electronics

MDPI

Article

Unmanned Aerial Vehicle Path-Planning Method Based on Improved P-RRT* Algorithm

Xing Xu [1], Feifan Zhang [2] and Yun Zhao [1,*]

[1] School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China; xuxing@zust.edu.cn
[2] School of Mechanical and Energy Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China; 222101855079@zust.edu.cn
* Correspondence: yunzhao@zust.edu.cn

Abstract: This paper proposed an improved potential rapidly exploring random tree star (P-RRT*) algorithm for unmanned aerial vehicles (UAV). The algorithm has faster expansion and convergence speeds and better path quality. Path planning is an important part of the UAV control system. Rapidly exploring random tree (RRT) is a path-planning algorithm that is widely used, including in UAV, and its altered body, P-RRT*, is an asymptotic optimal algorithm with bias sampling. The algorithm converges slowly and has a large random sampling area. To overcome the above drawbacks, we made the following improvements. First, the algorithm used the direction of the artificial potential field (APF) to determine whether to perform greedy expansion, increasing the search efficiency. Second, as the random tree obtained the initial path and updated the path cost, the algorithm rejected high-cost nodes and sampling points based on the heuristic cost and current path cost to speed up the convergence rate. Then, the random tree was pruned to remove the redundant nodes in the path. The simulation results demonstrated that the proposed algorithm could significantly decrease the path cost and inflection points, speed up initial path obtaining and convergence, and is suitable for the path planning of UAVs.

Keywords: path planning; RRT*; artificial potential field; greedy strategy; high-cost rejection

check for updates

Citation: Xu, X.; Zhang, F.; Zhao, Y. Unmanned Aerial Vehicle Path-Planning Method Based on Improved P-RRT* Algorithm. *Electronics* **2023**, *12*, 4576. https://doi.org/10.3390/electronics12224576

Academic Editor: Sergio Garcia-Nieto

Received: 19 September 2023
Revised: 5 November 2023
Accepted: 6 November 2023
Published: 9 November 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Unmanned aerial vehicles (UAVs) are ideal for carrying out tasks with relatively high human costs in indoor or outdoor environments because of their mobility, flexibility, rapid deployment, and wide applicability [1,2]. In recent years, with the development of technology and growth in demand, UAVs have played a significant role in agriculture [3], logistics and transportation [4,5], security monitoring [6], network communication [7], and rescue and emergency response [8]. Today, people recognize the importance of UAVs to human life and to production. UAVs must obtain more control and planning capacity due to changing environments. The path-planning problem is a significant issue that UAVs must resolve in practical applications and is one of the current research hotspots [9]. The problem refers to the requirement for UAVs to obtain a path from the starting point to the target point without coming into contact with any obstacles [10].

Research in path planning generally focuses on "path optimality" and "probabilistic completeness". Path optimality is the ability of an algorithm to find the path with the lowest cost, and probabilistic completeness is the ability of an algorithm to find a path that is feasible. It is difficult for path-planning algorithms to satisfy both fast planning and optimal planning. Early path-planning methods typically transformed the original map into a grid map and then searched for the target location. The most representative methods were the Dijkstra algorithm [11,12] and the A* algorithm [13,14]. Both could find the optimal path but would occupy a lot of memory and time. The artificial potential field method (APF) assumed the existence of an attractive potential of the target point

and the repulsive potential of the obstacle in the map. The path could be determined by the change in the gradient of the potential field [15]. However, there might be regions in the map where the potential field is in equilibrium, causing the UAVs to fall into local minima [16]. The ant colony algorithm (ACO) mimics the natural phenomenon in which ants leave pheromones on the path to find the shortest path, but the algorithm converges slowly when dealing with complex problems, and it also falls into local minima [17,18]. Genetic algorithms (GA) encode each path and, finally, form feasible paths after crossover, mutation, and selection [19]. However, the complex adaptation functions and coding make the algorithm memory- and time-intensive [20].

In recent years, sampling-based algorithms have gradually attracted the attention of researchers, and the most influential ones include the Probabilistic RoadMap (PRM) [21] and rapidly exploring random tree (RRT) [22]. The main idea of the PRM algorithm is to sample randomly in space and obtain a collision-free set of nodes. Then, the PRM is connected to the start and target points to obtain the paths. Furthermore, with a sufficiently large number of samples, the PRM is complete [23]. The RRT algorithm is a single-query planner in which the random tree connects nodes and grows from the starting point after randomly sampling the nodes obtained. The random tree gradually grows around the target point and, finally, finds a path connecting the starting point and the target point. Like the PRM, the RRT is complete for a sufficiently large number of samples [24]. The biggest advantage of the sampling-based algorithm is that it can find the path quickly and can meet the requirements of UAVs for fast maneuvers.

Despite the probabilistic completeness of both algorithms, the quality of the initial paths is usually not excellent, and the optimal path cannot be obtained. To address this issue, Karaman and Frazzoli proposed PRM* and RRT* [25] and demonstrated their asymptotic optimality. The new node tries to update the parent node continuously to obtain a smaller path cost [26]. RRT*-Smart [27] uses a bias sampling strategy for fast optimization of the obtained initial solution but the possibility of obtaining the globally optimal path is lost [28]. Bidirectional-RRT* (Bi-RRT*) [29] proposed a new algorithm for joint pathfinding by two random trees, i.e., each explores from the starting point and the target point towards the other to improve efficiency [30,31]. Informed-RRT* [32] sped up convergence by restricting sampling points to less costly elliptical regions. Quick-RRT* (Q-RRT*) [33] expanded the range of reselected parent nodes and used pruning means to reduce the path cost. The algorithm can be effectively integrated with other improved algorithms [34].

Potential-RRT* (P-RRT*) [35] successfully combines APF and RRT* to accelerate the exploration efficiency of RRT*. It also uses the random sampling property of RRT* to avoid falling into local minima, which is common in APF algorithms. The P-RRT* algorithm assumes that the sampling is affected by the target point, and the new nodes expanding from the random tree are then gradually biased towards the target point. Compared with the RRT* algorithm, the P-RRT* is more suitable for the path planning of UAVs. P-RRT* can also be combined with other algorithms such as potentially guided bidirectionalized RRT* (PB-RRT*) [30] and PF-RRT* [36]. The former makes two random trees attract each other through potential guidance. The latter combines with F-RRT* [37] to reduce the path cost by inserting the parent nodes. However, this algorithm still has the inherent drawback of the RRT* algorithm, i.e., a large amount of sampling and expansion occurs in high-cost regions, which is not useful for obtaining optimal paths.

UAVs need a responsive, high-quality, and smooth path-planning algorithm. This paper presents an improved UAV path-planning algorithm based on P-RRT* that can be applied to various environments. First, considering the need for rapid planning for UAVs, the algorithm contains a greedy strategy. The algorithm guides the greedy expansion with the direction of the artificial potential field on the node as the optimal direction, which can explore the depth of the map faster. Second, by calculating the heuristic costs of nodes and sampling points, comparing them to the existing path costs, and rejecting nodes and sampling points in the high-cost region, the algorithm increases the search efficiency of each iteration. Then, the random tree is pruned, and the parents of the nodes are reselected

to reduce unnecessary redundant nodes on the path. Finally, compared with the simulation results of the traditional RRT* algorithm and P-RRT* algorithm, the algorithm proposed in this paper has significant superiority in sparse, complex, as well as in realistic application scenarios.

The rest of this paper is presented as follows. Section 2 explains the necessary definitions, equations, etc., and introduces related algorithms. Section 3 describes in detail the principles and algorithmic contents of the proposed algorithms. Section 4 simulates each algorithm in different environments and compares and analyzes the simulation results, and Sections 5 and 6 present the discussion and conclusion of this work. The relevant algorithms' code has been released at: https://github.com/ZFF20231101/RRT-algorithms (accessed on 1 November 2023).

2. Background

This section defines path-planning problems and the notation used to represent them. It also describes the pseudocode and content of the RRT* and P-RRT* algorithms for the purpose of understanding the improvements proposed in Section 3.

2.1. Problem Definition

Let χ be the configuration space in n dimensions; $\chi_{obs} \subset \chi$ denotes the space occupied by obstacles and $\chi_{free} = \chi / \chi_{obs}$ denotes the free space reachable by the UAVs in the space. In the space, $x_{init} \in \chi_{free}$ is the initial state of path planning and $X_{goal} \subset \chi_{free}$ is the target region. In this paper, the target region is set as a target point x_{goal}. If the connections of two nodes $x_1, x_2 \in \chi$ are collision-free, they can be a path σ and calculate the Euclidean distance as the path cost c.

Problem 1. *Feasible Path Planning*: *For a given configuration space* $(\chi, \chi_{obs}, \chi_{free})$, *find a collision-free feasible path* $\sigma : [0,1] \epsilon \chi_{free}$ *with* $\sigma(0) = x_{init}$ *and* $\sigma(1) = x_{goal}$.

Problem 2. *Optimal Path Planning*: *For a given configuration space* $(\chi, \chi_{obs}, \chi_{free})$, *find an optimal path* σ^* *among all feasible paths and its path cost is* $c(\sigma^*) = \min\left\{c(\sigma) : \sigma \in \chi_{free}\right\}$.

*2.2. RRT**

This section focuses on RRT* [25], which is a sampling-based path-planning algorithm. Form a vertex set$V$ from each node, plus an edge set$E$ to form a random tree$T$. After the initialization of the algorithm, a random sampling point $x_{rand} \in \chi_{free}$ in the space is provided. The node $x_{nearest}$ closest to x_{rand} is taken as the parent node and extends a fixed extension length towards x_{rand} to obtain a new node, x_{new}. The *CollisionFree* procedure verifies that the edge formed by joining x_{new} and $x_{nearest}$ has not passed through χ_{obs}, then x_{new} can be added to $T = (V, E)$ as part of the random tree. The above steps are also the whole process of the RRT algorithm. The verified x_{new} can obtain a set of nearby nodes X_{near}, which is included in a circular region, where X_{near} satisfies the following relation

$$X_{near} = \{x \in T : d(x, x_{new}) \leq r := \alpha \sqrt{\frac{\log_n}{n}}\} \lim_{x \to \infty} \quad (1)$$

where α is an independent coefficient. In the *ChooseParent* procedure, x_{new} tries to connect to $x_{near} \in X_{near}$ and change the parent node to obtain a smaller path cost. Similar to *ChooseParent*, every $x_{near} \in X_{near}$ can try to treat x_{new} as the parent node and minimize the cost in the *Rewire* procedure. The procedure lowers the cost of nearby nodes. The pseudocode of the algorithm is given in Algorithm 1. The *ChooseParent* and *Rewire* procedures enable the random tree to gradually approach the optimal solution; the asymptotic optimality of the algorithm was proven in the paper [25].

Algorithm 1: RRT* (x_{init})

$V \leftarrow \{x_{init}\}$; $E \leftarrow \varnothing$;
$T \leftarrow (V, E)$;
for $i = 0$ **to** $maxIter$ **do**
  $x_{rand} \leftarrow Sample(rate)$;
  $x_{nearest} \leftarrow Nearest(T, x_{rand})$;
  $x_{new} \leftarrow Steer(x_{rand}, x_{nearest})$;
  if $CollisionFree(x_{new}, x_{nearest})$ **then**
    $X_{near} \leftarrow NearNodes(T, x_{new})$;
    $x_{newparent} \leftarrow ChooseParent(X_{near}, x_{new})$;
    $x_{naerparent} \leftarrow Rewire(X_{near}, x_{new})$;
    $T \leftarrow (x_{new}, x_{newparent}, x_{naerparent})$;
  end if
end for
return T;

2.3. P-RRT*

The artificial potential field method (APF) is a gradient descent planning method. For a UAV denoted as $x \in \chi$, it is assumed to be subjected to the joint action of the virtual attraction potential U_{att} of the target point x_{goal} and the repulsion potential U_{rep} of each obstacle. There is a gradient change in the entire configuration space χ, such that x moves toward the target point in the direction of the fastest decreasing gradient. The entire process can be seen as x being subjected to the combined force of the attractive force $\vec{F}_{att}$ and the repulsive force $\vec{F}_{rep}$. The formulas for the attractive and repulsive forces are given in Equations (2)–(5).

$$U_{att} = \begin{cases} \frac{1}{2}K_a\left|\left|x - x_{goal}\right|\right|^2 & \left|\left|x - x_{goal}\right|\right| > d_g \\ \frac{1}{2}K_a(d_g\left|\left|x - x_{goal}\right|\right| - d_g^2) & \left|\left|x - x_{goal}\right|\right| \leq d_g \end{cases} \tag{2}$$

$$\vec{F}_{att} = \begin{cases} -K_a\left|\left|x - x_{goal}\right|\right| & \left|\left|x - x_{goal}\right|\right| > d_g \\ -K_a d_g \frac{||x - x_{goal}||}{d(x, x_{goal})} & \left|\left|x - x_{goal}\right|\right| \leq d_g \end{cases} \tag{3}$$

$$U_{rep} = \begin{cases} \frac{1}{2}K_r(\frac{1}{d_{\min}} - \frac{1}{d_{obs}})^2 & d_{\min} \leq d_{obs} \\ 0 & d_{\min} > d_{obs} \end{cases} \tag{4}$$

$$\vec{F}_{rep} = \begin{cases} K_r(\frac{1}{d_{obs}} - \frac{1}{d_{\min}})\frac{1}{d_{\min}^2}\frac{\partial d_{\min}}{\partial x} & d_{\min} \leq d_{obs} \\ 0 & d_{\min} > d_{obs} \end{cases} \tag{5}$$

where K_a and K_r are coefficients that can be adjusted depending on the specific situation to control the strength of $\vec{F}_{att}$ and $\vec{F}_{rep}$, respectively. The d_g and d_{obs} indicate that different calculations of $\vec{F}_{att}$ and $\vec{F}_{rep}$ are applied at different action distances. If the distance from x to the target point x_{goal} is greater than d_g, $\vec{F}_{att}$ increases quadratically with distance. The UAV is subject to greater attraction at longer distances and thus rapidly approaches the target point. The attractive potential gradient becomes flatter when the two are too close together. $\vec{F}_{rep}$ is calculated as the opposite of $\vec{F}_{att}$. For an obstacle $x_{obs} \in \chi_{obs}$, the minimum distance between x and x_{obs} is d_{min}, and the repulsive force is 0 for $d_{min} > d_{obs}$, indicating that the obstacle at a long distance did not produce a repulsive force. When two objects are very close to one another, $\vec{F}_{rep}$ increases quickly with decreasing distance, allowing x to move away from the area.

The P-RRT* algorithm [35] adds the RGD procedure to the original RRT* algorithm, as shown in Algorithm 2. The random tree is influenced by the attractive force of the target

point in the configuration space, causing the tree as a whole to be more inclined to the target point. The process is explained in detail in Algorithm 3. For the newly generated sampling point x_{rand}, it slowly approaches x_{goal} due to the attraction potential field until it is too close to the obstacle. P-RRT* calculates $\overrightarrow{F}_{att}$ using only the first case in Equation (3), because in APF, it is necessary to avoid x crossing x_{goal} at smaller distances, while in RRT*, the connection between the random tree and x_{goal} at close distances can be determined directly. The random tree is then expanded towards x_{prand}. *RGD* introduces three parameters, k, d_{obs}, and λ. The k is the maximum times of deflections of x_{rand} toward x_{goal}, λ is the length of each deflection of x_{rand}, and d_{obs} is the closest allowable distance between x_{rand} and the obstacle.

Algorithm 2: P-RRT* (x_{init})

$V \leftarrow \{x_{init}\}$; $E \leftarrow \varnothing$;
$T \leftarrow (V, E)$;
for $i = 0$ **to** $maxIter$ **do**
  $x_{rand} \leftarrow Sample(rate)$;
  $x_{prand} \leftarrow RGD(x_{rand})$
  $x_{nearest} \leftarrow Nearest\left(T, x_{prand}\right)$;
  $x_{new} \leftarrow Steer\left(x_{prand}, x_{nearest}\right)$;
  if $CollisionFree(x_{new}, x_{nearest})$ **then**
    $X_{near} \leftarrow NearNodes(T, x_{new})$;
    $x_{newparent} \leftarrow ChooseParent(X_{near}, x_{new})$;
    $x_{naerparent} \leftarrow Rewire(X_{near}, x_{new})$;
    $T \leftarrow \left(x_{new}, x_{newparent}, x_{naerparent}\right)$;
  end if
end for
return T;

Algorithm 3: $RGD(x_{rand})$

$x_{prand} \leftarrow x_{rand}$;
for $i \leftarrow 0$ **to** k **do**
  $\overrightarrow{F}_{att} \leftarrow APG\left(x_{Goal}, x_{prand}\right)$;
  $d_{min} \leftarrow NearestObs\left(X_{obs}, x_{prand}\right)$;
  if $d_{min} \leq d_{obs}$ **then**
    return x_{prand};
  else
    $x_{prand} \leftarrow x_{prand} + \lambda\left(\overrightarrow{F}_{att}/|\overrightarrow{F}_{att}|\right)$;
end for
return x_{prand};

3. Improved P-RRT*

The increased target point attraction of the P-RRT* algorithm aids in speeding up the convergence, but due to the property of the algorithm, the random tree expands with low exploration efficiency, with only one new node per iteration. In addition, the configuration space contains a significant number of high-cost regions, such as edges of the map and regions far from the target points. The random sampling property of the algorithm makes it possible for the random tree to explore high-cost regions, but nodes located in such regions cannot be part of the optimal path. The algorithm should reduce the path nodes to smooth the flight path of UAVs. To address these problems, the following improvements were made in this paper.

- Greedy strategy: to address the problem of the slow exploration of random trees, a greedy strategy is incorporated to speed up the expansion. The greedy expansion is carried out if the APF direction of the node is within a certain deflection angle from the actual expansion direction;
- High-cost rejection: the heuristic cost of sampling points and nodes is calculated; if it exceeds the current optimal path cost c_{min}, the points and nodes are in the high-cost region. The algorithm rejects high-cost sampling points and nodes, resamples, and expands only on low-cost nodes;
- Path optimization: the random tree is pruned in order to remove redundant nodes from the path. This not only lowers the cost of the path but also enables the UAVs to pass through fewer path turning points.

The proposed algorithm based on the above ideas is shown in Algorithm 4, and each improvement point is described in detail in the Sections 3.1–3.3.

Algorithm 4: Improved P-RRT* (x_{init})

$V \leftarrow \{x_{init}\}$; $E \leftarrow \varnothing$;
$T \leftarrow (V, E)$; $c_{min} \leftarrow \infty$;
for $i = 0$ **to** $maxIter$ **do**
  $x_{rand} \leftarrow NewSample(c_{min}, rate)$;
  $x_{nearest} \leftarrow NewNearestPossible(T, x_{rand}, c_{min})$;
  while $CheckAngle(x_{rand}, x_{nearest}, x_{new} \leftarrow \varnothing)$ **do**
    $x_{new} \leftarrow Steer(x_{rand}, x_{nearest})$;
    $x_{new} \leftarrow NewRGD(x_{new})$;
    if $CollisionFree(x_{new}, x_{nearest})$ **then**
      $X_{near} \leftarrow NearNodes(T, x_{new})$;
      $x_{newparent} \leftarrow NewChooseParent(X_{near}, x_{new})$;
      $X_{naerparent} \leftarrow NewRewire(X_{near}, x_{new})$;
      $T \leftarrow (x_{new}, x_{newparent}, X_{naerparent})$;
    end if
  end while
end for
return T;

3.1. Greedy Strategy

The greedy expansion method can be applied to the RRT* algorithm to expedite the exploration. The random tree keeps expanding in the x_{rand} direction using this method until it runs into an obstacle, producing multiple x_{new} in the process. This method has an obvious drawback in that the greedy expansion direction is frequently undesirable and sometimes even far from the target point. The target greedy expansion method [38] was improved by only allowing greedy expansion to occur when *xrand* = *xgoal*, thereby minimizing greedy expansion in the wrong direction. The disadvantage of this method is that after obtaining the initial solution, it is not possible to perform greedy expansion. Because the nearest node $x_{nearest}$ of x_{goal} must be the last node of the initial path, the random tree cannot be expanded. In addition, greedy expansion toward the target point may trap the path in an area surrounded by numerous obstacles.

To solve the above problem, it was necessary to make the random tree expand greedily in the optimal direction and normally in the other directions. In this paper, we proposed a greedy expansion algorithm based on APF guidance. The proposed algorithm subjects the nodes to both attractive and repulsive potentials, and the expansion also takes into account the x_{rand} as an attractive source. The node is subjected to $\vec{F}_{rep}$ and $\vec{F}_{att1}$, as shown in Figure 1, and the APF force $\vec{F}_{p}$ (red line) is produced under the combined action of both. The node is simultaneously affected by the attraction $\vec{F}_{att2}$ of the sampled point x_{rand}, which results in the combined force $\vec{F}_{total}$ (green line) as the direction of expansion,

ultimately. Note that there are generally multiple obstacles in the configuration space, and that the repulsive force generated by each obstacle in the range of d_{obs} should be calculated to form the combined force $\sum \vec{F}_{rep}$. The expression of the combined force on the node is shown in Equation (6). The improved *NewRGD* procedure is shown in Algorithm 5.

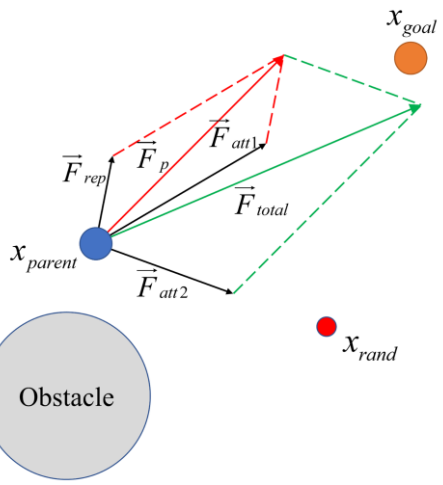

Figure 1. Change in expansion direction of the node.

$$\vec{F}_{\text{total}} = \sum \vec{F}_{att} + \sum \vec{F}_{rep} \tag{6}$$

Algorithm 5: $NewRGD(x_{new})$

$\vec{F}_{att1} \leftarrow APG(x_{Goal}, x_{new})$;
for each $x_{obs} \in \chi_{obs}$ **do**
  $d_{min} \leftarrow Distance(x_{obs}, x_{new})$;
  if $d_{min} \leq d_{obs}$ **then**
    $\vec{F}_{repi} \leftarrow RPG(d_{min})$;
    $\sum \vec{F}_{rep} \leftarrow \sum \vec{F}_{rep} + \vec{F}_{repi}$;
  end if
  $\vec{F} \leftarrow \vec{F}_{att1} + \sum \vec{F}_{rep}$;
end for
return $\vec{F}$;

The direction of the potential field force $\vec{F}_p$ obtained by the APF algorithm represents the optimal direction of the expansion in the local region. The proposed algorithm, combined with the greedy strategy, adds the *CheckAngle* procedure, which can determine whether the angle θ between the $\vec{F}_p$ and the expansion direction satisfies the greedy expansion condition. As shown in Figure 2, the starting point is expanded to obtain x_{new1}. At this time the node is only subjected to the attraction $\vec{F}_{att1}$ of x_{goal}, and the angle θ_1 is below the algorithmic threshold, which means that this expansion is toward the optimal direction and greedy expansion can be carried out. The newly generated x_{new2}, x_{new3} and x_{new4} all satisfy the condition, and the random tree is expanded to x_{new5}. The obstacle repulsive force $\vec{F}_{rep}$ has a significant effect on x_{new5}, and the greedy expansion stops when the pinch angle θ_5 is out of range. As shown in the figure, the random tree is greedily expanded to reach the favorable position quickly along the optimal direction, which greatly reduces the time required for exploration. It is worth noting that the APF direction only represents the optimal direction for a certain local region. Sometimes, random trees are extended to local minima regions, while the random expansion of the RRT algorithm can help the random tree leave these regions.

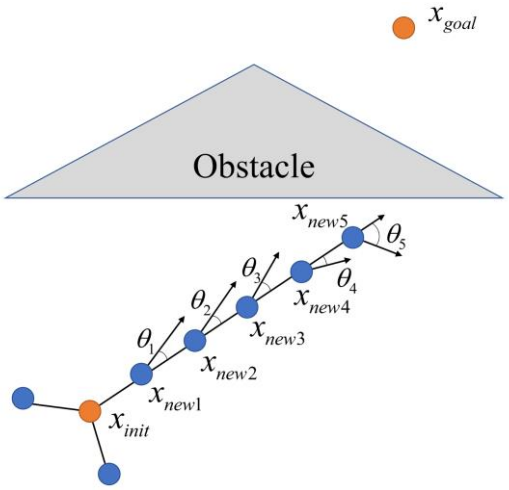

Figure 2. Fast expansion with greedy strategy.

3.2. High-Cost Rejection

As the algorithm obtained the initial solution, we divided the configuration space into high-cost and low-cost regions. The division was based on the current shortest path cost c_{min} and the heuristic cost f-value of the region. The cost g-value of the path length between x and x_{init} and the cost h-value of the linear distance between x and x_{goal} were calculated based on the method in [13]. From this, the heuristic function f-value was calculated as follows:

$$f(x) = g(x) + h(x) \tag{7}$$

where $f(x)$ was the theoretical minimum cost of the path through the point. If the $f(x)$ of a node or sampling point was greater than c_{min}, it was considered to be in the high-cost region.

According to the original RRT* and the P-RRT* algorithm, the randomness of sampling ensures that, regardless of the cost of the current optimal path, the sampling region is located throughout the configuration space. However, sampling points falling in high-cost regions make it difficult to accelerate convergence. The improved algorithm proposed in this paper generates x_{rand} randomly in χ_{free}, but valid sampling must satisfy $f(x_{rand}) < c_{min}$, where $f(x_{rand}) = g(x_{nearest}) + d(x_{nearest}, x_{rand}) + h(x_{rand})$. As shown in Figure 3a, after obtaining the initial path σ and the path cost c_{min}, the algorithm rejects the newly generated sampling points x_{rand1} and x_{rand2} successively, because $f(x_{rand}) > c_{min}$. By resampling to obtain x_{rand3}, as opposed to the original sampling point, it is more likely to obtain a better path. The algorithm for the sampling part is shown in Algorithm 6.

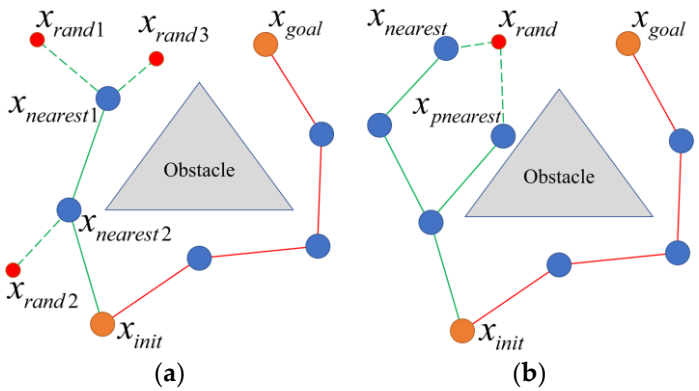

Figure 3. Reject high-cost sampling points and nodes: (**a**) reject sampling point; and (**b**) reject node.

Algorithm 6: *NewSample* $(c_{min}, rate)$

if $goalSampleRate(rate)$ **then**
  $x_{rand} \leftarrow x_{goal}$;
else
  while True
    $x_{rand} \leftarrow Random\left(\chi_{free}\right)$;
    $f(x_{rand}) \leftarrow x_{nearcost} + d(x_{rand}, x_{near}) + d\left(x_{rand}, x_{goal}\right)$;
    if $f(x_{rand}) < c_{min}$ **then**
      break;
    end if
  end while
end if
return x_{rand};

As the number of iterations of the RRT* algorithm increases and c_{min} became smaller and smaller, there were some nodes on the random tree with $f(x_{node}) > c_{min}$. It was impossible for these high-cost nodes to be part of the optimal path in subsequent iterations. Nodes were classified as possible nodes and impossible nodes based on $f(x_{node})$ and c_{min}:

$$X_{poss} = \{x : f(x) < c_{\min}\} \tag{8}$$

As shown in Figure 3b, the traditional algorithm took the nearest node $x_{nearest}$ as the parent node to extend the random tree to x_{rand}, but the path cost was too high for this to be the optimal path. The proposed algorithm rejects $x_{nearest}$ and used $x_{pnearest}$ as the parent node extension to avoid the path passing through the high-cost region. The pseudocode of the improved algorithm is shown in Algorithm 7. The process of traversing the nodes on the random tree by the algorithm sets the possibilities of the nodes, which do not make the algorithm more complex.

Algorithm 7: *NewNearestPossible* (T, x_{rand}, c_{min})

$X_{list} \leftarrow T$;
for each $x_{node} \in X_{list}$ **do**
  $D_{list} \leftarrow d(x_{node}, x_{rand})$;
  $f(x_{node}) \leftarrow x_{cost} + d\left(x_{node}, x_{goal}\right)$;
  $SetPossibility(f(x_{node}), c_{min})$;
end for
for $i = 0$ **to** N **do**
  $x_{nearest} \leftarrow \min(D_{list})$;
  if $CheckPossibility(x_{nearest})$ **then**
    break;
  else $D_{list}(x_{nearest}) \leftarrow \infty$;
end for
return $x_{nearest}$;

3.3. Path Optimization

The path generated by RRT* contained many redundant nodes, so the random tree can be pruned to turn the path from zigzag to straight, which reduces the path cost and smooths the UAVs' flight path. The RRT* algorithm only considers $x_{near} \in X_{near}$ as a potential parent node, while the proposed algorithm also considers other nodes along the path. The principle of the improved algorithm is shown in Figure 4. When the new node x_{new} is added to the random tree, the algorithm connects x_{new} with the most suitable node $x_{near} \in X_{near}$. Then, x_{new} can try to connect the parent node $x_{ancestor1}$ of the parent node $x_{parent} = x_{near}$ and determine whether it will collide with the obstacle. If the condition is met, x_{new} chooses $x_{ancestor1}$ as the parent node; the red dashed line reflects this process.

According to the triangle inequality, the path cost of x_{new} at this time must be smaller than in the previous case. The algorithm repeats this process until the condition is not satisfied when it is connected to $x_{ancestor3}$. Finally, the algorithm obtains a path with fewer nodes suitable for UAV flight. The pseudocode for this part is shown in Algorithm 8. *Rewire* is similar to the above process. The node $x_{near} \in X_{near}$ tries to connect with x_{new} and determine whether the path cost is smaller. If the condition is satisfied, x_{new} is set as x_{near}'s parent node. The above process can be repeated until the node cannot satisfy the condition.

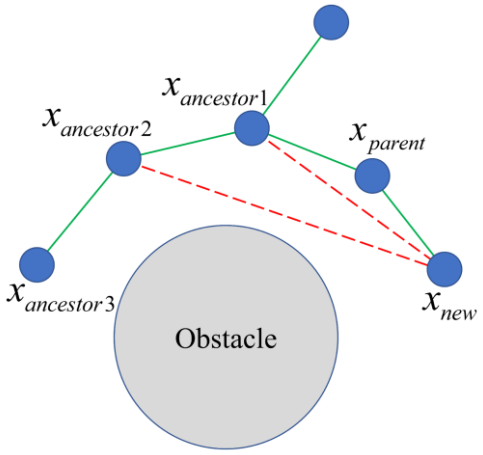

Figure 4. Reselect the parent node.

Algorithm 8: $NewChooseParent(X_{near}, x_{new})$

```
for each x_near ∈ X_near do
  if CollisionFree(x_new, x_near) then
    c_near ← Cost(x_near) + d(x_near, x_new);
    if c_near < c_min then
    x_parent ← x_near;
    c_min ← c_near;
    end if
  end if
end for
 while x_ancestor ≠ x_init do
   if CollisionFree(x_ancestor, x_new) then
     x_parent ← x_ancestor;
   else
     break
 end while
 return x_parent;
```

3.4. Algorithm Flow

The proposed algorithm is represented in Algorithm 4. First, the algorithm is initialized based on the information in the space to construct a random tree starting with x_{init}, and then the first sampling is performed in a loop. The initial sampling range is unrestricted. Once the initial path is obtained for the random tree, the algorithm rejects the high-cost region sampling points based on c_{min}. Then, all nodes are classified according to the f-value, and only nodes with $f(x_{node}) \leq c_{min}$ are set as possible nodes and connected to x_{new}. Due to the APF, each node expands x_{new} in the *Steer* procedure in the direction of the total combined force $\vec{F}_{total}$. After expansion in a certain direction, the proposed algorithm determines whether to start the greedy expansion based on the force acting on the node and the direction of the expansion. If the edge generated by joining x_{new} and x_{parent} passes the collision-free detection, x_{new} is added to the random tree. Next, x_{new} finds the parent

node with the lowest cost among the set of nearby nodes X_{near} and the ancestor nodes on the path, which removes a large number of redundant nodes. If the Euclidean distance between x_{new} and x_{goal} is less than the set value, the nodes can be connected, and the path is output in the algorithm. Otherwise, the next iteration is carried out. The loop ends after reaching the set number of iterations.

3.5. Algorithm Analysis

The probabilistic completeness and asymptotic optimality of RRT* and P-RRT* have been proven in their respective papers [25,35]. The proposed algorithm is identical to the sampling method of P-RRT* until the initial solution is obtained. The greedy strategy does not change the way new nodes are expanded by sampling. In addition, path optimization only changes the connection distribution of the random tree, and the nodes remain connected to the starting point. Hence the proposed algorithm is probabilistically complete, i.e., a feasible solution is obtained when the number of iterations tends to infinity. Asymptotic optimality requires that the algorithm obtains a solution with a minimum cost as the number of iterations tends to infinity. The nodes of the more optimal solution after the algorithm obtains the initial solution are necessarily distributed in the low-cost regions. The proposed algorithm restricts the sampling to the low-cost regions after obtaining the initial solution and does not miss the optimal solution. Moreover, the greedy strategy and path optimization do not affect the way the nodes are generated. Therefore, the proposed algorithm is also asymptotically optimal.

The computational complexity of the algorithms has been analyzed in the original papers proposing RRT* and P-RRT*. Time and space complexity is used to measure the time and space required by the algorithm. For two functions, $f(n)$ and $g(n)$, $f(n)$ is said to belong to $O(g(n))$ if

$$\lim_{n\to\infty} \mathbb{E}\left[\frac{f(n)}{g(n)}\right] < \infty \tag{9}$$

Let T_n^{PRO} be the amount of memory space occupied by the proposed algorithm after n iterations, it is obvious that $\left|T_n^{PRO}\right| = \left|V_n^{PRO}\right| + \left|E_n^{PRO}\right|$, where V_n^{PRO} and E_n^{PRO} are the number of nodes and edges.

Theorem 1. $\left|T_n^{PRO}\right| \in O(n)$.

Proof of Theorem 1. The greedy strategy increases the number of nodes and edges produced in each iteration. However, the boundary of the map limits the number of nodes added by the greedy strategy, $\left|V_n^{PRO}\right| = n + a$, where a is the total number of nodes added in n iterations. The number of edges is the same as the number of nodes, so that $\left|E_n^{PRO}\right| = n + a$ and $\left|T_n^{PRO}\right| = 2n + 2a$. Hence Theorem 1 can be proved from Equation (9). □

The time complexity needs to be calculated for the program that the algorithm executes the most frequently in the loop, which in this paper's algorithm is the *CollisionFree* program. Let M_n^{PRO} be the total time for the algorithm to call *CollisionFree* under n iterations, and C_n^{PRO} and R_n^{PRO} be the time for *NewChooseParent* and *NewRewire* to call *CollisionFree*, respectively, then $M_n^{PRO} = C_n^{PRO} + R_n^{PRO}$.

Theorem 2. $M_n^{PRO} \in O(nlogn)$.

Proof of Theorem 2. Let a total of k nodes $x_{near} \in X_{near}$ participate in the *NewChooseParent* process; it has been shown in [25] that obstacle collision detection needs to run in $k(\log n)$ time. The proposed algorithm adds a nodes and an optimization path step. In the worst case, x_{new} selects a parent node with a total of $n + a - k$ ancestor nodes, and all of them participate in the path optimization. Hence,$C_n^{PRO} \leq (1 + a)(k + n + a - k)\log n$,$C_n^{PRO} \in O(nlogn)$. The k x_{near} in *NewRewire* needs to perform the optimization path step for the $n + a - k$ ancestor nodes

of x_{new} in the worst case, so $R_n^{PRO} \leq (1+a)[k\cdot(n+a-k)]\log n$ and $R_n^{PRO} \in O(n\log n)$. Thus Theorem 2 can be proved. □

4. Simulation Results

In this section, the proposed algorithm with RRT* and P-RRT* was compared to verify the superiority. The simulation experiments were run on a computer with a 2.50 GHz processor and 16 GB RAM. To fully analyze the effectiveness of the proposed algorithm, four 2D maps and two 3D maps were used to simulate the various UAV scenes. Due to the randomness of the RRT algorithm, 100 simulations for each map were run to analyze the performances of different algorithms. The black graph on the map represents the obstacle. The starting point x_{init} and the target point x_{goal} are represented by the red x. Each black dot represents a node in the random tree. The path between two nodes is represented by a green line, and the red line is the optimal path at the current iteration.

The UAVs needed to find the initial solution quickly to complete the path planning and obtain a small cost path. Based on the above requirements, three evaluation indicators were used: c_{min} was the path cost obtained by the algorithm within the specified number of iterations, the smaller the number of iterations, the better the path quality; t_{init} was the time cost of the algorithm obtaining the initial path, the lower the cost the better for fast planning; t_{cost} was the time cost the of the algorithm finding an excellent path with a cost of $1.05c(\sigma^*)$, which measured the convergence speed of the algorithm, where $c(\sigma^*)$ was the cost of the optimal path. Failure meant that the algorithm could not find an excellent path within 10,000 iterations. In addition to the three evaluation indicators, the simulation data included the nodes and runtimes generated by the algorithms executing 500 or 1000 iterations in each environment.

4.1. 2D Environment

In this section, the 2D environments used for simulation testing included a sparse environment Map A (Figure 5), a cluttered environment Map B (Figure 6), a simple maze environment Map C (Figure 7), and a complex maze environment Map D (Figure 8). The size of each map is 100 × 100. Table 1 shows the average data of 100 simulations for each algorithm in the four maps, and the data visualization is shown in Figure 9.

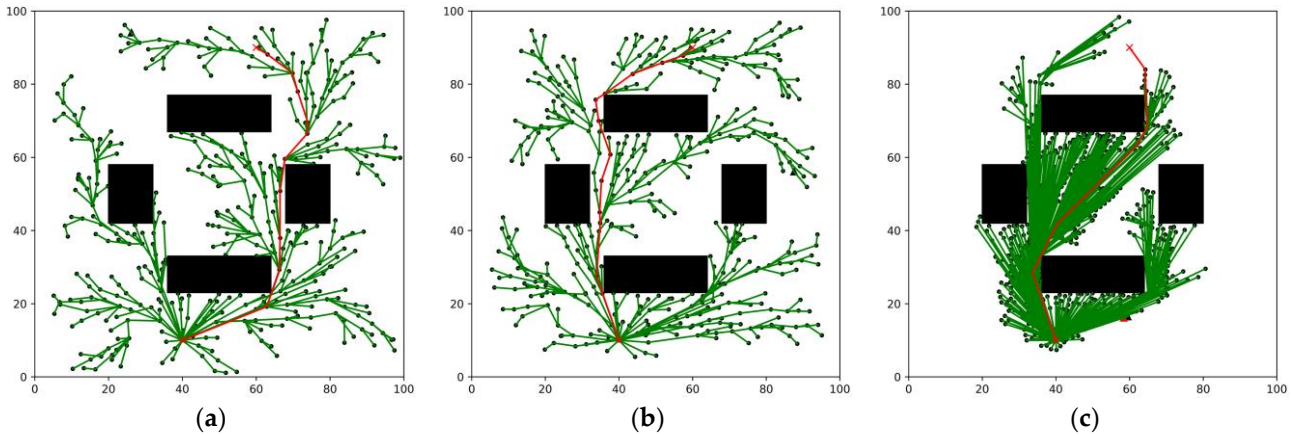

Figure 5. Simulation results in Map A: (**a**) RRT*; (**b**) P-RRT*; and (**c**) improved P-RRT*.

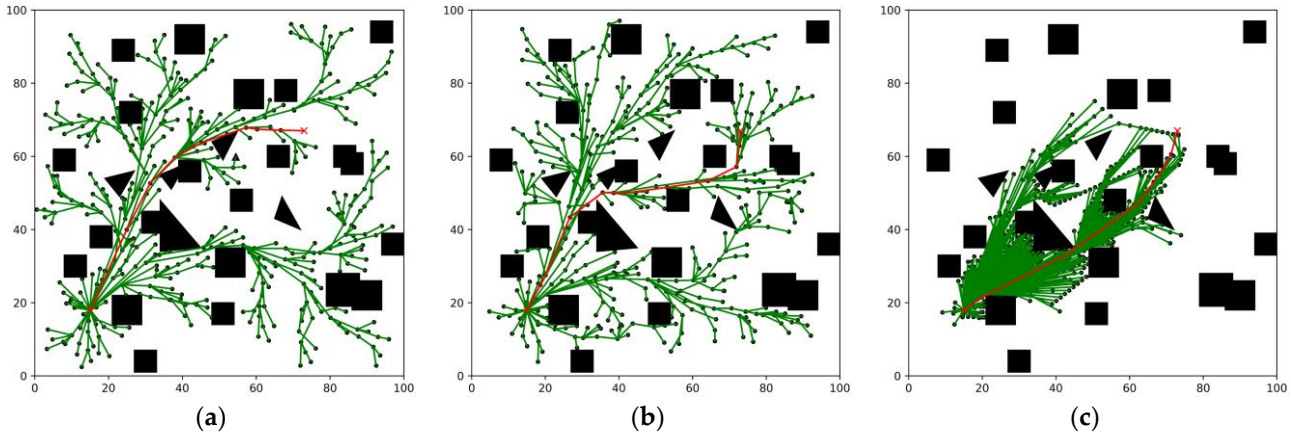

Figure 6. Simulation results in Map B. (**a**) RRT*; (**b**) P-RRT*; (**c**) Improved P-RRT*.

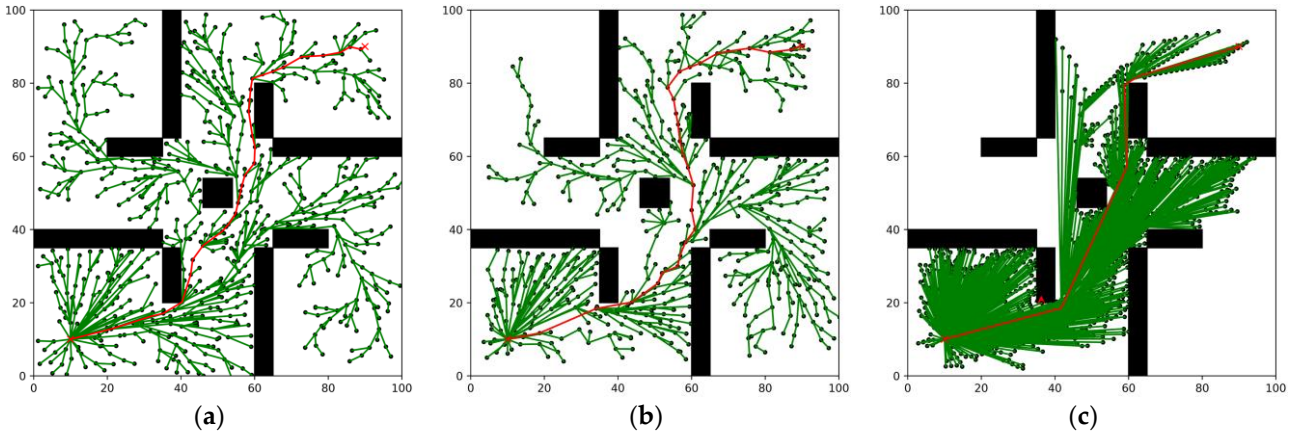

Figure 7. Simulation results in Map C: (**a**) RRT*; (**b**) P-RRT*; and (**c**) improved P-RRT*.

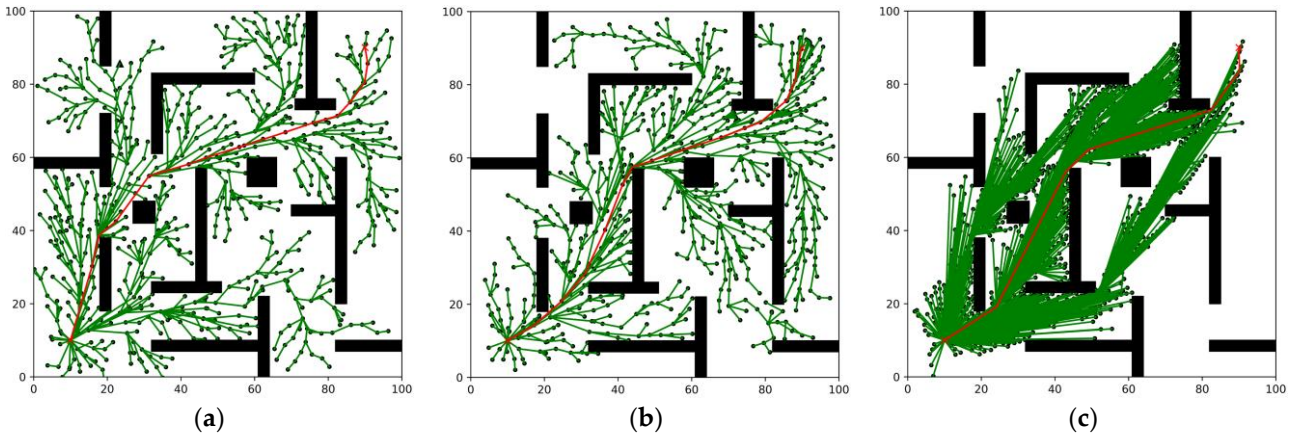

Figure 8. Simulation results in Map D: (**a**) RRT*; (**b**) P-RRT*; and (**c**) improved P-RRT*.

Table 1. Simulation data in 2D environment.

Environment	Algorithm	c_{min}(m)	t_{init}(s)	t_{cost}(s)	Fail	Nodes	Runtime
Map A	RRT*	107.86	0.288	13.00	31	415	0.308
	P-RRT*	105.32	0.338	11.10	11	397	0.477
	Improved P-RRT*	96.69	0.181	1.73	0	684	1.873
Map B	RRT*	87.34	0.552	25.47	5	394	1.179
	P-RRT*	83.53	0.642	19.96	6	375	2.105
	Improved P-RRT*	78.67	0.389	1.91	0	671	13.750
Map C	RRT*	145.84	0.550	14.27	18	631	0.880
	P-RRT*	141.12	1.011	14.72	3	610	1.592
	Improved P-RRT*	134.15	0.406	3.13	0	1199	5.149
Map D	RRT*	132.75	1.069	15.30	2	731	1.588
	P-RRT*	128.29	1.271	7.07	1	679	2.750
	Improved P-RRT*	122.66	0.791	2.24	0	1637	14.117

The cases plotted in the four maps highlights the characteristics of each algorithm. The random tree of RRT* was distributed over the whole area of the map, which slowed down the convergence of the algorithm. While the random tree of P-RRT* grew toward the target point as a whole, a large number of nodes were distributed at the edge of the map. The proposed algorithm has a large number of continuous nodes generated by the greedy strategy and the random tree is distributed in the center of the map. The proposed algorithm has a large number of continuous nodes generated by the greedy strategy and the random tree is distributed in the center of the map. Benefiting from pruning to remove redundant nodes, the proposed algorithm has the least number of path inflection points. As shown in Figure 5, RRT* has eight inflection points and P-RRT* has nine inflection points, while the proposed algorithm has only five inflection points, which is more suitable for planning the flight paths of UAVs.

The data in Table 1 and the data distribution reflect the advantages of the proposed algorithm. The proposed algorithm has the strongest search ability, and hence c_{min} is minimized. Taking Map A as an example, the c_{min} of the algorithm was reduced by 10.36% and 8.19% compared to RRT* and P-RRT*, respectively, which was mainly due to the high-cost rejection and pruning methods implemented by the algorithm. Among all four maps, P-RRT* had the largest time t_{init} searching for the initial path, RRT* had the second largest, and the proposed algorithm had the smallest. Since P-RRT* added the step of sampling-point deflection in the sampling procedure, this increased the time of each iteration. The proposed algorithm expanded the random tree according to the direction of the artificial potential field and incorporated a greedy strategy that allowed for fast expansion toward the target point at the early stage of the algorithm operation. In an empty environment, such as Map A, the proposed algorithm had the greatest advantage: t_{init} was reduced by 37.15% compared to RRT*. The t_{cost} data show that P-RRT* had a faster convergence speed compared to RRT* in complex environments. However, the boxplots in Figure 9a and c are very close to each other, which indicates that P-RRT* has no advantage in empty environments, such as those of Map A and Map C. The proposed algorithm sped up the convergence by rejecting high-cost nodes and sampling points, and the data show that the method is effective. In Map B, the proposed algorithm searched for excellent paths in only 1.91 s, compared with 25.47 s and 19.96 s for RRT* and P-RRT*, respectively, and they failed five and six times, respectively. The runtime data show that the proposed algorithm took more time. However, in practice, the proposed algorithm requires fewer iterations to obtain better quality paths, hence the time cost is lower.

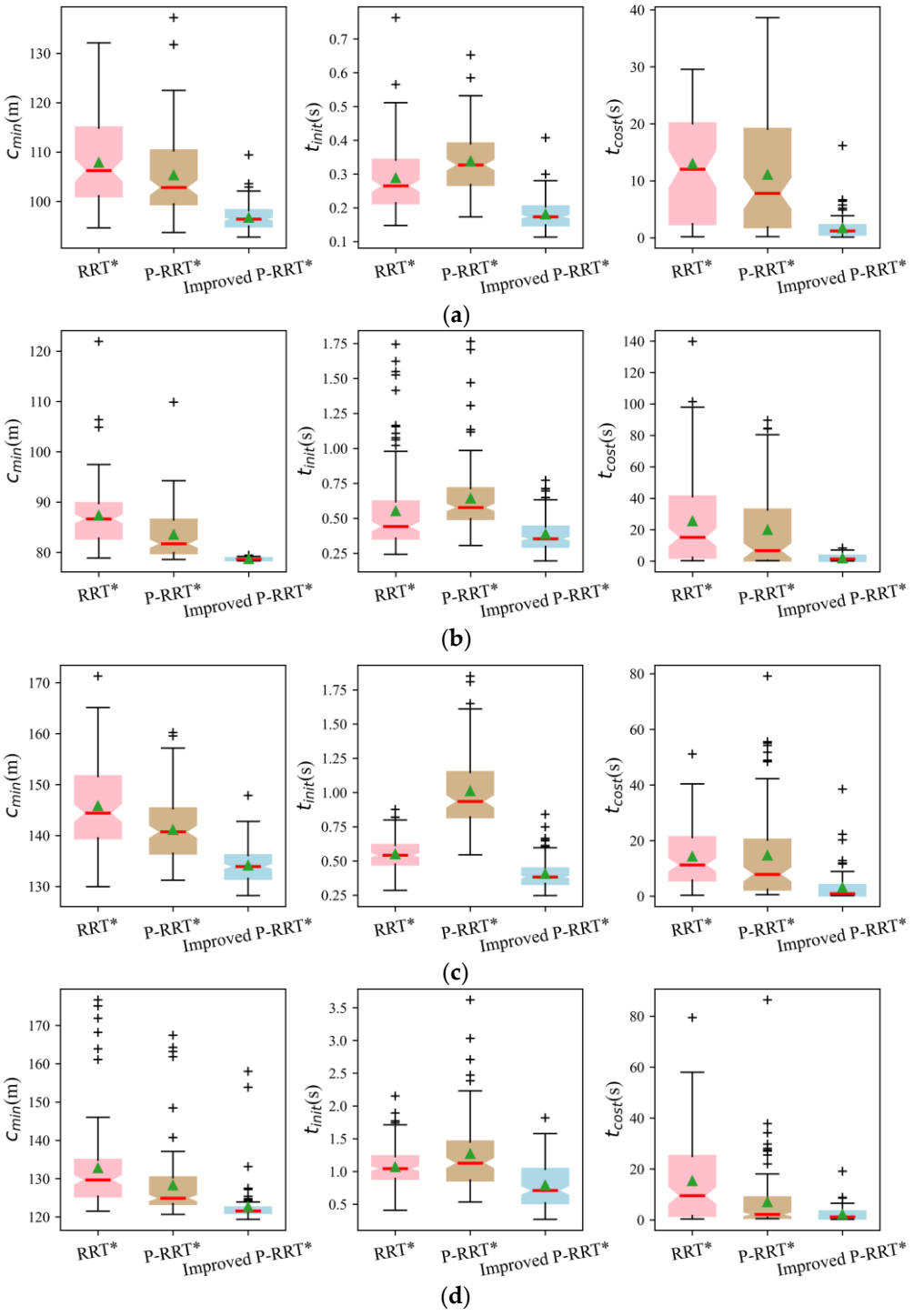

Figure 9. Simulation data distribution in 2D environment: (**a**) boxplots in Map A; (**b**) boxplots in Map B; (**c**) boxplots in Map C; and (**d**) boxplots in Map D. The + sign is the outliers, the green triangle is the mean, and the red line is the median.

4.2. 3D Environment

In reality, UAVs need to consider six directions of movement, so we constructed two 3D maps to simulate the realistic scenarios that UAVs may encounter. Map E, shown in Figure 10, simulated the environment of dense woods, where UAVs need to fly for operations such as plant data collection or pesticide spraying. To simplify the problem, only the main trunks of trees were kept as obstacles in this paper, and the map size is $50 \times 50 \times 20$. Map F, shown in Figure 11, simulated a real plant factory environment, with

tall plant cultivation racks used to grow crops with the greatest possible space utilization. The UAVs needed to quickly reach the target point according to the operator's command and perform pest and disease detection on crops that are at different heights on the plant cultivation racks. The size of Map F is 50 × 50 × 10. Table 2 shows the average data of 100 simulations of each algorithm in two maps, and the data visualization is depicted in Figure 12.

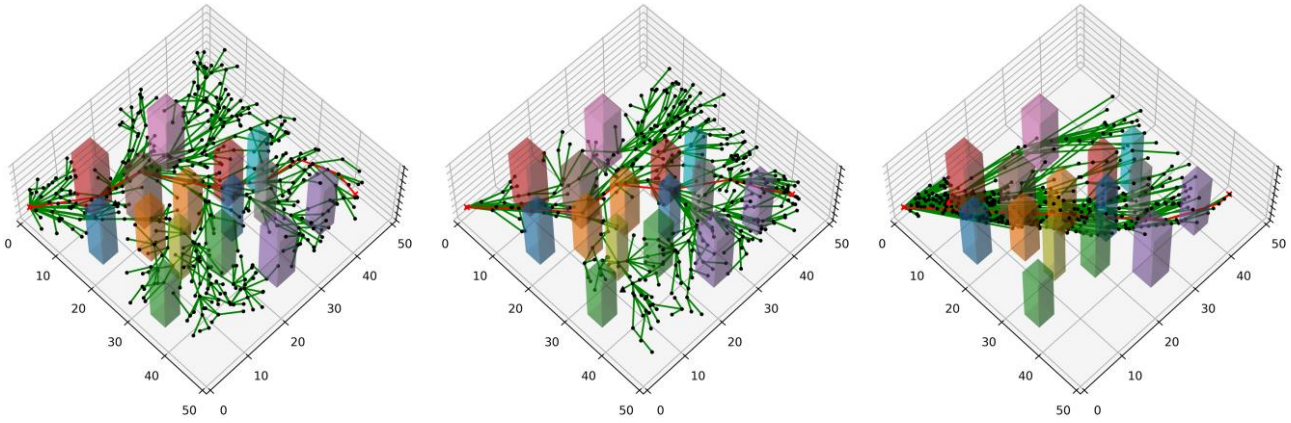

Figure 10. Simulation results in Map E: (**a**) RRT*; (**b**) P-RRT*; and (**c**) improved P-RRT*.

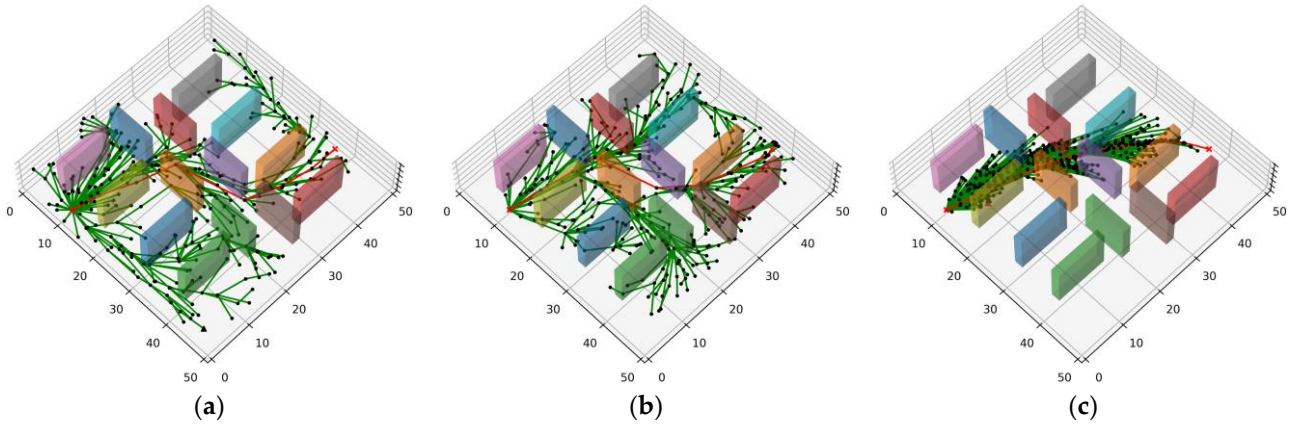

Figure 11. Simulation results in Map F: (**a**) RRT*; (**b**) P-RRT*; and (**c**) improved P-RRT*.

Table 2. Simulation data in 3D environment.

Environment	Algorithm	c_{min}(m)	t_{init}(s)	t_{cost}(s)	Fail	Nodes	Runtime
Map E	RRT*	70.12	0.328	16.99	29	414	0.548
	P-RRT*	67.33	0.400	10.22	2	384	1.185
	Improved P-RRT*	65.62	0.233	1.81	0	476	15.388
Map F	RRT*	59.54	0.274	9.38	0	335	0.527
	P-RRT*	57.96	0.352	6.71	0	287	0.998
	Improved P-RRT*	56.08	0.205	1.65	0	313	7.282

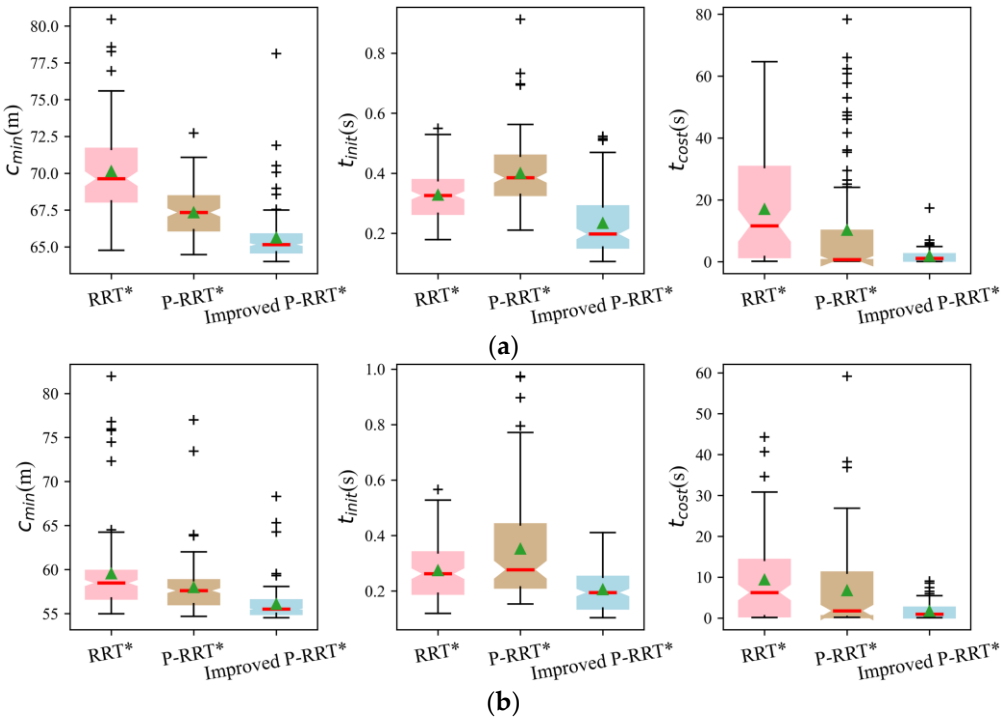

Figure 12. Simulation data distribution in 3D environment: (**a**) boxplots in Map E; and (**b**) boxplots in Map F. The + sign is the outliers, the green triangle is the mean, and the red line is the median.

Similar to the cases in the 2D environment, the random trees generated by RRT* were still distributed throughout the configuration space, while P-RRT* tried to make the random trees grow toward the target point under the gravitational force of the target point, but with limited effect. The proposed algorithm not only obtained paths with smaller costs but also with fewer inflection points. The c_{min} of the proposed algorithms in Map E and F was reduced by 2.54% and 3.24%, respectively, compared to that of P-RRT*, although the value of c_{min} was already very close to the optimal value. In the 3D environment, the proposed algorithm reduced the time by at least 25.18% more compared to that of RRT*. Contrary to the cases in the 2D environment, RRT* had difficulty in searching for excellent paths in the complex 3D environment and therefore had the largest t_{cost} and it had as many as 29 failures in Map E. The distribution of the t_{cost} data in Figure 12a shows that P-RRT* had a smaller median compared to that of the proposed algorithm, but due to the large number of outliers in the former, both the box and the mean were higher than those of the latter. The proposed algorithm limited the expansion of the random tree in the high-cost region and significantly improved the convergence speed. The data in Table 2 and the data distribution show that the proposed algorithm is more suitable for UAVs to deploy and fly rapidly to accomplish realistic application tasks.

5. Discussion

In this paper we improved the P-RRT* algorithm by using the artificial potential field to guide the greedy expansion and limit the exploration of high-cost regions. During the simulation experiments, we discovered that some algorithmic parameters can greatly affect the results. Theoretically, a smaller extension length can help the algorithm generate better paths, but this consumes more exploration time when the algorithm runs on maps with larger areas. The proportional setting of each component of $\vec{F}_{total}$ can also have an impact on the effectiveness of the algorithm. When the ratio of potential field forces exceeded 0.5, the random tree, as a whole, was obviously biased toward the target point, producing either superior or inferior effects in different maps. The proposed algorithm improved from P-RRT* required the calculation of attractive and repulsive forces, so that the algorithm can

set different K_a, K_r and d_{obs} at each map. Theoretically, larger d_{obs} can keep P-RRT* and improved P-RRT* away from obstacles, but we found that smaller d_{obs} produced better results in environments with cluttered obstacles. We understood that the goal-biased strategy was adopted in some improved RRT algorithms, but after simulation we did not find an obvious improvement in the algorithm when incorporating this strategy.

6. Conclusions

In this paper, we proposed an improved algorithm based on P-RRT* applied to UAVs, with the main goal of combining greedy strategy, rejecting high-cost nodes and sampling points, and optimizing paths. The greedy strategy can help with the expansion of the random tree to the optimal direction quickly and greatly reduce the time to obtain the initial solution. The algorithm rejected nodes and sampling points located in the high-cost region using the current path cost as the criterion, which improved the algorithm's iteration efficiency and sped up convergence. Finally, the random tree was pruned to reduce redundant nodes and complete the optimized path. Compared to P-RRT*, the improved P-RRT* not only had less cost and fewer path inflection points, but also obtained the initial solution and converged to the optimal solution with faster efficiency. Simulations in six environments showed that the proposed algorithm has significant advantages. Comparing the c_{min}, t_{init}, and t_{cost} indicators of the three algorithms, the results demonstrated that the proposed algorithm reduced the path cost by at least 2.54% and shortened the search time by at least 68.32% compared to P-RRT* in each environment. Moreover, the paths of the proposed algorithm had fewer inflection points. In conclusion, the proposed algorithm performed better under the same conditions, enabled the UAVs to complete planning more quickly, and generated a better path with fewer turning points.

The improved algorithm proposed in this paper was applied to the global path planning of UAVs, which was suitable for the environment where the information was known and there were no moving obstacles, The algorithm can be deployed on the ROS system under the Ubuntu system and replace the global planner in the move_base feature pack, which enables the UAVs to perform global path planning according to the customized algorithm. However, in real applications, UAVs will run into many unknown obstacles and need to use a local path-planning algorithm for dynamic obstacle avoidance. Future research will focus on combining global path planning and local path planning to adapt UAVs to real-world application environments.

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

Funding: This research was funded by National Key Research and Development Program of China (2019YFE0126100); Science and Technology project of Zhejiang Province (2019C54005); National Natural Science Foundation of China (61605173) and (61403346).

Data Availability Statement: Data are contained within the article.

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

References

1. Mozaffari, M.; Saad, W.; Bennis, M.; Nam, Y.-H.; Debbah, M. A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2334–2360. [CrossRef]
2. Shakhatreh, H.; Sawalmeh, A.H.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access* **2019**, *7*, 48572–48634. [CrossRef]
3. Maimaitijiang, M.; Sagan, V.; Sidike, P.; Hartling, S.; Esposito, F.; Fritschi, F.B. Soybean yield prediction from UAV using multimodal data fusion and deep learning. *Remote Sens. Environ.* **2020**, *237*, 111599. [CrossRef]

4. Lee, H.-W. Research on multi-functional logistics intelligent Unmanned Aerial Vehicle. *Eng. Appl. Artif. Intell.* **2022**, *116*, 105341. [CrossRef]
5. Rao, J.; Xiang, C.; Xi, J.; Chen, J.; Lei, J.; Giernacki, W.; Liu, M. Path planning for dual UAVs cooperative suspension transport based on artificial potential field-A* algorithm. *Knowl.-Based Syst.* **2023**, *277*, 110797. [CrossRef]
6. Sun, W.; Dai, L.; Zhang, X.; Chang, P.; He, X. RSOD: Real-time small object detection algorithm in UAV-based traffic monitoring. *Appl. Intell.* **2022**, *52*, 8448–8463. [CrossRef]
7. Hu, Z.; Zhang, Y.; Huang, H.; Wen, X.; Agbodike, O.; Chen, J. Reinforcement learning for energy efficiency improvement in UAV-BS access networks: A knowledge transfer scheme. *Eng. Appl. Artif. Intell.* **2023**, *120*, 105930. [CrossRef]
8. Silvagni, M.; Tonoli, A.; Zenerino, E.; Chiaberge, M. Multipurpose UAV for search and rescue operations in mountain avalanche events. *Geomat. Nat. Hazards Risk* **2017**, *8*, 18–33. [CrossRef]
9. Li, K.; Ge, F.; Han, Y.; Xu, W. Path planning of multiple UAVs with online changing tasks by an ORPFOA algorithm. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103807. [CrossRef]
10. Yang, Y.; Leeghim, H.; Kim, D. Dubins Path-Oriented Rapidly Exploring Random Tree* for Three-Dimensional Path Planning of Unmanned Aerial Vehicles. *Electronics* **2022**, *11*, 2338. [CrossRef]
11. Bell, M.G. Hyperstar: A multi-path Astar algorithm for risk averse vehicle navigation. *Transp. Res. Part B Methodol.* **2009**, *43*, 97–107. [CrossRef]
12. Dijkstra, E. A Note on Two Problems in Connexion with Graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
13. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
14. Hawa, M. Light-assisted A∗ path planning. *Eng. Appl. Artif. Intell.* **2013**, *26*, 888–898. [CrossRef]
15. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]
16. Sang, H.; You, Y.; Sun, X.; Zhou, Y.; Liu, F. The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations. *Ocean Eng.* **2021**, *223*, 108709. [CrossRef]
17. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **1996**, *26*, 29–41. [CrossRef]
18. Miao, C.; Chen, G.; Yan, C.; Wu, Y. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. *Comput. Ind. Eng.* **2021**, *156*, 107230. [CrossRef]
19. Gerke, M. Genetic path planning for mobile robots. In Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251), San Diego, CA, USA, 2–4 June 1999; IEEE: Piscataway, NJ, USA, 1999; pp. 2424–2429.
20. Niu, H.; Ji, Z.; Savvaris, A.; Tsourdos, A. Energy efficient path planning for Unmanned Surface Vehicle in spatially-temporally variant environment. *Ocean Eng.* **2020**, *196*, 106766. [CrossRef]
21. Kavraki, L.E.; Svestka, P.; Latombe, J.-C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
22. LaValle, S.M. Rapidly-Exploring random trees: A new tool for path planning. *Annu. Res. Rep.* 1998. Available online: http://lavalle.pl/papers/Lav98c.pdf (accessed on 1 November 2023).
23. Kavraki, L.E.; Kolountzakis, M.N.; Latombe, J.-C. Analysis of probabilistic roadmaps for path planning. *IEEE Trans. Robot. Autom.* **1998**, *14*, 166–171. [CrossRef]
24. LaValle, S.M.; Kuffner, J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
25. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
26. Lonklang, A.; Botzheim, J. Improved rapidly exploring random tree with bacterial mutation and node deletion for offline path planning of mobile robot. *Electronics* **2022**, *11*, 1459. [CrossRef]
27. Islam, F.; Nasir, J.; Malik, U.; Ayaz, Y.; Hasan, O. Rrt*-Smart: Rapid convergence implementation of RRT* towards optimal solution. In Proceedings of the 2012 IEEE International Conference on Mechatronics and Automation, Chengdu, China, 5–8 August 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1651–1656.
28. Tak, H.-T.; Park, C.-G.; Lee, S.-C. Improvement of RRT*-smart algorithm for optimal path planning and application of the algorithm in 2 & 3-dimension environment. *J. Korean Soc. Aviat. Aeronaut.* **2019**, *27*, 1–8.
29. Jordan, M.; Perez, A. Optimal Bidirectional Rapidly-Exploring Random Trees. *Res. Rep.* 2013. Available online: https://dspace.mit.edu/bitstream/handle/1721.1/79884/MIT-CSAIL-TR-2013-021.pdf (accessed on 1 November 2023).
30. Tahir, Z.; Qureshi, A.H.; Ayaz, Y.; Nawaz, R. Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments. *Robot. Auton. Syst.* **2018**, *108*, 13–27. [CrossRef]
31. Wang, J.; Li, B.; Meng, M.Q.-H. Kinematic Constrained Bi-directional RRT with Efficient Branch Pruning for robot path planning. *Expert Syst. Appl.* **2021**, *170*, 114541. [CrossRef]
32. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 2997–3004.
33. 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]
34. Li, Y.; Wei, W.; Gao, Y.; Wang, D.; Fan, Z. PQ-RRT*: An improved path planning algorithm for mobile robots. *Expert Syst. Appl.* **2020**, *152*, 113425. [CrossRef]

35. Qureshi, A.H.; Ayaz, Y. Potential functions based sampling heuristic for optimal path planning. *Auton. Robot.* **2016**, *40*, 1079–1093. [CrossRef]
36. Fan, J.; Chen, X.; Wang, Y.; Chen, X. UAV trajectory planning in cluttered environments based on PF-RRT* algorithm with goal-biased strategy. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105182. [CrossRef]
37. Liao, B.; Wan, F.; Hua, Y.; Ma, R.; Zhu, S.; Qing, X. F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate. *Expert Syst. Appl.* **2021**, *184*, 115457. [CrossRef]
38. Yang, K. Anytime synchronized-biased-greedy rapidly-exploring random tree path planning in two dimensional complex environments. *Int. J. Control Autom. Syst.* **2011**, *9*, 750–758. [CrossRef]

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