

Table 4. Obstacle parameters.

Number	Obstacle Coordinates (cm)	Obstacle Radius (cm)
1	(25,55,48)	5
2	(25,53,47)	5
3	(25,51,46)	5
4	(25,49,45)	5

In the formula, $R = 5$ cm is the radius of the obstacle ball, and $r = 3$ cm is the radius of the cylinder.

Figure 12c shows the Franka manipulator using the TO-RRT algorithm to plan its path, and the minimum-snap trajectory optimization algorithm was used to smooth the trajectory of the manipulator [37,38]. Figure 13 shows the shortest distance.

3.3. Comparative Experiments in a Virtual Picking Environment

The motion-planning experiment of the Franka manipulator was initially realized through Robotics toolbox, which proved that the TO-RRT algorithm was feasible in the motion of the manipulator. MoveIt! was used in this section to build a virtual picking environment and to conduct comparative experiments on different algorithms in this environment. The experimental parameters are shown in Table 5.

During the experiment, the maximum search time was 10 min, the maximum number of failed searches was 10,000, and the search domain was $\{x, y, z| -1 < x < 1, -1 < y < 1, -1 < z < 1\}(\mathrm{m})$. Due to the large number of sampling points generated, the global search time of the RRT algorithm was 243.322451 s. Compared with the RRT algorithm, the search time of the biased-RRT algorithm was only 3.720342 s. However, affected by the nature of obstacles and the probability threshold, the collision-free path generated by the biased-RRT algorithm was less smooth. In contrast, since the NC-RRT algorithm controlled the sampling interval, its trajectory was the smoothest among all the algorithms. Compared with the previous algorithms, the TO-RRT search time and path length were only 0.074915 s and 0.63548128 m, respectively, due to the generation of smaller random trees. The simulation results are shown in Table 6 and Figure 14.

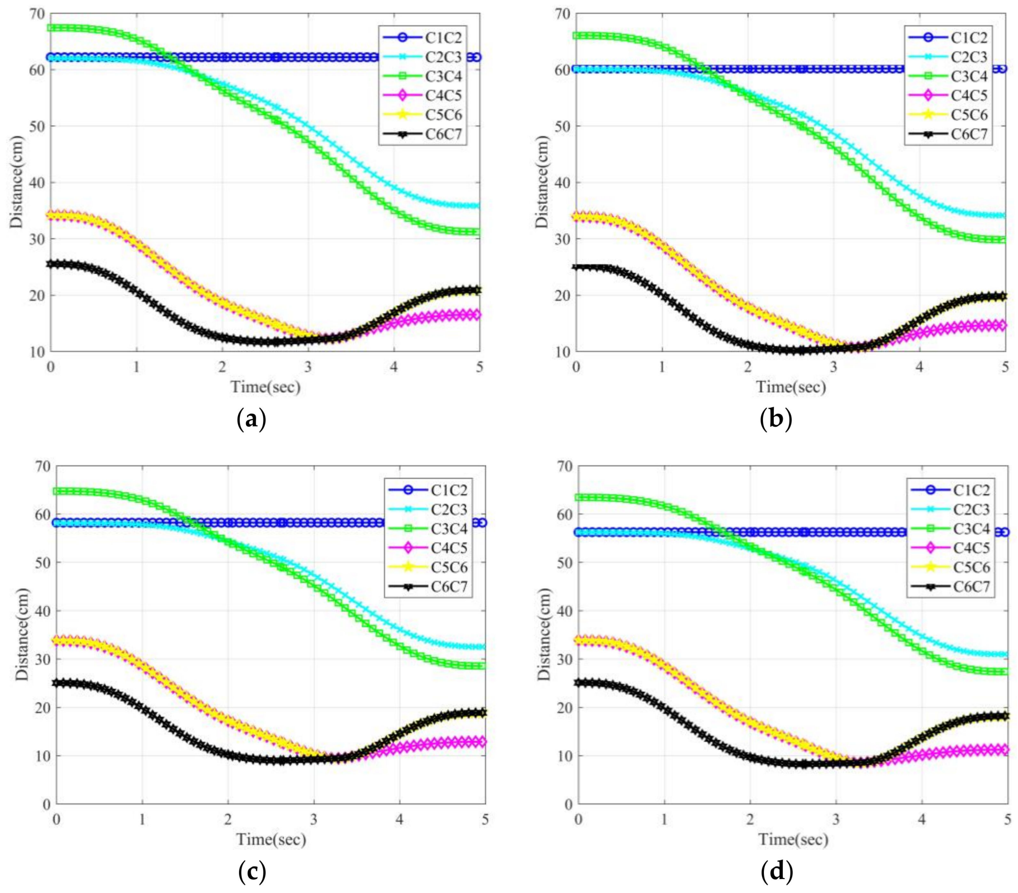


Figure 13. The shortest distance from the center of the sphere to the origin of the coordinate system of adjacent links of the manipulator. (**a**) Obstacle with coordinates (25,55,48); (**b**) Obstacle with coordinates (25,53,47); (**c**) Obstacle with coordinates (25,51,46); (**d**) Obstacle with coordinates (25,49,45). C_iC_{i+1} represents the distance between the line segment between coordinate system *i* and coordinate system *i* + 1 and the center of the sphere.

Table 5. Experimental parameters.

	Initial Pose	Pose of Citrus 1	Pose of Citrus 2
Position	(0.3595, 0, 0.643499)	(0.106155, 0.227978, 0.744871)	(−0.234434, 0.360095, 0.737649)
Orientation	(−0.65328, −0.270598, 0.653283, 0.270599)	(−0.636052, 0.309414, 0.231336, 0.66797)	(−0.771505, 0.309187, 0.226895, 0.507644)

Table 6. Experimental data using MoveIt!.

	RRT	Biased-RRT	TO-RRT	RRT-BCR	NC-RRT
Global planning time(s)	243.322451	3.720342	0.074915	1.222014	0.181070
Global waypoints(number)	41	29	7	20	15
Path length at obstacle avoidance(m)	1.89919096	1.46801193	0.63548128	0.592291	0.53239712

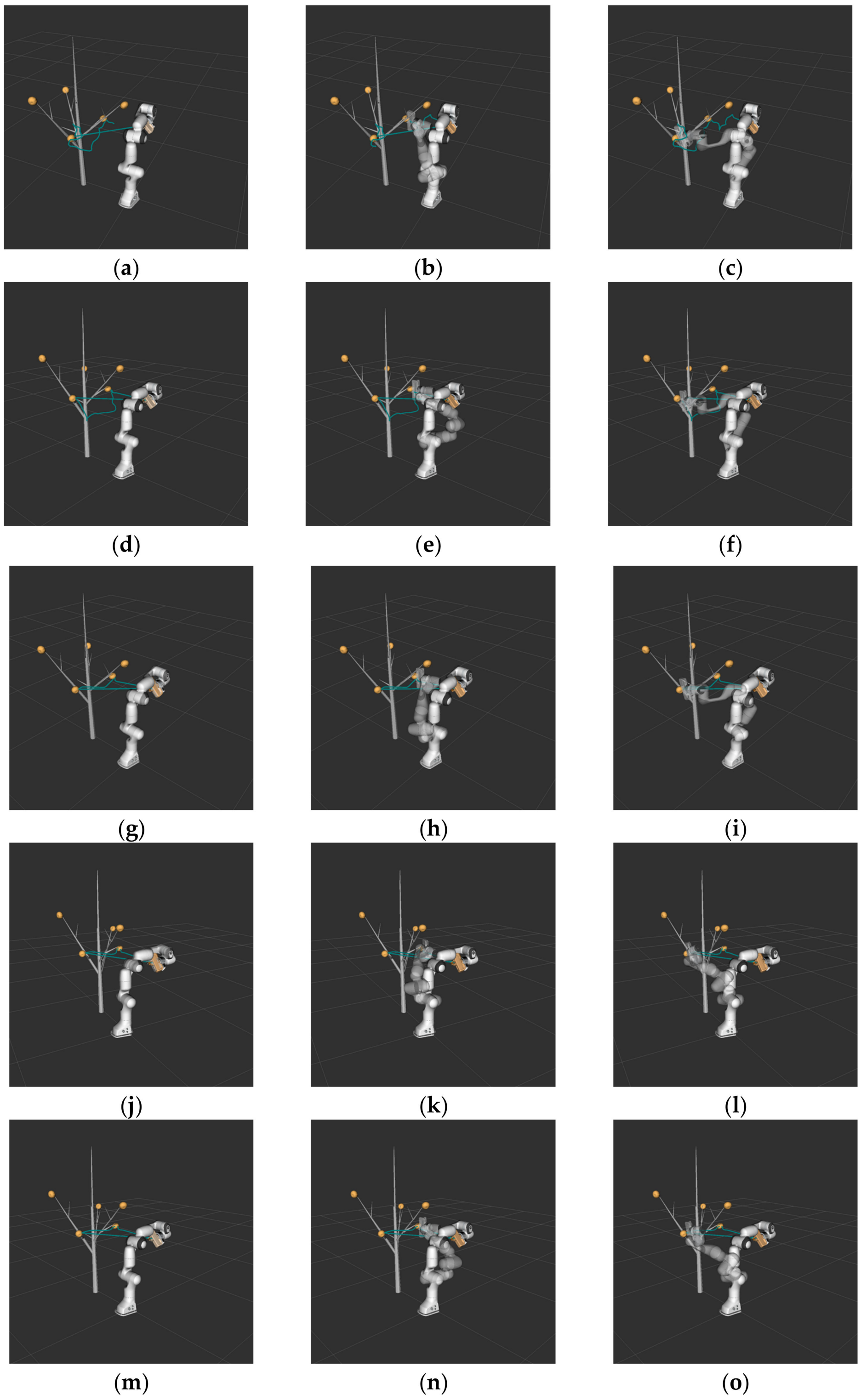


Figure 14. The use of MoveIt! with: the RRT algorithm (**a**–**c**); the biased-RRT algorithm with a target offset probability of 50% (**d**–**f**); the TO-RRT algorithm (**g**–**i**); the RRT-BCR algorithm (**j**–**l**); and the NC-RRT algorithm (**m**–**o**).

3.4. Contrastive Experiments in Real Environments

To test the performance of TO-RRT in actual picking, the Franka manipulator was taken as the moving object, the citrus as the operation object, and the tree trunk as the obstacle avoidance object to construct a multi-objective citrus-picking environment. The environmental parameters are shown in Tables 7 and 8. First, the manipulator adjusted its pose to the initial state, and its joint angle was $(0, -\frac{\pi}{4}, 0, -\frac{\pi}{2}, 0, \frac{\pi}{3}, 0)$. Second, the three-dimensional coordinates of the citrus, the parameter information of obstacles, and the picking pose of the manipulator were transmitted to the planning thread, and the continuous and collision-free trajectory was obtained through inverse kinematics. Finally, MoveIt! published the trajectory through moveit_commander to move_group and transmitted the control signal to the robot controllers to complete the picking action. The control block diagram is shown in Figure 15. The experimental results showed that the TO-RRT algorithm could be used to effectively reduce the nodes, shorten the planning time, and reduce the movement time of the manipulator, as shown in Figure 16 and Table 9.

Table 7. Obstacle information.

Number	Obstacle Coordinates (m)	Obstacle Radius (cm)
1	(0.369822, −0.153781, 1.04791)	1
2	(0.426765, −0.149826, 1.00189)	1
3	(0.45418, −0.186812, 0.947317)	1
4	(0.330284, −0.344084, 1.01095)	1.5
5	(0.384351, −0.371103, 0.94411)	1.5
6	(0.48388, −0.335959, 0.897789)	1.5

Table 8. Target information.

	Coordinates (m)
Base coordinates	(0,0,0)
Citrus 1 coordinates	(0.208763, −0.432806, 0.764728)
Citrus 2 coordinates	(0.423718, 0.0602042, 0.994)

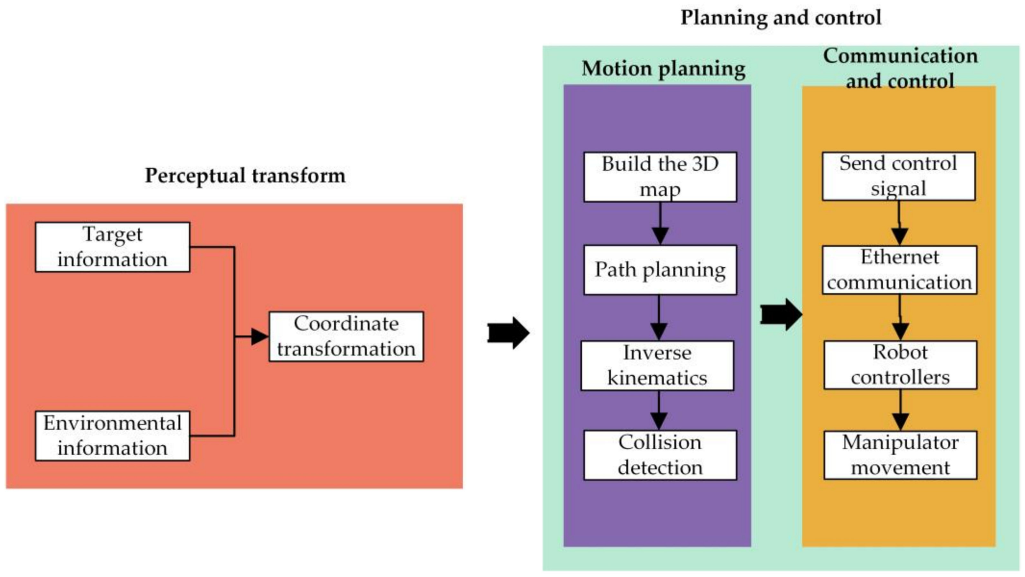


Figure 15. Control block diagram.

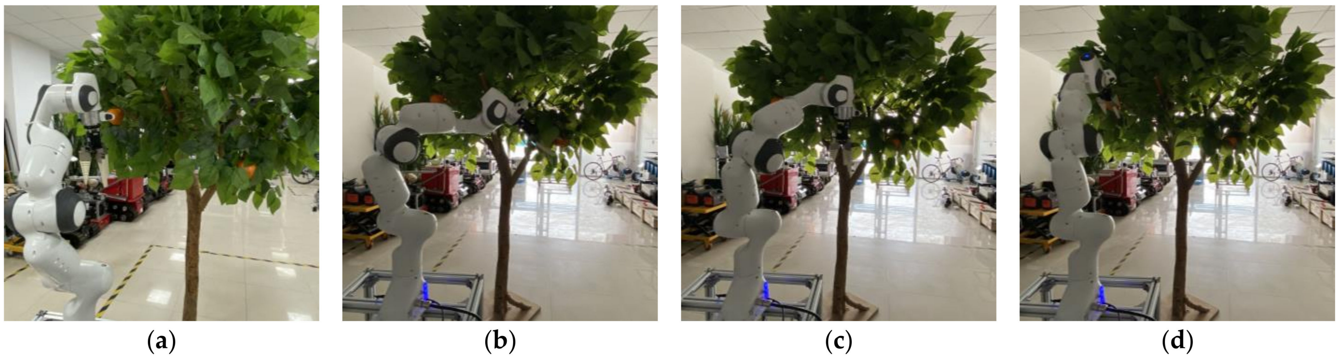


Figure 16. The manipulator reached Citrus 1 and Citrus 2 and avoided the branches. (**a**) Initial state of manipulator; (**b**) The manipulator reaches the first citrus; (**c**) Obstacle avoidance of the manipulator; (**d**) The manipulator reaches the second citrus.

Table 9. Comparison of the planning time and movement time.

Algorithm Type	Planning Time(s)	Movement Time(s)
RRT	53.873985	84.3975
Biased-RRT	0.0883	18.0498
TO-RRT	0.0508	17.3703
RRT-BCR	0.0771	17.9238
NC-RRT	0.0649	17.7131

4. Discussion

4.1. Analysis

From Figure 10a–d, since the RRT algorithm did not consider the effect of target offset probability, the entire workspace was searched in all environments. The above problems led to the huge scale of the random tree and caused more collision detection times. Therefore, the path length and movement time of the manipulator were the longest among all the algorithms, as shown in Tables 6 and 9. From Table 2, the biased-RRT algorithm avoided redundant searching through heuristic guidance, effectively reducing the number of tree nodes and collision detection times. From the average index in Table 2, since the RRT-BCR algorithm removed nodes that collided multiple times, its node failure growth rate was very low. However, this approach took a considerable amount of computation time, only 0.0112 s less than the biased-RRT algorithm, as shown in Table 9. From the average index in Table 2, the path length of the NC-RRT algorithm was the shortest, and the running time was second only to the TO-RRT algorithm. As can be seen from the multi-rectangle environment in Table 2, the NC-RRT algorithm had to continuously expand its sampling space when facing obstacles with large occlusion areas, resulting in 55,077 collision detections (which was the highest among all the algorithms). From Table 2, the TO-RRT algorithm reduced the numbers of path nodes and collision detections through an attractive step size, reduced the number of node failure growth through the node-first search strategy, and, finally, enhanced the escape ability through the regression superposition algorithm. However, the TO-RRT algorithm produced larger steps near obstacles, which led to a slightly longer path length than the other improved algorithms, as shown in Table 6.

4.2. Future Work

Industry 5.0 is a new generation of the industrial revolution representing "personalization", in which personalized products and services are created for humans by using the creativity of human experts to interact with efficient, intelligent, and precise machines. The key technologies of Industry 5.0, such as human–computer interaction, collaborative robots, and edge computing (EC), can provide ideas and technical support for Agriculture 5.0 [39].

As the number of China's aging population increases by the year, the number of rural employees has dropped sharply, and original agricultural production methods can no longer meet the development needs of the current citrus industry. Through the high integration of artificial intelligence and mechanical equipment, the transformation and upgrade of the production mode of China's agricultural industry can be realized. The improved method proposed in this paper can be used in the fields for picking robots and pruning robots and for the path planning of orchard patrol robots [40–42]. By analyzing the characteristics of a citrus tree environment, the work presented in this paper aimed to optimize the time required and improve it on the basis of a traditional algorithm to greatly shorten the planning time of the manipulator and reduce the movement time of the manipulator to a certain extent. However, the detection of obstacles is an objective challenge faced by this method.

In recent years, path planning through deep reinforcement learning (DRL) has become a research hotspot. A robot senses environmental information through sensors and trains the samples in the process of continuous interaction with the environment to complete an efficient, accurate, and low-environment-dependence path-planning method. The fusion of deep reinforcement learning and traditional path-planning algorithms has gradually become a research trend. For example, LM-RRT determines the selection probability of extension and connection trees based on reinforcement learning and guides the trees to pass through narrow channels quickly [43]. Based on this, the research on improving the TO-RRT algorithm by reinforcement learning will be discussed in the next stage.

5. Conclusions

A time-optimal RRT algorithm based on the characteristics of the complex environment of citrus trees was proposed in this paper. The constructed algorithm had an attractive potential field and a repulsive potential field for the target node and obstacle, respectively. In addition, dynamic adjustment of the probability threshold under the action of the superimposed potential field was achieved, and a node-first search strategy was used to solve the "falling into a trap" problem. In addition, an attractive step size and a "step-size dichotomy" were introduced in this algorithm so that the random tree could expand the step size as much as possible on the premise of reducing the number of collisions. Finally, a regression superposition algorithm was used to improve the search efficiency of the random tree in the range of the obstacle repulsive potential field. The TO-RRT algorithm was simulated in complex environments, and the motion-planning of the Franka manipulator was carried out using Robotics Toolbox and MoveIt! It can be seen from the simulation results that the TO-RRT algorithm had fewer tree nodes, collision detection times, and failed growth times, so this algorithm had a shorter planning time than the RRT algorithm, the biased-RRT algorithm, the RRT-BCR algorithm, and the NC-RRT algorithm, especially when the random tree faced a large obstacle area. To obtain the performance of the algorithm in real work, we built a real picking environment indoors. Through the performance evaluation of various indicators of the different algorithms, it was proved that the TO-RRT algorithm still had a good performance in movement time.

Author Contributions: Conceptualization, C.L., L.X. and Q.F.; methodology, C.L., L.X. and Q.F.; software, C.L., Z.T. and X.W.; writing—original draft preparation, C.L. and L.X.; writing—review and editing, C.L., L.X. and Q.F.; visualization, C.L., Z.T., J.G. and X.W.; supervision, L.X. and Q.F.; funding acquisition, L.X. and Q.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Beijing Science and Technology Plan Project: Z201100008020009; Key R&D project of Science and Technology Department of Sichuan Province: 2020YFN0025; Key Projects of Innovation and Entrepreneurship of Sichuan Science and Technology Department: 2021JDRC0091; Chengdu Technology Innovation R&D Project: 2021-YF05-01744-SN.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

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

References

1. Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. In Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA, 25–28 March 1985; pp. 396–404. [CrossRef]
2. LaValle, S.M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.1853&rep=rep1&type=pdf (accessed on 10 March 2022).
3. LaValle, S.M.; Kuffner, J.J. Randomized Kinodynamic Planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
4. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. Proceedings 2000 ICRA. Millennium Conference. In Proceedings of the IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; pp. 995–1001. [CrossRef]
5. Karaman, S.; Frazzoli, E. Incremental sampling-based algorithms for optimal motion planning. In Proceedings of the Robotics Science and Systems 2010, Zaragoza, Spain, 27 June 2010; Volume 104. Available online: http://www.roboticsproceedings.org/rss06/p34.pdf (accessed on 15 March 2022). [CrossRef]
6. Mohammed, H.; Romdhane, L.; Jaradat, M.A. RRT* N: An efficient approach to path planning in 3D for Static and Dynamic Environments. *Adv. Robot.* **2021**, *35*, 168–180. [CrossRef]
7. Akgun, B.; Stilman, M. Sampling heuristics for optimal motion planning in high dimensions. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 2640–2645. [CrossRef]
8. Jeong, I.B.; Lee, S.J.; Kim, J.H. RRT*-quick: A motion planning algorithm with faster convergence rate. In *Robot Intelligence Technology and Applications 3*; Intelligent Systems and Computing: Cham, Switzerland, 2015; pp. 67–76. [CrossRef]
9. Jeong, I.B.; Lee, S.J.; Kim, J.H. Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. *Expert Syst. Appl.* **2019**, *123*, 82–90. [CrossRef]
10. Adiyatov, O.; Varol, H.A. Rapidly-exploring random tree based memory efficient motion planning. In Proceedings of the 2013 IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Japan, 4–7 August 2013; pp. 354–359. [CrossRef]
11. Cao, X.; Zou, X.; Jia, C.; Chen, M.; Zeng, Z. RRT-based path planning for an intelligent litchi-picking manipulator. *Comput. Electron. Agric.* **2019**, *156*, 105–118. [CrossRef]
12. Wang, X.; Luo, X.; Han, B.; Chen, Y.; Liang, G.; Zheng, K. Collision-free path planning method for robots based on an improved rapidly-exploring random tree algorithm. *Appl. Sci.* **2020**, *10*, 1381. [CrossRef]
13. Zhang, H.; Wang, Y.; Zheng, J.; Yu, J. Path planning of industrial robot based on improved RRT algorithm in complex environments. *IEEE Access* **2018**, *6*, 53296–53306. [CrossRef]
14. Gong, H.; Yin, C.; Zhang, F.; Hou, Z.; Zhang, R. A path planning algorithm for unmanned vehicles based on target-oriented rapidly-exploring random tree. In Proceedings of the 2017 11th Asian Control Conference (ASCC), Gold Coast, QLD, Australia, 17–20 December 2017; pp. 760–765. [CrossRef]
15. Li, B.; Chen, B. An Adaptive Rapidly-Exploring Random Tree. *IEEE/CAA J. Autom. Sin.* **2021**, *9*, 283–294. [CrossRef]
16. Gao, X.; Wu, H.; Zhai, L.; Sun, H.; Jia, Q.; Wang, Y.; Wu, L. A rapidly exploring random tree optimization algorithm for space robotic manipulators guided by obstacle avoidance independent potential field. *Int. J. Adv. Robot. Syst.* **2018**, *15*, 1729881418782240. [CrossRef]
17. Wang, J.; Li, X.; Meng, M.Q.H. An improved RRT algorithm incorporating obstacle boundary information. In Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, China, 3–7 December 2016; pp. 625–630. [CrossRef]
18. Veras, L.G.; Medeiros, F.; Guimaraes, L. Systematic literature review of sampling process in rapidly-exploring random trees. *IEEE Access* **2019**, *7*, 50933–50953. [CrossRef]
19. Zu, W.; Fan, G.; Gao, Y.; Ma, H.; Zhang, H.; Zeng, H. Multi-UAVs Cooperative Path Planning Method based on Improved RRT Algorithm. In Proceedings of the 2018 IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, China, 5–8 August 2018; pp. 1563–1567. [CrossRef]
20. Li, H.; Liang, Y.; Wang, M.; Dan, T. Design and implementation of improved RRT algorithm for collision free motion planning of high-dimensional robot in complex environment. In Proceedings of the 2012 2nd International Conference on Computer Science and Network Technology, Changchun, China, 29–31 December 2012; pp. 1391–1397. [CrossRef]
21. Kang, G.; Kim, Y.B.; You, W.S.; Lee, Y.H.; Oh, H.S.; Moon, H.; Choi, H.R. Sampling-based path planning with goal oriented sampling. In Proceedings of the 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Banff, AB, Canada, 12–15 July 2016; pp. 1285–1290. [CrossRef]
22. Ahmadyan, S.N.; Kumar, J.A.; Vasudevan, S. Goal-oriented stimulus generation for analog circuits. In Proceedings of the 49th Annual Design Automation Conference, New York, NY, USA, 3 June 2012; pp. 1018–1023. [CrossRef]
23. Wang, J.; Wu, S.; Li, H.; Zou, J. Path planning combining improved rapidly-exploring random trees with dynamic window approach in ROS. In Proceedings of the 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, China, 31 May–2 June 2018; pp. 1296–1301. [CrossRef]

24. Qi, J.; Yang, H.; Sun, H. MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment. *IEEE Trans. Ind. Electron.* **2020**, *68*, 7244–7251. [CrossRef]
25. Yuan, C.; Zhang, W.; Liu, G.; Pan, X.; Liu, X. A heuristic rapidly-exploring random trees method for manipulator motion planning. *IEEE Access* **2019**, *8*, 900–910. [CrossRef]
26. Wen, N.; Zhang, R.; Liu, G.; Wu, J.; Qu, X. Online planning low-cost paths for unmanned surface vehicles based on the artificial vector field and environmental heuristics. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420969076. [CrossRef]
27. Qureshi, A.H.; Ayaz, Y. Potential functions based sampling heuristic for optimal path planning. *Auton. Robot.* **2016**, *40*, 1079–1093. [CrossRef]
28. Zaid, T.; Qureshi, A.H.; Yasar, A.; Raheel, N. Potentially guided bidirectionalized rrt* for fast optimal path planning in cluttered environments. *Robot. Auton. Syst.* **2018**, *108*, 13–27. [CrossRef]
29. Yang, Y.; Liu, J.; Zheng, Y.; Huang, Q. Obstacle Avoidance Path Planning of Manipulator of Forestry Felling & Cultivation Machine. *Sci. Silvae Sin.* **2021**, *57*, 179–192. [CrossRef]
30. Liu, Y.; Zhao, H.; Liu, X.; Xu, Y. An Improved RRT Industrial Robot Path Avoidance Planning Algorithm. *Inf. Control* **2021**, *50*, 235–246. [CrossRef]
31. Liu, C.; Han, J.; An, K. Dynamic Path Planning Based on an Improved RRT Algorithm for RoboCup Robot. *Robot* **2017**, *39*, 8–15. [CrossRef]
32. Li, Y.; Xu, D. Cooperative Path Planning of Dual-arm Robot Based on Attractive Force Self-adaptive Step Size RRT. *Robot* **2020**, *42*, 606–616. [CrossRef]
33. Ruan, X.; Zhou, J.; Zhang, J.; Zhu, X. Robot goal guide RRT path planning based on sub-target search. *Control Decis.* **2020**, *35*, 2543–2548. [CrossRef]
34. Lin, N.; Zhang, Y. An adaptive RRT based on dynamic step for UAVs route planning. In Proceedings of the 2014 5th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 27–29 June 2014; pp. 1111–1114. [CrossRef]
35. Wang, C.; Meng, Q.H. Variant step size RRT: An efficient path planner for UAV in complex environments. In Proceedings of the 2016 IEEE International Conference on Real-Time Computing and Robotics (RCAR), Angkor Wat, Cambodia, 6–10 June 2016; pp. 555–560. [CrossRef]
36. Li, Y.; Wang, S.; Jiang, L.; Meng, J.; Xie, Y. Motion Planning of Mobile Manipulator Based on RRT with Sparse Nodes. *China Mech. Eng.* **2020**, *32*, 9.
37. Mellinger, D.; Kumar, V. Minimum snap trajectory generation and control for quadrotors. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2520–2525. [CrossRef]
38. Richter, C.A.; Bry, A.P.; Roy, N. Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. In *Robotics Research*; Springer Tracts in Advanced Robotics; Springer: Cham, Switzerland, 23 April 2016; Volume 114, pp. 649–666. [CrossRef]
39. Reddy, P.; Pham, Q.V.; Prabadevi, B.; Deepa, N.; Dev, K.; Gadekallu, T.; Ruby, R.; Liyanage, M. Industry 5.0: A survey on enabling technologies and potential applications. *J. Ind. Inf. Integr.* **2021**, *26*, 100257. [CrossRef]
40. Gadekallu, T.R.; Rajput, D.S.; Reddy, M.P.K.; Lakshmanna, K.; Bhattacharya, S.; Singh, S.; Jolfaei, A.; Alazab, M. A novel PCA–whale optimization-based deep neural network model for classification of tomato plant diseases using GPU. *J. Real-Time Image Process.* **2021**, *18*, 1383–1396. [CrossRef]
41. Li, T.; Feng, Q.; Qiu, Q.; Xie, F.; Zhao, C. Occluded Apple Fruit Detection and Localization with a Frustum-Based Point-Cloud Processing Approach for Robotic Harvesting. *Remote Sensing.* **2022**, *14*, 482. [CrossRef]
42. Li, T.; Qiu, Q.; Zhao, C. Task planning of multi-arm harvesting robots for high-density dwarf orchards. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2021**, *37*, 1–10. [CrossRef]
43. Wang, W.; Zuo, L.; Xu, X. A learning-based multi-RRT approach for robot path planning in narrow passages. *J. Intell. Robot. Syst.* **2018**, *90*, 81–100. [CrossRef]