robotics

MDPI

Article

Double-Layer RRT* Objective Bias Anytime Motion Planning Algorithm

Hamada Esmaiel [1,2], Guolin Zhao [3], Zeyad A. H. Qasem [4], Jie Qi [3,*] and Haixin Sun [5]

[1] Department of Electronics and Communication Engineering, College of Engineering, A'Sharqiyah University, Ibra 400, Oman; hamada.esmaiel@asu.edu.om or h.esmaiel@aswu.edu.eg
[2] Department of Electrical Engineering, Faculty of Engineering, Aswan University, Aswan 81542, Egypt
[3] College of Electronic Science and Technology, Xiamen University, Xiamen 361005, China; glzhao@stu.xmu.edu.cn
[4] School of Electronic and Computer Engineering, Peking University, Shenzhen 518005, China; zeyadqasem@pku.edu.cn
[5] Department of Information and Communication, School of Informatics, Xiamen University, Xiamen 316005, China; hxsun@xmu.edu.cn
* Correspondence: qijie@xmu.edu.cn

Abstract: This paper proposes a double-layer structure RRT* algorithm based on objective bias called DOB-RRT*. The algorithm adopts an initial path with an online optimization structure for motion planning. The first layer of RRT* introduces a feedback-based objective bias strategy with segment forward pruning processing to quickly obtain a smooth initial path. The second layer of RRT* uses the heuristics of the initial tree structure to optimize the path by using reverse maintenance strategies. Compared with conventional RRT and RRT* algorithms, the proposed algorithm can obtain the initial path with high quality, and it can quickly converge to the progressive optimal path during the optimization process. The performance of the proposed algorithm is effectively evaluated and tested in real experiments on an actual wheeled robotic vehicle running ROS Kinetic in a real environment.

Keywords: motion planning; objective bias; online optimization; rapidly exploring random tree (RRT)

Citation: Esmaiel, H.; Zhao, G.; Qasem, Z.A.H.; Qi, J.; Sun, H. Double-Layer RRT* Objective Bias Anytime Motion Planning Algorithm. *Robotics* **2024**, *13*, 41. https://doi.org/10.3390/robotics13030041

Academic Editor: Marco Ceccarelli

Received: 13 January 2024
Revised: 11 February 2024
Accepted: 13 February 2024
Published: 1 March 2024

1. Introduction

Planning is often applied to determine how to move along; the solution should be provided in a way that complies with the mechanical limitations of autonomous systems, such as robotics [1,2], graphic animation [3], manufacturing [4], and minimally invasive surgical procedures [5]. In the state space, the problem of robot motion planning [6–8] involves starting in some initial state and trying to arrive at a specified goal instead of goal states to minimize resource consumption, such as energy or time.

Rapidly exploring random tree (RRT) [9] is one of the most popular approaches applied to handle nonholonomic problems of path planning in a high-dimensional configuration space. RRT's unique advantage is that it does not require any connections between configurations in pairs of states; hence, it can directly be applied to nonholonomic and Kino dynamic planning [10–17]. In robot navigation problems, RRT addresses the path planning problem by creating a random extended tree to find a suitable path, but, unfortunately, the RRT-suggested path can be a non-optimal one. To handle this issue, a modified RRT algorithm called RRT* has been proposed by Karaman and Frazzoli [18]. The RRT* algorithm tries to obtain the shortest path, whether by distance or other metrics. The modified RRT algorithm RRT* improves the path quality by introducing the major features of tree rewiring and a best neighbor search. However, it obtains asymptotic optimality at the expense of the execution time and convergence rate.

The RRT*-variant-based scheme has been proposed as an anytime scheme [19,20]. The main feature of the anytime scheme is that the planned time is unknown in advance, and

it can be terminated at any time. When time permits, any solution should be obtained as soon as possible, and then the chosen solution should be updated to be more conventional. The RRT*-variant anytime algorithm's [19,20] criteria are to firstly and quickly obtain some motion plans; these plans are not necessarily optimal but feasible. Then, RRT* is used to improve the path quality online over time. The lower bound tree-RRT (LBT-RRT) [21] allows continuous interpolation between RRT, RRT*, and RRG and achieves high-quality anytime motion planning by maintaining two roadmaps at the same time. The RRR^X proposed in [22] first obtains the initial plan and refines it to be the best solution through continuous repair. In [23], a doubletree RRT* (DT-RRT*) has been proposed based on a dual-tree structure to separate the expansion process from the optimization process. DT-RRT* initiates the path through the expansion process and the shortcut principle is used for optimization. However, unfortunately, the current anytime algorithms cannot support the new generation of unmanned driving systems due to the long latency time, in addition to inaccurate path selection metrics as the smoothing path quality is not considered in these schemes.

To cope with the new generation of unmanned driving systems' requirements, this paper proposes a real-time anytime planning algorithm suitable for unmanned driving systems. To address the drawbacks of the current anytime motion planning schemes, the proposed anytime planning algorithm adopts the initial path and online-optimized double-layer structure for motion planning, considering the latency time in addition to the selected path smoothing level. In the proposed scheme, the first layer of the RRT* scheme uses a feedback-based biased sampling strategy to obtain the initial path and uses segment forward pruning processing to evaluate the smoothing level of the selected initial path. The second layer in the RRT* scheme performs the online optimization of the selected initial path, which adopts the reverse maintenance spanning-tree scheme. In summary, the main contributions of this paper are as follows.

(1) This paper proposes a double-layer RRT* tree structure. In the first layer, initial path information is provided. In the second layer, an iterative optimization process is used to update the selected path and improve the path selection accuracy.
(2) A new feedback-based objective bias strategy is used to obtain the initial path, and the initial path is smoothed by removing redundant processing through segmentation forward pruning.
(3) The second layer of the RRT* scheme optimizes the selected path using the heuristic information provided by the spanning tree structure of the initial path.

Compared to the conventional RRT and RRT*, the proposed algorithm can allow the path to gradually approach the optimal solution while ensuring the running speed.

The rest of the paper is as follows. In Section 2, the problem of motion planning is stated. In Section 3, the initial path generation algorithm is described in detail. In Section 4, the optimization algorithm is described in detail. In Sections 5 and 6, the performance of the proposed scheme is assessed in an experiment in a real environment. Finally, the paper is concluded in Section 7.

2. Motion Planning Problem Statements

2.1. Robot Model and Control Input

The two-wheel differential robot model discussed in this paper is shown in Figure 1. The dynamic models of the two-wheel model are described as follows.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_r \\ u_l \end{bmatrix}, \tag{1}$$

where $\dot{\mathbf{x}} = (\dot{x},\ \dot{x}, \theta)$ is the state vector, which is used as the control input of the robot. This state vector is determined by the Cartesian coordinate system (x, y), the operation θ, and

the dynamic constraints u_r and u_l. In addition, two-wheeled differential robots generally have no backward motion and can support rotation in place.

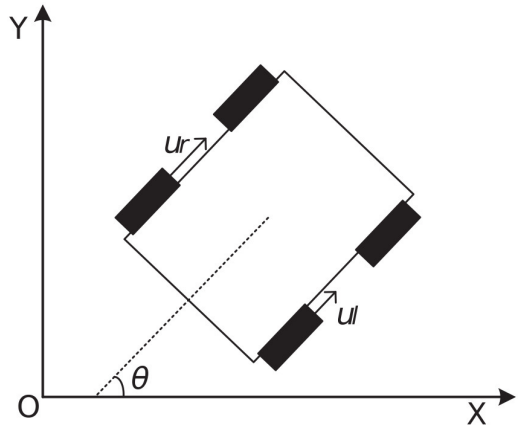


Figure 1. The schematic diagram of the differential motion model.

2.2. Problem Statement

The motion planning problem is used to calculate a continuous path from the initial configuration q_{start} to the goal state q_{goal}, avoiding collision with existing obstacles. Assuming that the environment and the geometry of the robot are described in an n-dimensional state space, the motion plan can be expressed as a path in the state space. Let $q \subseteq R^n$ n-dimensional configuration space C [24] and C_{obs} is the obstacle region. Therefore, $C_{free} = C/C_{obs}$ can be used to represent the free region of the configuration space. In the RRT algorithm, the spanning tree is represented as $T = (V,\ E)$, where V is the set of vertices of the spanning tree in the configuration space and E is the set of edges between vertices. RRT expands the spanning tree T by searching for random samples in the configuration space. The goal of the exercise plan is to return the trajectory $\tau(t) : [0, s] \rightarrow C_{free}$, where $\tau(0) = q_{start}$, $\tau(s) = q_{goal}$, corresponding to the control input causing the robot to move from q_{start} to q_{goal}.

3. Initial Path Generation Algorithm

In this section, we introduce and discuss the proposed initial path acquisition algorithm using an objective bias strategy and segmented forward pruning.

3.1. Initial Path Generation

Before navigation, the initial path is generated by the first layer in RRT*. Algorithm 1 shows the overall flow of the proposed initial path generation algorithm. The input parameters of the algorithm include the binary map of the environment map, the initial point q_{start}, and the goal point q_{goal}. The following terms are used to describe the algorithm.

$ObjBiasSample()$: returns the random node $q_{rand}(x, y)$ in the free configuration.

$FindQNearest(q,\ V)$: finds the nearest node to $q_{rand}(x, y)$ from the current spanning tree by Euclidean distance.

$FindQNearSets(q,\ V)$: returns the set of all nodes within r range near q_{rand} from the current spanning tree.

$Steer(q_1, q_2)$: a partial path from q_1 to q_2.

$CollisionDetection\ (q_1, q_2)$: connects q_1 and q_2 and judges whether the connection has an intersection with the obstacle.

$ChooseParent\ (q_1, q_2)$: finds the nearest neighbors within a defined radius around the node q_1 as a candidate to replace the parent node of q_2.

$Rewire\ (q_1, q_2)$: after reselecting the parent node for q_1, performs rewiring to ensure that the cost of the connection between the random tree nodes is as small as possible.

$Cost(q)$: returns the cost value of the path from the root nodes of the spanning tree to q.

$DistanceCost(q_1, q_2)$: returns the cost value of the path between q_1 and q_2.

$PathPruning(V,\ E)$: uses the segment forward pruning strategy to optimize the path.

Algorithm 1. $T =$ Build Initial Path $\left(map, q_{start},\ q_{goal}\right)$.

1: $V \leftarrow q_{start}; E = \varnothing; T = (V, E);$
2: for i from 0 to K do
3: $q_{rand} \leftarrow ObjBiasSample();$
4: $q_{near} \leftarrow FindQNearest(V, q_{rand});$
5: $q_{new} \leftarrow Steer(q_{near}, q_{rand});$
6: if $CollisionDetection\ (q_{near}, q_{new})$ then
7: $q_{new} = q_{temp};$
8: else
9: $Continue;$
10: end if
11: $Q_{near} \leftarrow FindQNearSets(q_{new}, V, r);$
12: for $q_{near} \in Q_{near}$ do
13: $T \leftarrow ChooseParent(q_{near}, q_{near});$
14: $T \leftarrow Rewire(q_{near}, q_{near});$
15: end for
16: $C_{cur} \leftarrow Cost(T);$
17: if $Distance\left(q_{new}, q_{goal}\right) \leq d_{min}$ then
18: return $T = (V, E);$
19: end if
20: end for
21: $T \leftarrow PathPruning(V, E);$

The difference between the proposed initial path generation algorithm and the conventional RRT* algorithm is mainly in the method of selecting sampling points and adding post-prune processing. $ObjBiasSample()$ in the third line of Algorithm 1 is the objective-biased sampling strategy. After q_{new} is added to the tree at the minimum cost, the rewire optimization process is performed. First, obtain the set Q_{near}, which is the node within the distance r around q_{new}. Then, calculate the cost for each q_{near} in the set Q_{near} and select the node with the smallest cost as the parent node of q_{new}. Finally, check whether q_{new} can be the parent node of each q_{near}; if yes, connect the new line. As the tree expands to q_{goal}, the algorithm returns the path node. Because of the existence of multiple redundant nodes on this path, a segmented forward pruning strategy is proposed for optimization, as shown in the 14th line of Algorithm 1.

3.2. Objective-Biased Sampling Strategy

Figure 2 shows the node expansion strategy of the conventional RRT*; the node q_{start} is the initial position, and the node q_{near} is a node that has been added to the spanning tree. As shown in Figure 2a, the node q_{rand} is a random sampling point in C_{free}. The new node q_{new} in Figure 2b is generated by RRT* to the random sampling point q_{rand} with an appropriate extension length. Due to the global randomness of the points taken by the conventional RRT*, it often leads to an excessive search on the map.

The feedback-based biased sampling strategy proposed in this paper uses the probability P to refer to the node q_{prev} that is added to RRT* as a reference and reduces the sampling area of q_{rand} by q_{prev} for a relative position and the goal state, enhancing the inspiration of the goal state. The relationship between q_{prev} and the goal state q_{bias} is used as a reference for the next selected point, as shown in Figure 3. At this time, q_{bias} is located in the upper right corner of q_{prev}, choosing a quarter-circle area with q_{prev} as the center. The distance between q_{prev} and q_{bias} is considered as the radius of the sampling point selection area. By this method, under the premise of ensuring random sampling, there is a probability

P of obtaining biased sampling points, avoiding unnecessary or excessive searches and speeding up the acquisition of the initial path.

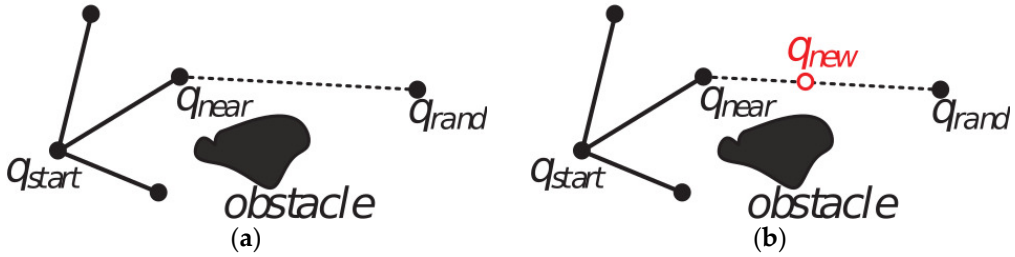


Figure 2. The schematic diagram for the node expansion of the convolutional RRT* scheme. (**a**) The robot at the nearest position q_{near}; (**b**) the new position of the robot generated by RRT* to move from the nearest position q_{near} to the random sampling point q_{rand}.

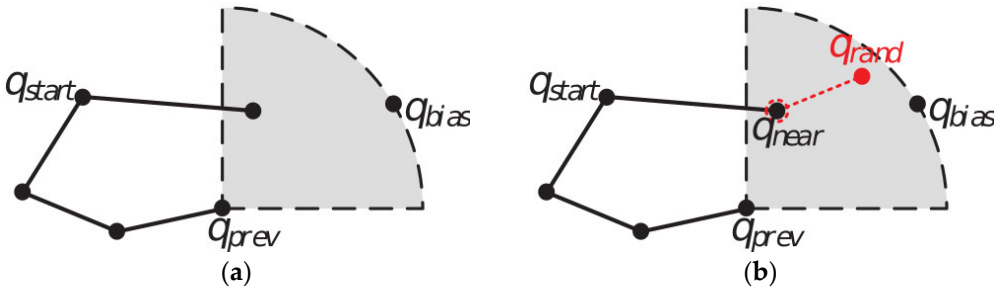
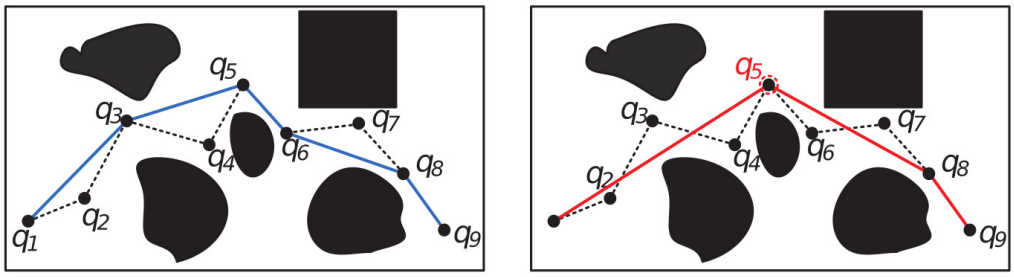


Figure 3. The schematic diagram of the objective-biased sampling strategy and choosing a quarter-circle area with q_{prev} as the center. (**a**) The robot at the nearest point to the previous position; (**b**) The random position for a robot RRT* used as a random sampling point q_{rand} in the quarter-circle.

3.3. Segmented Forward Pruning

Under normal circumstances, the paths obtained through the RRT* and RRT*-variant algorithms are generally more complicated and tortuous paths, including more unnecessary and redundant nodes. To ensure the high-speed navigation of the robot, the generated path must remove as many nodes as necessary under the premise of ensuring no collision. The algorithm uses pruning to obtain a shorter and less tortuous initial path. The short-cutting approach [25] has been widely used to eliminate the pruning operation of redundant nodes. In Figure 4, let q_1 be the starting point, q_9 the endpoint, and the dotted line segment as the initial path. The short-cutting approach starts from q_1 to connect other nodes in sequence, i.e., connect q_1 to q_3 first and judge whether the connection is legal. If there is no intersection with the obstacle, delete the other nodes between q_1 and q_3. Until the line collides, i.e., q_1 and q_4 intersect with the obstacle, repeat the above operation with q_3 as a new starting point until finally reaching the goal state q_9. The resulting path is shown in the blue line in Figure 4a. An obvious defect of this pruning method is that the current connection stops prematurely. For example, if q_1 and q_4 cannot be connected, the connection to the next starting point q_3 is entered. However, we can see that q_1 and q_5 can be connected. Such a result will lead to poor pruning. We modify the short-cutting approach and propose a segmented forward pruning approach. As shown in Figure 4b, the node at the midpoint is selected to divide the path into two segments, i.e., one segment from the start point to the midpoint and one segment from the midpoint to the endpoint. The sequential connection is changed to the reversed connection, i.e., we first determine whether q_1 and q_6 can be connected. If the connection is illegal, the search will continue to determine the connection between q_1 and q_5, and so on. This method can effectively avoid the situation in which the redundancy removal of the traditional pruning method is not thorough enough.



(**a**) Short-cutting approach. (**b**) Segmented forward pruning approach.

Figure 4. The schematic diagram for the path-pruned approach. (**a**) The selected robot path from the start point to end point based on the short-cutting approach starts; (**b**) The selected robot path from the start point to end point by using segmented forward pruning approach.

4. Optimization Algorithm

In this section, we introduce the proposed path optimization algorithm using a reverse maintenance strategy.

4.1. Optimization Algorithm

After submitting the initial path, the robot obtains an initial motion plan. Since only one path is solved, the path cannot guarantee the optimal solution. Therefore, we make the path asymptotically optimal through the optimization algorithm, which uses a reverse maintenance strategy. The optimization process will continue until the robot reaches the goal state. Algorithm 2 shows the overall flow of the optimization algorithm. The algorithm initializes by q_{goal} as the root node q_{root}; then, in the fourth line, $UpdateLocation()$ updates the robot position; in the fifth line, $HeuristicSample()$ uses the heuristic information provided by the initial path for sampling; and the eighteenth line calculates the cost of the current path.

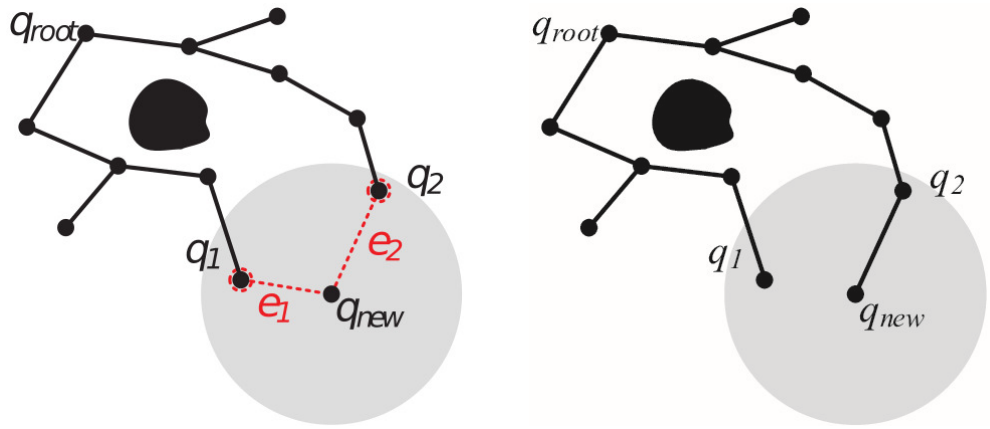
Algorithm 2. $T = \text{Optimization}\left(map, T_{Initial}, q_{goal}\right)$.

1: $T \leftarrow T_{Initial}; q_{root} \leftarrow q_{goal};$
2: $C_{min} \leftarrow Cost(T);$
3: **while** $NotreachGoal$ **do**
4: $UpdateLocation();$
5: $q_{rand} \leftarrow HeuristicSample();$
6: $q_{near} \leftarrow FindQNearest(V, q_{rand});$
7: $q_{temp} = Steer(q_{near}, q_{rand});$
8: **if** $CollisionDetection(q_{near}, q_{new})$ **then**
9: $q_{new} = q_{temp};$
10: **else**
11: $Continue;$
12: **end if**
13: $Q_{near} \leftarrow FindQNearSets(q_{new}, V, r);$
14: **for** $q_{near} \in Q_{near}$ **do**
15: $T \leftarrow ChooseParent(q_{new}, q_{near});$
16: $T \leftarrow Rewire(q_{new}, q_{near});$
17: **end for**
18: $C_{cur} \leftarrow Cost(T);$
19: **if** $C_{cur} < C_{min}$ **then**
20: $C_{min} \leftarrow C_{cur};$
21: $Trajectory \leftarrow TractionBSpline(T);$
22: **end if**
23: **end while**

4.2. Reverse Maintenance Strategy

The second layer of the RRT* scheme is a reverse maintenance strategy, i.e., using q_{goal} as a root node of DOB-RRT* to expand, which makes it easy for the planner to calculate $CostToGo()$ and obtain the residual path cost. In the proposed DOB-RRT* scheme, the heuristic information provided by the initial path will be used with a certain probability P based on the initial path node and reference node q_{ref}. The sampling points are obtained through the relative positions of q_{ref} and q_{start}, and the sampling area selection will be consistent with the selection method explained in Section 3.

In the proposed DOB-RRT* scheme, the cost function is modified to consider the *Euclidean* distance and trajectory angle to improve the rationality of the cost function. In practical applications, the path generated by the algorithm needs to be as smooth as possible, which can reduce the speed change of the mobile robot and increase the execution speed. Therefore, the optimal path not only needs to consider the path length but also the smoothness of the path. As depicted in Figure 5, in the selection process of the parent node and after a new node q_{new} is obtained, the candidate parent nodes are q_1 and q_2, and the extension paths from the root node q_{root} to q_{new} are e_1 and e_2, respectively. If we consider the *Euclidean* distance and trajectory angle to select q_2 as a parent node of q_{new}, this means that although the path length of e_1 is shorter than that of e_2, the path of e_2 is smoother than that of e_1.



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

Figure 5. The schematic diagram for selection of the parent node. (**a**) Selection process of the parent node (parent nodes are q_1 and q_2); (**b**) The extension paths from the root node q_2 to q_{new} through the extension path e_2 (e_2 is selected as its smoother).

Hence, the path of e_2 will reduce the speed change of the robot in practical applications. Because the distance and angle are different types of data, they need to be normalized before comprehensive consideration. The cost function is defined as

$$Cost(q_{new}, q_i) = \lambda_d \frac{d(q_{new}, q_i) - d_{min}}{d_{max} - d_{min}} + \lambda_\theta \frac{\theta(q_{new}, q_i) - \theta_{min}}{\theta_{max} - \theta_{min}}, \tag{2}$$

$$\lambda_d + \lambda_\theta = 1 \quad and \ \lambda_d, \lambda_\theta \in [0, 1], \tag{3}$$

where $d(q_{new}, q_i)$ is the *Euclidean* distance between the new node q_{new} and node q_i, and d_{max}, d_{min} represent the maximum and minimum distances between q_{new} and nearby nodes, respectively. The angle value of the path between the new node q_{new} and node q_i is $\theta(q_{new}, q_i)$, and the maximum and minimum distances of the angle value of the path between q_{new} and nearby nodes are expressed as θ_{max} and θ_{min}, respectively. The weights of the distance and angle in the cost function are λ_d and λ_θ, respectively. The cost function can be adapted to different requirements by modifying the weights of the distance and

angle. The substitute value of each node is a cumulative sum of the costs from the current node to the root node.

5. Simulation and Experimental Results

In this section, we conduct some numerical simulations to test the effectiveness of our proposed motion planning algorithm. The first simulation compares the initial path generation method with the RRT algorithm, and the other simulation verifies the feasibility of the DOB-RRT* algorithm. The simulation is performed in MATLAB 2020a, and the computer used is an i7-8700 CPU with 16G memory.

5.1. Initial Path Generation Simulation

First, we verify the effectiveness of the initial path generated by DOB-RRT* and select RRT and RRT* for comparison. We choose two maps for simulation. One is a map with dense obstacles and the other is a map with only two obstacles, and the size of both maps is 500×500. In addition, the initial path generated by DOB-RRT* uses a segmented forward pruning strategy to optimize the path length. To achieve the contrast effect, we also perform the same pruning process on the paths generated by RRT and RRT*.

Figure 6 illustrates the paths generated by 10 simulations for each algorithm. Tables 1 and 2 show the result statistics of 100 simulations for each algorithm, including the time required to generate the path and the quality of the selected path, which is represented by the length of the path. The simulation results show that RRT takes the smallest amount of time to generate a path, but RRT cannot guarantee the path quality. RRT* can guarantee the path quality better than RRT, but it takes a long time. Although DOB-RRT* requires slightly more time to generate a path than RRT, it is better than the conventional RRT scheme in terms of path quality. Compared with RRT*, the proposed DOB-RRT* is better than RRT* in terms of the time required to generate the path, while the path quality is similar to that of RRT*. In addition, DOB-RRT* has better stability in maps with dense obstacles. In summary, these results show that DOB-RRT* is intermediate between RRT and RRT* and it is feasible to obtain high-quality initial paths within an acceptable latency time.

Table 1. Two obstacles: map planning results.

		Mean	**Min**	**Max**
RRT	Path Time (s)	0.23	0.13	0.41
	Plan Length	749.52	697.15	958.43
RRT*	Path Time (s)	3.81	1.51	9.22
	Plan Length	694.15	673.88	888.41
DOB-RRT*	Path Time (s)	0.42	0.26	1.13
	Plan Length	697.91	674.29	892.13

Table 2. Dense obstacles: map planning results.

		Mean	**Min**	**Max**
RRT	Path Time (s)	0.48	0.26	0.98
	Plan Length	711.71	656.1	811.21
RRT*	Path Time (s)	3.81	1.51	9.22
	Plan Length	664.71	640.94	721.05
DOB-RRT*	Path Time (s)	0.68	0.44	1.24
	Plan Length	660.21	642.47	726.72

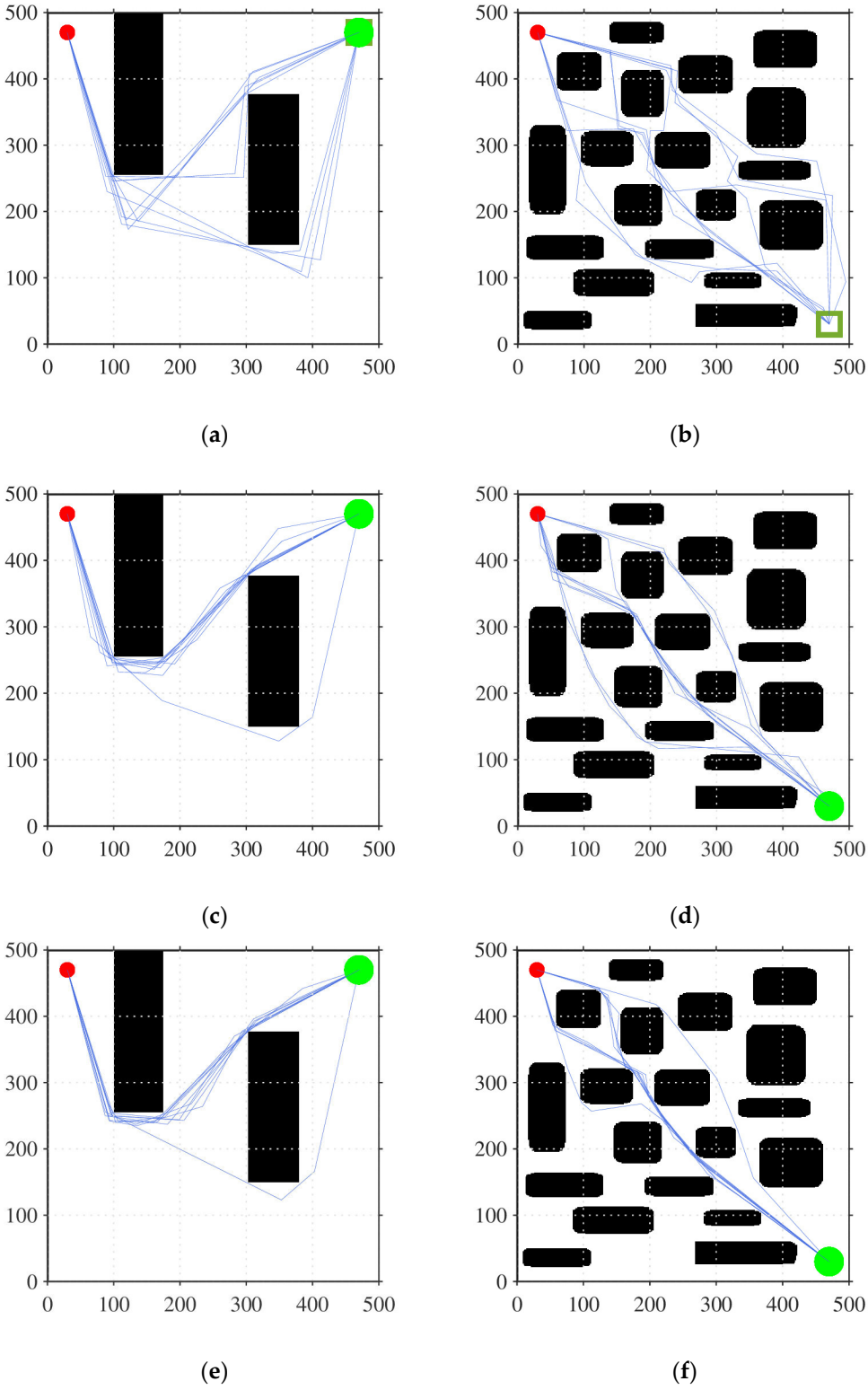


Figure 6. The simulation results of the two maps; the red circle in the figure represents the initial point, and the green area represents the goal area. (**a**,**b**) belong to the RRT scheme; (**c**,**d**) belong to the RRT* scheme; (**e**,**f**) belong to the proposed DOB- RRT* scheme.

5.2. Optimization Simulation

We choose a map with dense obstacles as an environment for the online optimization simulation with a map size of 500 × 500 (pixels). Considering the simulated robot as a point, the maximum linear velocity of the robot is set to be 1 m/s, and the maximum acceleration

is set to be 0.5 m/s^2. The planner first needs to find a feasible path from the initial point shown by the red dot in the upper left corner of the environment to the goal area shown as a green area. From our simulation and verification of the effectiveness of the proposed DOB-RRT* in online optimization, we select RRT* for comparison.

The simulation result of the proposed DOB-RRT* in real-time optimization is shown in Figure 7. The yellow dot in Figure 7 represents the current position of the robot, and the blue line represents the currently executed plan. Figure 7a shows the initial path and state tree structure, allowing the robot to travel along a relatively expensive path. When the robot starts to execute the plan, the number of states optimized online obtains a lower-cost route; see Figure 7b. The process of online optimization will continue until the robot reaches the goal area. We compare the proposed DOB-RRT* scheme with the conventional RRT* in terms of the generated path; the simulation results of the online optimization based on RRT* are shown in Figure 8. The simulation results of each algorithm are listed in Table 3. T_i represents the time required to calculate the initial path, and T_o represents the average time required for online optimization.

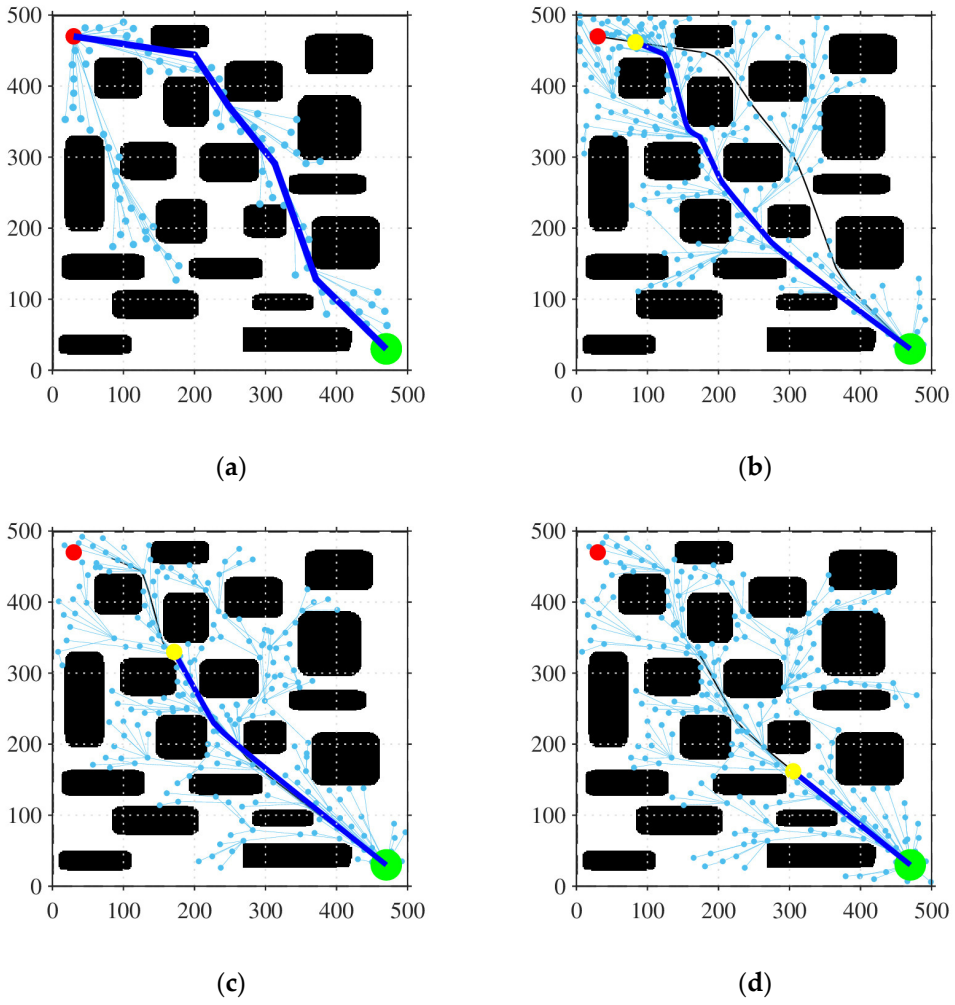


Figure 7. The DOB-RRT* simulation results. (**a**) The initial path and state tree structure; (**b**) the first online-optimized lower-cost route before 100 pixels; (**c**) the online-optimized lower-cost route within 100~200 pixels; (**d**) the online-optimized lower-cost route at 300 pixels.

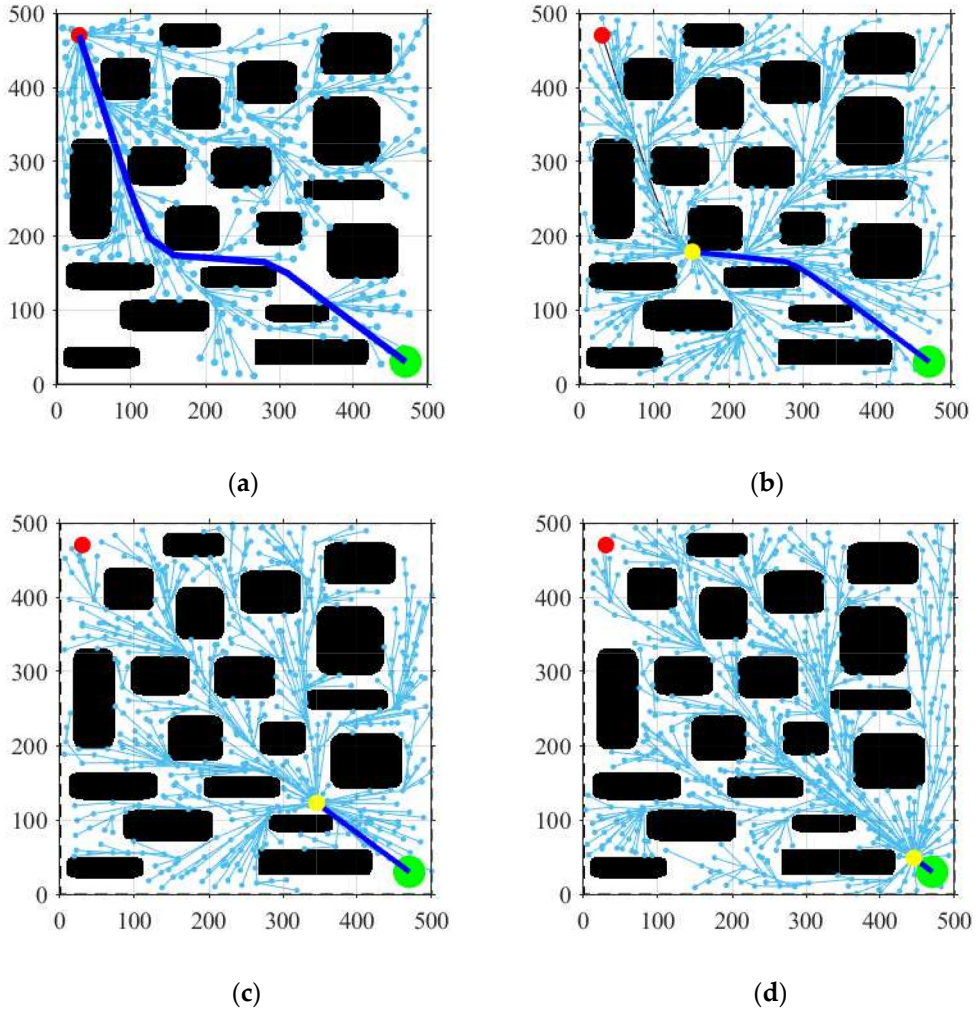


Figure 8. The simulation results of the anytime algorithm based on the RRT* scheme. (**a**) The initial path and the selected path; (**b**) the robot in the selected path before 100 pixels; (**c**) the robot in the selected path within 100~200 pixels; (**d**) the robot in the selected path at 300 pixels.

Table 3. Planning results of optimization simulation.

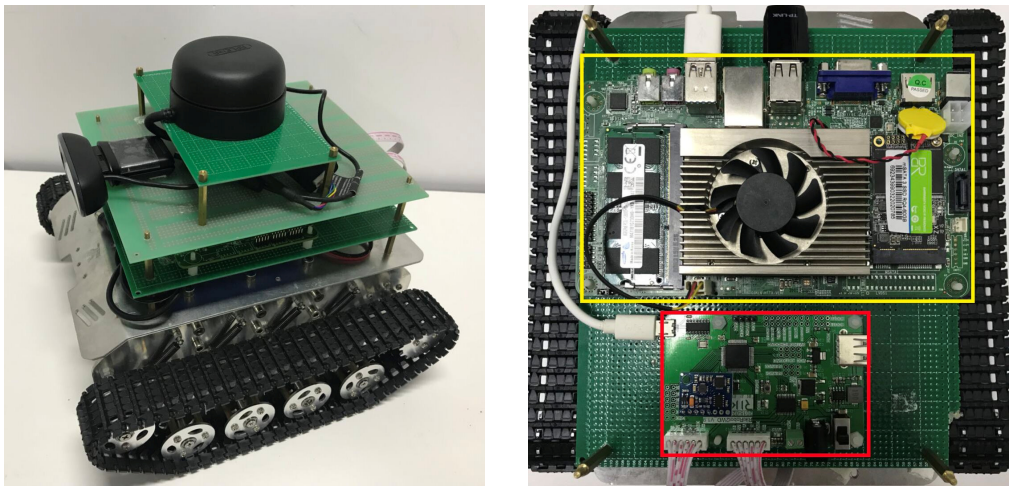
	$T_i(s)$	$T_o(s)$
RRT*	2.71	2.33
DOB-RRT*	0.53	0.38

From the simulation results, we can see that a shorter time is required by the proposed DOB-RRT* to obtain the initial path compared to the conventional RRT* scheme and it can converge to the asymptotically optimal path faster. As the RRT* online optimization calculation time is longer, the robot has already executed some of the costlier paths. DOB-RRT*'s online optimization solution can optimize the tree structure using the execution plan time and improve the path quality, including the path length and smoothness. The optimization of the proposed DOB-RRT* scheme allows it to find the path faster than the conventional RRT* scheme by 400 ms. This search time reduction in the proposed DOB-RRT* scheme proves that it can be effective in online optimization and is suitable for unmanned driving systems.

6. DOB-RRT* Evaluations in Real Environment

6.1. Experimental Environment Configuration

To prove the validity of the proposed path-tracking algorithm in a real environment, a real experiment is performed. The tracked robot used in this experiment is shown in Figure 9a. The vehicle size is 270 mm × 222 mm with a 229 mm wheelbase. An industrial computer with an i7-7500U processor (the yellow lined area in Figure 9b) and running the Kinetic robot operating system (ROS) is used in these experiments. The mapping package supported by the ROS is used to build the environment map. The adaptive Monte Carlo Positioning (AMCL) package is used for robot self-positioning and updates the real-time position. The STM32F103 (the red-lined area in Figure 9b) is used to communicate with the ROS software and control the robot motor. In this experiment, the maximum linear velocity of the robot is set to 0.5 m/s, with 0.2 m/s^2 maximum acceleration. When the robot is less than 5 cm from the goal state, we consider the robot to have reached the goal state.



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

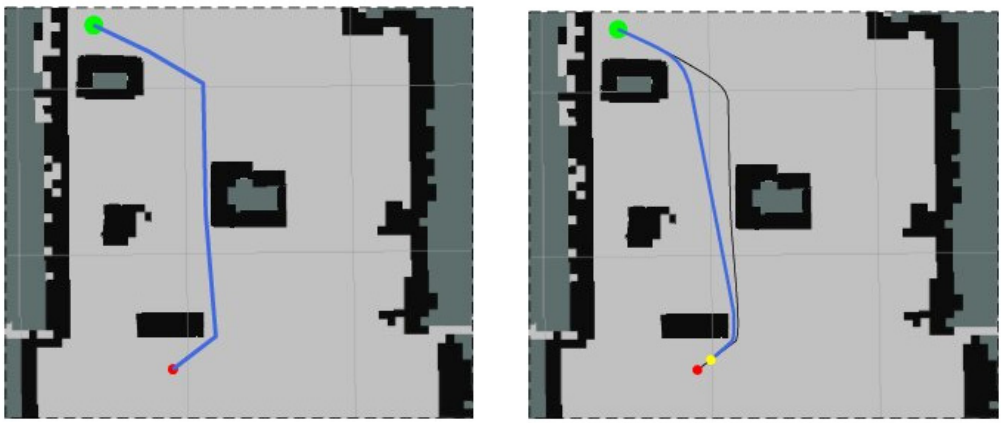
Figure 9. The mobile robot used in the real experiment; (**a**) the tracked robot used in this experiment; (**b**) the robot operating system (ROS).

6.2. Mobile Robot Applications

To further verify the effectiveness of the algorithm in the actual environment, we conduct indoor experiments. The experiment is carried out using a crawler robot, as shown in Figure 9. We implement our motion planning algorithm in the ROS environment. Motion planning algorithms can also be transplanted to other robots running ROS. The experimental environment is shown in Figure 10. As illustrated in Figure 10a, in the beginning, and based on the surrounding environment, using a laser point, the robot can generate a laser point cloud map. This cloud map is stored as its recognized map. The initial path generated by the first layer of RRT* is shown in Figure 11a, and the optimized path obtained by the second layer of RRT* is shown in Figure 11b. The time required for the robot to complete the navigation is 3.26 s, and the values of the linear velocity and angular velocity during navigation are shown in Figure 12. The detailed data of the operation are shown in Table 4. Based on these detailed data collected from the real experiment, the robot can quickly find the initial path based on the proposed DOB-RRT* algorithm and use the online optimization plan to improve the path quality during the execution of the plan. From the experimental results, we can see the strong similarity between the data obtained from the real experiment and the data obtained from the simulation results.

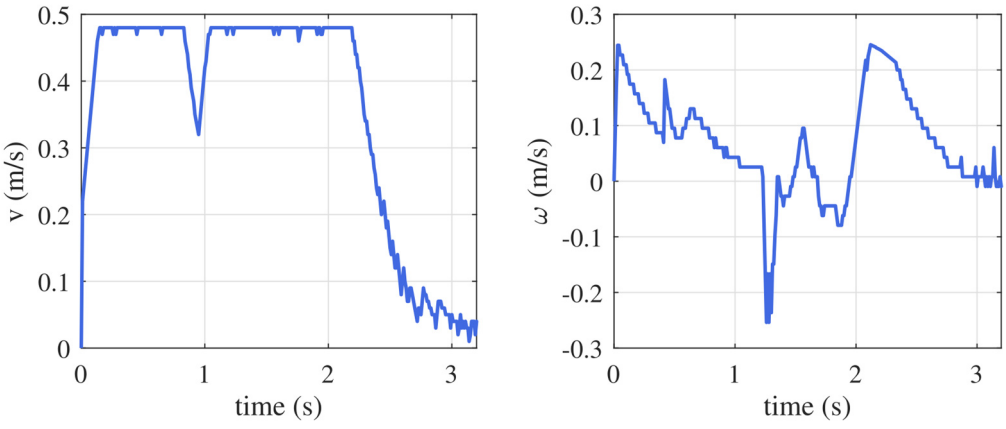
(**a**) (**b**)

Figure 10. Experimental environments: (**a**) the real surrounding environment; (**b**) the stored recognized map.



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

Figure 11. Trajectories of the tracked robot: the mobile robot started from the green point and targeted the red point. (**a**) The initial path generated by the first layer of RRT*; (**b**) the path generated by the second RRT* layer.



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

Figure 12. The values of linear velocity and angular velocity during robot navigation (velocity profiles): (**a**) linear velocity; (**b**) angular velocity.

Table 4. Summary of the mobile robot real experiment.

Plan Time (s)	Path Length (m)	$T_i(s)$	$T_o(s)$
0.34	2.57	2.24	3.26

7. Conclusions

This paper studies the problem of motion planning in an anytime motion scenario, focusing on the online solving and real-time optimization of motion planning problems. To achieve this goal, a double-layer RRT* algorithm is proposed. The proposed DOB-RRT* algorithm uses one tree to explore the initial path and another tree to optimize and update the selected path. The performance of the proposed DOB-RRT* algorithm is evaluated in a simulation and real experiments. The simulation results show that the proposed DOB-RRT* algorithm can reduce the calculation cost of the planned path. The proposed DOB-RRT* is superior to other reference algorithms in terms of its stability, path length, and smoothness. Through practical applications, we verify the effectiveness of the DOB-RRT* algorithm in real environments. In future research, the construction of a motion planning algorithm for multi-robot collaborative work will be conducted.

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

Funding: This research received no external funding.

Data Availability Statement: The authors declare that the data used to support the findings of this study will be available from the corresponding author upon request.

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

References

1. Latombe, J.-C. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *Int. J. Robot. Res.* **1999**, *18*, 1119–1128. [CrossRef]
2. Gao, H.; Hou, X.; Xu, J.; Guan, B. Quad-Rotor Unmanned Aerial Vehicle Path Planning Based on the Target Bias Extension and Dynamic Step Size RRT* Algorithm. *World Electr. Veh. J.* **2024**, *15*, 29. [CrossRef]
3. Liu, Y.; Badler, N.I. Real-time reach planning for animated characters using hardware acceleration. In Proceedings of the 11th IEEE International Workshop on Program Comprehension, New Brunswick, NJ, USA, 8–9 May 2003; pp. 86–93.
4. Thompson, B.; Yoon, H.-S. Efficient path planning algorithm for additive manufacturing systems. *IEEE Trans. Compon. Packag. Manuf. Technol.* **2014**, *4*, 1555–1563. [CrossRef]
5. Chen, Y.; Xu, W.; Li, Z.; Song, S.; Lim, C.M.; Wang, Y.; Ren, H. Safety-enhanced motion planning for flexible surgical manipulator using neural dynamics. *IEEE Trans. Control Syst. Technol.* **2016**, *25*, 1711–1723. [CrossRef]
6. LaValle, S. Planning Algorithms. *Camb. Univ. Press Google Sch.* **2006**, *2*, 3671–3678.
7. Huang, Y.; Lee, H.-H. Adaptive Informed RRT*: Asymptotically Optimal Path Planning With Elliptical Sampling Pools in Narrow Passages. *Int. J. Control Autom. Syst.* **2024**, *22*, 241–251. [CrossRef]
8. Huang, Y.; Tsao, C.-T.; Lee, H.-H. Efficiency Improvement to Neural-Network-Driven Optimal Path Planning via Region and Guideline Prediction. *IEEE Robot. Autom. Lett.* **2024**, *9*, 1851–1858. [CrossRef]
9. LaValle, S. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811* **1998**.
10. Hsu, D.; Kindel, R.; Latombe, J.-C.; Rock, S. Randomized kinodynamic motion planning with moving obstacles. *Int. J. Robot. Res.* **2002**, *21*, 233–255. [CrossRef]
11. LaValle, S.M.; Kuffner, J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
12. Cheng, P. *Sampling-Based Motion Planning with Differential Constraints*; University of Illinois at Urbana-Champaign: Champaign, IL, USA, 2005.
13. Iehl, R.; Cortés, J.; Simeon, T. Costmap planning in high dimensional configuration spaces. In Proceedings of the 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Kaohsiung, Taiwan, 11–14 July 2012; pp. 166–172.
14. Shkolnik, A.; Walter, M.; Tedrake, R. Reachability-guided sampling for planning under differential constraints. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 2–17 May 2009; pp. 2859–2865.

15. Yang, K.; Jung, D.; Sukkarieh, S. Continuous curvature path-smoothing algorithm using cubic B zier spiral curves for non-holonomic robots. *Adv. Robot.* **2013**, *27*, 247–258. [CrossRef]
16. Yang, K.; Sukkarieh, S. An analytical continuous-curvature path-smoothing algorithm. *IEEE Trans. Robot.* **2010**, *26*, 561–568. [CrossRef]
17. Elbanhawi, M.; Simic, M.; Jazar, R. Randomized bidirectional B-spline parameterization motion planning. *IEEE Trans. Intell. Transp. Syst.* **2015**, *17*, 406–419. [CrossRef]
18. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
19. Karaman, S.; Walter, M.R.; Perez, A.; Frazzoli, E.; Teller, S. Anytime motion planning using the RRT. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1478–1483.
20. 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]
21. Salzman, O.; Halperin, D. Asymptotically near-optimal RRT for fast, high-quality motion planning. *IEEE Trans. Robot.* **2016**, *32*, 473–483. [CrossRef]
22. Otte, M.; Frazzoli, E. RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *Int. J. Robot. Res.* **2016**, *35*, 797–822. [CrossRef]
23. Chen, L.; Shan, Y.; Tian, W.; Li, B.; Cao, D. A fast and efficient double-tree RRT*-like sampling-based planner applying on mobile robotic systems. *IEEE/ASME Trans. Mechatron.* **2018**, *23*, 2568–2578. [CrossRef]
24. Merat, F. Introduction to robotics: Mechanics and control. *IEEE J. Robot. Autom.* **1987**, *3*, 166. [CrossRef]
25. Geraerts, R.; Overmars, M.H. Creating high-quality paths for motion planning. *Int. J. Robot. Res.* **2007**, *26*, 845–863. [CrossRef]