MDPI

Article

A Sampling-Based Algorithm with the Metropolis Acceptance Criterion for Robot Motion Planning

Yiyang Liu [1,2,3,4], Yang Zhao [1,2,3,5,*], Shuaihua Yan [1,2,3,6], Chunhe Song [1,2,3,*] and Fei Li [1,2,3,7]

1 Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, Shenyang 110016, China
2 Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
3 Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China
4 Kunshan Intelligent Equipment Research Institute, Kunshan 215300, China
5 School of Automation and Electrical Engineering, Shenyang Ligong University, Shenyang 110159, China
6 School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China
7 College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
* Correspondence: zhaoyang2@sia.cn (Y.Z.); songchunhe@sia.cn (C.S.)

Abstract: Motion planning is one of the important research topics of robotics. As an improvement of Rapidly exploring Random Tree (RRT), the RRT* motion planning algorithm is widely used because of its asymptotic optimality. However, the running time of RRT* increases rapidly with the number of potential path vertices, resulting in slow convergence or even an inability to converge, which seriously reduces the performance and practical value of RRT*. To solve this issue, this paper proposes a two-phase motion planning algorithm named Metropolis RRT* (M-RRT*) based on the Metropolis acceptance criterion. First, to efficiently obtain the initial path and start the optimal path search phase earlier, an asymptotic vertex acceptance criterion is defined in the initial path estimation phase of M-RRT*. Second, to improve the convergence rate of the algorithm, a nonlinear dynamic vertex acceptance criterion is defined in the optimal path search phase, which preferentially accepts vertices that may improve the current path. The effectiveness of M-RRT* is verified by comparing it with existing algorithms through the simulation results in three test environments.

Keywords: motion planning; sampling-based algorithms; RRT; Metropolis acceptance criterion; asymptotic optimality

check for updates

Citation: Liu, Y.; Zhao, Y.; Yan, S.; Song, C.; Li, F. A Sampling-Based Algorithm with the Metropolis Acceptance Criterion for Robot Motion Planning. *Sensors* **2022**, *22*, 9203. https://doi.org/10.3390/s22239203

Academic Editors: Luis Payá, Oscar Reinoso García and Helder Jesus Araújo

Received: 21 October 2022
Accepted: 23 November 2022
Published: 26 November 2022

1. Introduction

Robotics is evolving rapidly and has dramatically improved the efficiency of industrial production and the convenience of people's lives. Motion planning is indispensable in robotics, which requires finding a feasible path from the initial state to the target state subject to obstacle avoidance constraints. Nowadays, motion planning has been widely applied to various robots, including but not limited to industrial robots [1,2], free-floating space robots [3], rescue robots [4], medical robots [5], and autonomous vehicles [6].

According to the research order and fundamental principles, various motion planning algorithms can be mainly divided into four categories: bionic algorithms, artificial potential field methods, grid-based searches, and sampling-based algorithms [7]. The ant colony algorithm [8], one of the representative bionic algorithms, draws a lesson from the behavior of ants exploring paths to find food, showing strong robustness. However, it has the problems of slow convergence and a poor quality of the solution when dealing with large-scale problems [9]. The artificial potential field (APF) [10] method assumes a gravitational force of the goal state and repulsive forces of the obstacles. By calculating their resultant force, the following motion state of the moving object can be determined. Though APF has a simple structure and a small amount of computation, it is faced with the disadvantages of a local minimum value, large path oscillations, and complex path searches between similar obstacles [11]. The grid-based searches represented by A* [12,13] map motion planning problems into graphs and solve them in discrete state spaces. Although A* can return to the path with a minimal cost, the computation time and storage space of the data grow exponentially as the dimension of the state space increases.

Compared with grid-based searches, sampling-based algorithms avoid discretizing the state space in motion planning and efficiently perform in high-dimensional state spaces, such as Rapidly exploring Random Tree (RRT) [14] and Probabilistic RoadMap (PRM) [15]. Since RRT-based algorithms have the characteristics of a high search efficiency and low resource consumption, they have been frequently used to solve manipulators, autonomous surface vehicles, and other robot motion planning problems [16–18]. Furthermore, many variants of RRT have emerged. RRT-connect [19] simultaneously expands two trees from the initial and target states and then connects the two at the appropriate position, speeding up the pathfinding process. Kang et al. proposed a triangular inequality-based rewiring method for the RRT-connect algorithm [20]. The improved algorithm shows a shorter path length than RRT-connect. The above algorithms have probabilistic completeness, i.e., as the number of iterations approaches infinity, the probability of finding a feasible solution tends towards one. However, none of these methods can ensure optimality. RRT* [21] solves the problem by adopting the ChooseParent and Rewire procedures, which provide asymptotic optimality. An algorithm with asymptotic optimality means that as the number of iterations approaches infinity, the algorithm guarantees that the probability of finding an optimal solution approaches one [22].

Although RRT* can improve the initial solution to the optimum, it has a slow convergence rate due to the large amount of computation caused by the continuous increase in the number of vertices. Therefore, enhancing the convergence rate of RRT* is of great significance and has attracted extensive attention. Jonathan et al. proposed Informed-RRT* [23] based on direct sampling. This method defines the acceptable sampling space as a hyper-ellipsoid and samples directly from it. By narrowing the sampling region, Informed-RRT* may converge to the optimal path rapidly. However, when the relevant hyper-ellipsoid exceeds the region of the motion planning problem, the algorithm will not be applicable. Quick-RRT* [24] uses triangular inequality to improve the ChooseParent and Rewire procedures and converges faster than RRT*. The downside is that it wastes many computing resources on useless vertices, which do not help in finding the optimal path. MOD-RRT* [25] introduces a re-planning procedure to improve the performance of the algorithm, which is used to modify unfeasible paths to generate high-quality paths. GMR-RRT* [26] uses Gaussian mixture regression (GRM). This algorithm learns the human driving path and finds key features through GRM to form a probability distribution to guide sampling. Jun et al. proposed Feedback-RRT* (F-RRT*) [27], with a data-driven risk network and feedback module. F-RRT* can use the information extracted from situation data to constrain the growth of the random tree and make biased adjustments to improve planning efficiency.

This paper proposes Metropolis-RRT* (M-RRT*) for the motion planning of mobile robots. Based on the principle of the Metropolis acceptance criterion, on one hand, in the initial path estimation phase of the algorithm, an asymptotic vertex acceptance criterion is developed, which preferentially accepts vertices close to the target state. On the other hand, in the optimal path search phase, a nonlinear dynamic vertex acceptance criterion is also developed, which preferentially accepts vertices that may lower the current optimal path cost. Therefore, M-RRT* substantially reduces the number of vertices required for planning, thereby obtaining the initial path more efficiently and converging to the optimal path faster.

The M-RRT* algorithm proposed in this study is applied to solve the motion planning problem of mobile robots, with the following main contributions:

- Propose an asymptotic vertex acceptance criterion in the initial path estimation phase of the algorithm to effectively reduce the time of finding the initial path and make the algorithm start searching for the optimal path earlier.

- Propose a nonlinear dynamic vertex acceptance criterion in the optimal path search phase of the algorithm. This criterion can reduce the number of vertices in the algorithm that are not capable of improving the current path so as to rapidly converge to the optimal path.
- The experimental results in three common types of environments show that the proposed algorithm has an outstanding performance. It takes less time to find the initial path and has a fast convergence rate in the cluttered environment and the regular environment. Due to the complexity of the maze, M-RRT* does not improve much in the initial path estimation phase, but more importantly, its convergence speed still increases significantly.

The rest of this paper is structured as follows. Section 2 introduces the definition of the motion planning problem and related algorithms. Section 3 explains the proposed algorithm in detail. Section 4 analyzes the simulation results compared with RRT*, Informed-RRT*, and Q-RRT*. Section 5 summarizes this paper and looks forward to future research plans.

2. Problem Definition and Related Work

This section defines the motion planning problem and details the basic algorithm of M-RRT*. To understand the comparative analysis in Section 4, this section also provides an explanation of Informed-RRT* and Q-RRT*.

2.1. Problem Definition

When a robot performs a given task, the robot itself or a certain part moves from the initial state to the target state. Motion planning can provide the robot with a collision-free path from the initial state to the target state, which is called a feasible path. As mentioned in the previous section, motion planning has been applied to various robots, and the planning results directly affect the efficiency of the robot in completing a given task. Planners require motion planning not only to be able to search for an optimal path but also to make the planning process as fast as possible. The mathematical definition of motion planning is described in detail below.

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a d-dimensional configuration space, $\mathcal{X}_{obs} \subset \mathcal{X}$ be the obstacle region, and $\mathcal{X}_{free} = cl(\mathcal{X} \backslash \mathcal{X}_{obs})$ be the obstacle-free region, where $d \in \mathbb{N}$, $d \geq 2$, and $cl(\cdot)$ refers to the closure of the set. $x_{init} \in \mathcal{X}_{free}$ is the initial state, and $\mathcal{X}_{goal} \subset \mathcal{X}_{free}$ is the goal region, where the motion planning will be completed if the object reaches this region. The continuous function σ: $[0, 1] \mapsto \mathcal{X}$ is called a path. A path σ: $[0, 1] \mapsto \mathcal{X}_{free}$ is collision-free if $\forall \tau \in [0, 1]$, $\sigma(\tau) \in \mathcal{X}_{free}$.

Definition 1. *(Feasible motion planning) Given a motion planning problem* $\{x_{init}, \mathcal{X}_{goal}, \mathcal{X}_{obs}\}$*, find a collision-free path* σ*, where* $\sigma(0) = x_{init}$, $\sigma(1) \in \mathcal{X}_{goal}$*. This path is called a feasible path.*

Definition 2. *(Optimal motion planning) Given a motion planning problem* $\{x_{init}, \mathcal{X}_{goal}, \mathcal{X}_{obs}\}$*, find a feasible path* σ *that minimizes the cost* $c(\sigma^*)$*, i.e.,* $c(\sigma^*) = min\{c(\sigma^*): \sigma \in \Sigma\}$*, where* $c(\sigma)$ *is the cost of a feasible path* σ *in* $\mathcal{X}$ *measured by the Euclidean distance, and* Σ *is the set of all feasible paths.*

Definition 3. *(Fast motion planning) Given a motion planning problem* $\{x_{init}, \mathcal{X}_{goal}, \mathcal{X}_{obs}\}$*, find the optimal feasible path* σ^* *in the shortest possible time.*

2.2. RRT*

RRT* is a sampling-based motion planning algorithm. Since RRT* is an extended algorithm of RRT, a brief description of its basic algorithm is required first. RRT gradually explores the collision-free path from the initial state x_{init} which is the only vertex with no edges. In each iteration, x_{rand} is randomly sampled in the collision-free space $\mathcal{X}_{free}$. Then, we find the vertex $x_{nearest}$ closest to the sample x_{rand} from the tree based on the Euclidean metric. After finding $x_{nearest}$, extend a distance from $x_{nearest}$ towards x_{rand} to obtain a new vertex x_{new}, and this distance η is called the step size. Determine whether the path from $x_{nearest}$ to x_{new} collides with obstacles based on the environmental information. If a collision occurs, the path is discarded, and the next iteration proceeds. If no collision occurs, it is a feasible path, and this path and the vertex x_{new} are added to the tree. Repeat the above steps until a feasible path is found and output. As shown in Algorithm 1, the major difference from RRT is that RRT* contains the ChooseParent and Rewire procedures, which makes it asymptotically optimal.

Algorithm 1 RRT*

```
1:  $G \leftarrow (V, E); V \leftarrow x_{init};$
2:  for $i = 1$ to $n$ do
3:      $x_{rand} \leftarrow Sample(i);$
4:      $x_{nearest} \leftarrow Nearest(V, x_{rand});$
5:      $(x_{new}, \sigma) \leftarrow Steer(x_{nearest}, x_{rand});$
6:      if $CollisionFree(\sigma)$ then
7:          $X_{near} \leftarrow Near(V, x_{new});$
8:          $(x_{parent}, \sigma_{parent}) \leftarrow ChooseParent(X_{near}, x_{nearest}, x_{new}, \sigma);$
9:          $V \leftarrow AddVertex(x_{new});$
10:         $E \leftarrow AddEdge(x_{parent}, x_{new});$
11:         $G \leftarrow Rewire(G, x_{new}, X_{near});$
12:     end if
13: end for
14: return $G;$
```

In the ChooseParent procedure, RRT* searches for a vertex in a hypersphere of a specific radius centered at x_{new} such that the path through this vertex to x_{new} has the lowest cost. It is then used as the parent vertex of x_{new}, as shown in Figure 1a. In the Rewire procedure, the vertices in the hypersphere are represented by x_{near} in turn. Compare the cost of the current path to x_{near} with the cost of the path through x_{new} to x_{near} until every vertex in the hypersphere has been compared. During each comparison, if the path through x_{new} is less costly, change its wiring, as shown in Figure 1b. Algorithm 2 shows the ChooseParent procedure, and Algorithm 3 shows the Rewire procedure.

Algorithm 2 ChooseParent($X_{near}, x_{nearest}, x_{new}, \sigma_{nearest}$)

```
1:  $x_{min} \leftarrow x_{nearest};$
2:  $\sigma_{min} \leftarrow \sigma_{nearest};$
3:  $c_{min} \leftarrow Cost(x_{min}) + Cost(\sigma_{min});$
4:  for each $x_{near} \in X_{near}$ do
5:      $\sigma \leftarrow Connect(x_{near}, x_{new});$
6:      $c \leftarrow Cost(x_{near}) + Cost(\sigma);$
7:      if $c < c_{min}$ then
8:          if $CollisionFree(\sigma)$ then
9:              $x_{min} \leftarrow x_{near};$
10:             $\sigma_{min} \leftarrow \sigma;$
11:             $c_{min} \leftarrow c;$
12:         end if
13:     end if
14: end for
15: return $(x_{min}, \sigma_{min});$
```

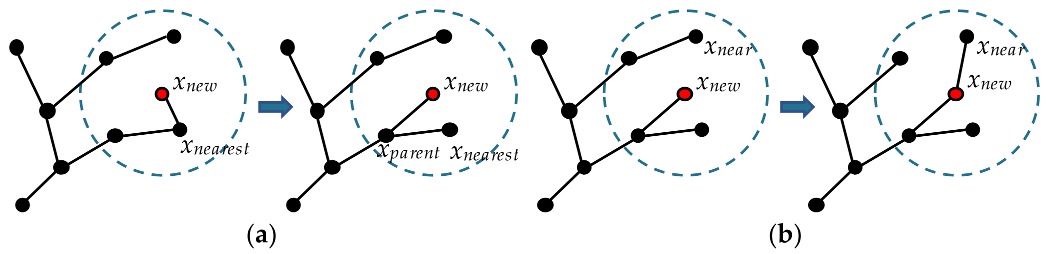

Figure 1. (**a**) ChooseParent (RRT*); (**b**) Rewire (RRT*). (red dot: x_{new}, dashed blue circle: *Near*(x_{new}.)).

Algorithm 3 Rewire(G, x_{new}, X_{near})

```
1: for each $x_{near} \in X_{near}$ do
2:     $\sigma \leftarrow Connect(x_{new}, x_{near})$;
3:     if $Cost(x_{new}) + Cost(\sigma) < Cost(x_{near})$ then
4:         if $CollisionFree(\sigma)$ then
5:             $G \leftarrow Reconnect(G, x_{new}, X_{near}, \sigma)$;
6:         end if
7:     end if
8: end for
9: return $G$;
```

The following briefly describes the basic procedures used in the RRT* algorithm.

- *Sample*: Returns a random sample from $\mathcal{X}$.
- *Nearest*: Given a graph $G = (V, E)$ and a state x, it returns the vertex closest to x according to the given Euclidean distance function.
- *Steer*: Given two states $x_s, x_t \in \mathcal{X}$, it extends x_s to x_t by a given distance to obtain the state $x_d \in \mathcal{X}$. Then, it returns x_d and the path from x_s to x_t.
- *CollisionFree*: Checks whether the given path σ is a feasible path.
- *Near*: Given a graph $G = (V, E)$ and a state x, it sets a hypersphere of a given radius centered at x and returns the set X_{near} of vertices in V that are contained in this hypersphere.
- *AddVertex*: Given a state x, it adds x to the graph.
- *AddEdge*: Given two states $x_s, x_t \in \mathcal{X}$, it adds the path from x_s to x_t to the graph.
- *Connect*: Given two states $x_s, x_t \in \mathcal{X}$, it returns the path from x_s to x_t.
- *Reconnect*: Given a graph $G = (V, E)$, two states $x_s, x_t \in \mathcal{X}$, and a path σ from x_s to x_t, it replaces the parent vertex of x_t with x_s and adds σ to graph G.
- RRT searches for a feasible path by imitating a randomly grown tree, but this algorithm cannot find the optimal path. Unlike RRT, RRT* has asymptotic optimality by introducing the ChooseParent and Rewire procedures. However, due to the frequent running of these two procedures, the convergence speed of RRT* is slow.

2.3. Informed-RRT*

Informed-RRT* has the same procedure as RRT* until the initial path is found. In the optimal path search phase, Informed-RRT* constructs a hyper-ellipsoid with focal points x_{init} and x_{goal}. It then samples directly inside the hyper-ellipsoid to find the optimal path. The sampling region of the set of vertices satisfies

$$\|x - x_{init}\| + \|x_{goal} - x\| \leq c(\sigma*), \tag{1}$$

where x is the random sample, σ* is the best path with the minimum cost at the current time, $c(\sigma*)$ is the cost of σ*, and $\|x - y\|$ is the Euclidean distance between x and y. Figure 2 shows the hyper-ellipsoid set by Informed-RRT*. When the current best solution is updated, its cost $c(\sigma*)$ is also updated. Therefore, the sampling region, i.e., the hyper-ellipsoid, gradually becomes smaller, and the probability of the sample being on the optimal path is greater.

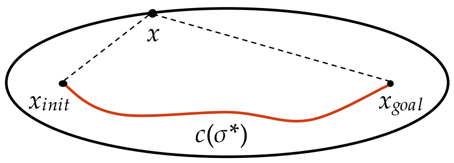

Figure 2. Hyper-ellipsoid of Informed-RRT*.

The direct sampling method significantly increases the convergence rate of In-formed-RRT*. However, Informed-RRT* does not improve the efficiency of finding the initial solution. Moreover, although Informed-RRT* converges faster than RRT*, its efficiency decreases when the region of the hyper-ellipsoid exceeds the configuration space.

2.4. Q-RRT*

Quick-RRT* (Q-RRT*) makes adjustments to the ChooseParent and Rewire procedures. In the ChooseParent procedure, RRT* searches for the potential parent vertex of x_{new} from X_{near}, while the search scope of Quick-RRT* also includes the ancestors of X_{near}. Quick-RRT* defines a depth in advance, which is used to set the number of vertices backtracked when searching for ancestors. In Figure 3a, through the ChooseParent procedure, x_{new} selects x_{parent} as the parent vertex, and x_{parent} is the ancestor vertex of $x_{nearest}$, with a depth of 2. Similar to the ChooseParent procedure, the Rewire procedure also considers the ancestor of the vertex x_{new}, as shown in Figure 3b.

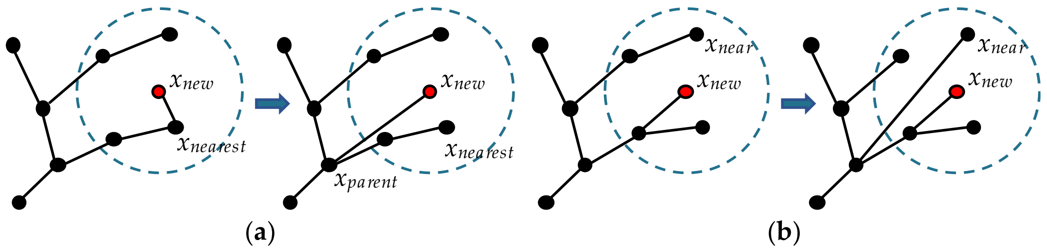

Figure 3. (**a**) ChooseParent (Q-RRT*); (**b**) Rewire (Q-RRT*).

Q-RRT* essentially improves the structure of the random tree. Compared with RRT*, the ChooseParent and Rewire procedures of Q-RRT* make the path more optimized. Although this algorithm can improve the structure, it needs to search more vertices. If most of the vertices do not meet the connection conditions, it means that the algorithm wastes a lot of time.

3. Methods

This section describes the proposed M-RRT* algorithm in detail. Most algorithms based on RRT* are divided into two phases: finding the initial path and finding the optimal path. The initial path is the first feasible path found by the algorithm. Since the initial path of the sampling-based algorithm is not optimal, it is necessary to continue sampling to find the optimal path.

Similarly, M-RRT* includes the initial path estimation and optimal path search phases. Inspired by the Metropolis acceptance criterion, we design different vertex acceptance criteria in the above two phases to calculate the retention probability of the new vertex generated by each iteration. M-RRT* uses the asymptotic vertex acceptance criterion in the initial path estimation phase. These enable the algorithm to obtain the initial solution faster. In the optimal path search phase, M-RRT* uses the nonlinear dynamic vertex acceptance criterion to improve the efficiency of finding the optimal solution. The following briefly introduces the Metropolis acceptance criterion.

3.1. Metropolis Acceptance Criterion

The core of the Metropolis acceptance criterion is the limited acceptance of inferior solutions. It is generally used in the simulated annealing algorithm to calculate the acceptance probability of a solution. The probability of a new solution being accepted is given by

$$P=\begin{cases} 1 & ,\Delta E<0 \\ \exp(-\Delta E/T), & \Delta E\geq 0 \end{cases} \tag{2}$$

where exp(·) refers to an exponential function, T is temperature, defined as the control parameter, E is internal energy, defined as the objective function, and $\Delta E = E(n+1) - E(n)$. The energy of the current state n of the system is $E(n)$, and the energy of the next state is $E(n+1)$. The algorithm tries to gradually decrease the value of the objective function as the temperature T decreases until E tends to the global minimum, just like the solid annealing process. A new solution will change the internal energy at the corresponding temperature, and the size of the change is ΔE. It can be seen from (2) that if $\Delta E < 0$, the new solution is accepted. If $\Delta E \geq 0$, the new solution is accepted by probability $P = \exp(-\Delta E/T)$. The following describes the effect of the variation of T on the candidate solutions.

- If T is a fixed value, the probability of accepting the candidate solution that reduces the value of E is greater than the probability of accepting the solution that increases the value.
- If T gradually decreases, the probability of accepting the candidate solution that increases the value of E also decreases.
- If T tends to zero, the algorithm only accepts the candidate solution that reduces the value of E.

Generally, a large T will perform a global search, but the computational cost will increase. While a small T will search locally and make a fast convergence rate, it tends to trap the algorithm in a local optimum.

3.2. Asymptotic Vertex Acceptance Criterion

The ChooseParent and Rewire procedures make RRT* asymptotically optimal, but frequent collision detection and searching for neighboring vertices increase the algorithm's complexity. Therefore, this paper introduces the asymptotic vertex acceptance criterion into the initial path estimation phase. After M-RRT* samples a new vertex, the impact of this vertex on finding the initial solution is quantified as a prediction value through a prediction function, and the probability of retaining this vertex is calculated according to the predicted value. If the new vertex is not retained, M-RRT* does not perform the subsequent calculation steps of this iteration and directly starts the next iteration. This method enables the random tree to grow to the target region faster; thus, an initial feasible solution is found more quickly.

To facilitate the description, we denote the target state by x_{goal}, the new vertex generated by the nth iteration by x_n, and the vertex closest to the x_{goal}, i.e., the vertex with the smallest Euclidean distance to x_{goal}, by x_{peak}. We can describe the process of finding an initial path as making x_{peak} asymptotically approach x_{goal} until it reaches the location of x_{goal}. The admissible heuristic estimate $\hat{h}$ is cost-to-go, i.e., the cost to go from any state to the goal. The prediction function for x_n is as follows,

$$C(n)=\hat{h}(x_n)-\hat{h}(x_{peak}) \tag{3}$$

where $\hat{h}(x_n)$ is the Euclidean distance from x_n to x_{goal} and $\hat{h}(x_{peak})$ is the Euclidean distance from x_{peak} to x_{goal}. Similar to the Metropolis acceptance criterion, we can judge whether x_n is closer to x_{goal} by $C(n)$. As shown in Figure 4, one of the following two cases occurs during the nth iteration.

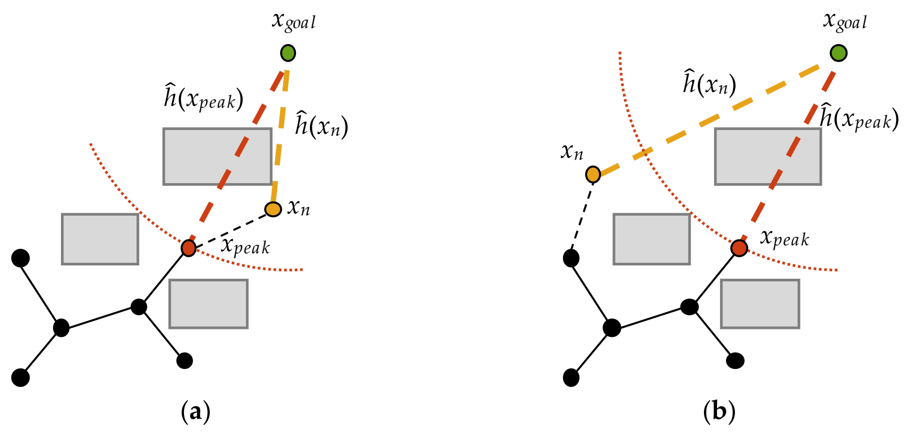

Figure 4. Tree construction of case (**a**) and case (**b**).

- If $C(n) < 0$, x_n lies in a spherical range with x_{goal} as the center and $\hat{h}(x_{peak})$ as the radius; this means that x_n is closer to x_{goal} than x_{peak}. Therefore, x_n is likely to play a positive role in finding the initial feasible path, at which point x_n is retained with a probability of one and is defined as x_{peak}, as shown in Figure 4a.
- If $C(n) \geq 0$, x_n is outside the spherical range, and it is further away from x_{goal} than x_{peak}. In this case, the probability of retaining x_n must be calculated based on the predicted value, as shown in Figure 4b.

It is worth noting that at the beginning of M-RRT*, the only vertex in the state space, x_{init}, which is the start state, is at the same position as x_{peak}. To quickly estimate the initial feasible solution, the heuristic value of each extended vertex preferably shows a downward trend. However, when $C(n) \geq 0$, x_n must be accepted with a certain probability to prevent the algorithm from falling into a local optimum. Therefore, this paper proposes the asymptotic vertex acceptance criterion combined with the prediction function. The probability of accepting x_n is

$$P_n = \begin{cases} 1 & , C(n) < 0 \\ \exp(-C(n)/\hat{h}(x_{init})), & C(n) \geq 0 \end{cases} \quad (4)$$

where $\hat{h}(x_{init})$ is the Euclidean distance from x_{init} to x_{goal}. Compared with x_{peak}, it is clear that the probability of acceptance is smaller if x_n is further away from the goal, and vice versa. Since $\hat{h}(x_{init})$ is the cost of the theoretical optimal path, it has a reference value for the calculation of the acceptance probability of vertices. In this paper, we define $\hat{h}(x_{init})$ as the control parameter.

In this phase, the probability of accepting each expanded vertex is greater than zero; thus, M-RRT* is capable of jumping out of the local optimum, but in some complex environments, it may take longer. We design a method to jump out of the local optimum quickly. If the value of $\hat{h}(x_{peak})$ does not change after using Equation (4) to calculate the probability 20 times, the algorithm is considered to fall into the local optimum. The acceptance probability of the subsequent extended new vertex is one until the value of $\hat{h}(x_{peak})$ changes. Then, the random tree is closer to the goal, indicating that the algorithm jumps out of the local optimum and continues to use Equation (4).

The asymptotic vertex acceptance criterion makes M-RRT* obtain the initial feasible path faster. At the same time, the algorithm performs the optimal path search phase earlier and improves the overall efficiency.

3.3. Nonlinear Dynamic Vertex Acceptance Criterion

Given an optimal path σ from x_{init} through $x \in \mathcal{X}$; to x_{goal}, $g(x)$ is equal to the cost of the optimal path from x_{init} to x, $h(x)$ is equal to the cost of the optimal path from x to x_{goal}, and the cost of σ is $c(\sigma) = g(x) + h(x)$. $\hat{g}(x)$ and $\hat{h}(x)$ are admissibility heuristics for

$g(x)$ and $h(x)$, respectively. In the problem of solving the optimal path length in $\mathbb{R}^d$, the Euclidean distance applies to both heuristics.

An algorithm based on RRT* will continue to iterate to search for the optimal path after completing the process of finding the initial feasible path. When x_n is generated at the nth iteration, there exists a feasible path σ^* with the minimum cost at the current time. To make σ^* approach the optimal path asymptotically, M-RRT* not only estimates the path cost from the new vertex x_n to x_{goal} but also considers the path cost from x_{init} to x_n. Next, we will introduce the nonlinear dynamic vertex acceptance criterion in detail, which is applied to the optimal path search phase of M-RRT*.

In Figure 5, $\hat{g}(x_n)$ is the Euclidean distance from x_{init} to x_n, $\hat{h}(x_n)$ is the Euclidean distance from x_n to x_{goal}, σ^* is the best path with the minimum cost at the current time, and $c(\sigma*)$ is the cost of σ^*. Obviously, $\hat{g}(x_n) + \hat{h}(x_n)$ is the cost of the theoretically optimal path through x_n.

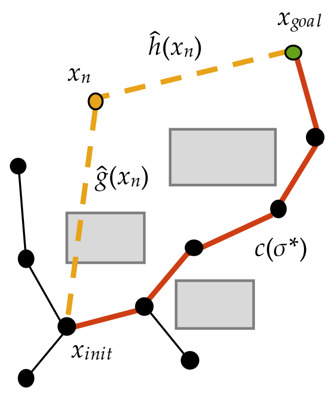

Figure 5. Tree construction.

If $c(\sigma*) < \hat{g}(x_n) + \hat{h}(x_n)$, it indicates that the cost of any path through x_n cannot be less than $c(\sigma*)$. Then, the probability of accepting x_n is $P_n = 0$, and M-RRT* starts the next iteration directly. Therefore, it removes vertices that are entirely impossible to improve the path length, avoiding wasting time on these useless vertices.

If $c(\sigma*) \geq \hat{g}(x_n) + \hat{h}(x_n)$, we must refer to the actual cost $g(x_n)$ from x_{init} to x_n and establish the prediction function

$$C(n) = g(x_n) + \hat{h}(x_n) - c(\sigma*) \tag{5}$$

where $g(x_n)$ is the actual cost of the path from x_{init} to x_n, and $\hat{h}(x_n)$ is the estimated cost from x_n to x_{goal}, which is the Euclidean distance. Similar to the Metropolis acceptance criterion, we predict whether the new vertex x_n can improve the current optimal path σ^* by $C(n)$. As shown in Figure 6, x_n is generated by extending the distance η from the nearest vertex $x_{nearest}$ on the random tree; thus, the actual cost $g(x_n) = g(x_{nearest}) + \eta$. As mentioned in the previous section, $\hat{h}(x_n)$ is the estimated cost of x_n to x_{goal}. One of the following two cases occurs during the nth iteration.

- If $C(n) < 0$, the predicted value is lower than the current minimum cost $c(\sigma*)$. Therefore, x_n is likely to play a positive role in finding the optimal path, and the probability of retaining x_n is one.
- If $C(n) \geq 0$, the probability of retaining x_n must be calculated based on the predicted value.

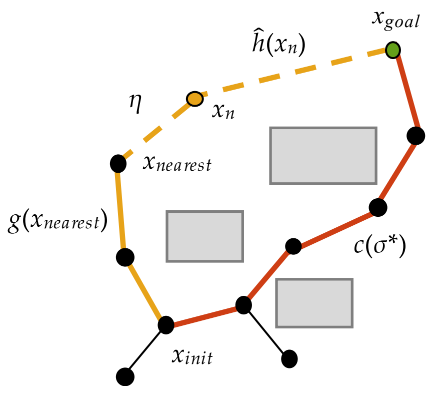

Figure 6. Tree construction. ($g(x_n) = g(x_{nearest}) + \eta$.).

In the case of $C(n) \geq 0$, the probability of retaining x_n cannot be zero, because after the ChooseParent and Rewire procedures, the actual cost from x_{init} to x_n may be reduced. In this case, x_n has the possibility of optimizing the path, but, of course, the larger the value of the prediction function, the smaller the possibility. Combined with the prediction function, the probability of accepting a vertex is

$$P_n = \begin{cases} 1 & ,C(n) < 0 \\ \exp(-\frac{C(n)}{c(\sigma*)/\ln(n-N-1+\mathrm{e})}) & ,C(n) \geq 0 \end{cases} \tag{6}$$

where $c(\sigma*)$ is the current minimum cost, n is the current number of iterations, and N is the number of iterations when σ^* is found. Therefore, N changes dynamically with the update of σ^*, and the next iteration number n after the update is equal to N + 1. The nonlinear dynamic cost $\{c(\sigma*)/\ln(n - N - 1 + \mathrm{e})\}$ increases from $c(\sigma*)$, and then the algorithm gradually reduces the probability of accepting vertices that make $C(n) \geq 0$. The nonlinear dynamic vertex acceptance criterion makes the algorithm sample in an extensive range in the early stage to prevent falling into the local optimum. As the number of iterations increases, the retained samples are more targeted, enabling the algorithm to quickly converge and find the optimal path. Moreover, there is always a probability of global sampling, allowing M-RRT* to explore a wider area while converging quickly, ensuring its asymptotic optimality.

3.4. M-RRT*

Compared with RRT*, M-RRT* adopts the asymptotic vertex acceptance criterion and nonlinear dynamic vertex acceptance criterion in finding the initial path and optimal path, respectively. It not only preserves the probability of global sampling but also selectively accepts vertices conducive to finding solutions and removes many useless vertices. M-RRT* prunes the random tree, significantly improving computational efficiency and reducing memory usage. Algorithm 4 shows the pseudocode of M-RRT*.

The following briefly describes the basic procedures used in the M-RRT* algorithm.

- *Dis*: Given two states $x_s, x_t \in \mathcal{X}$, it returns the Euclidean distance between x_s and x_t.
- *AVAC*: Asymptotic vertex acceptance criterion.
- *NDVAC*: Nonlinear dynamic vertex acceptance criterion.
- *NearGoal*: Given a graph $G = (V, E)$, a state x, and the current number of iterations, if x is within a given range around x_{goal}, this procedure returns the feasible path σ^* with the least cost and the current number of iterations N.
- Figure 7 shows the flow chart of M-RRT*, and l_{out} is the set path length to which to converge. The ChooseParent and Rewire procedures have been described in related work, so they are not shown in detail in the figure. The NearGoal procedure has also been introduced in this section.

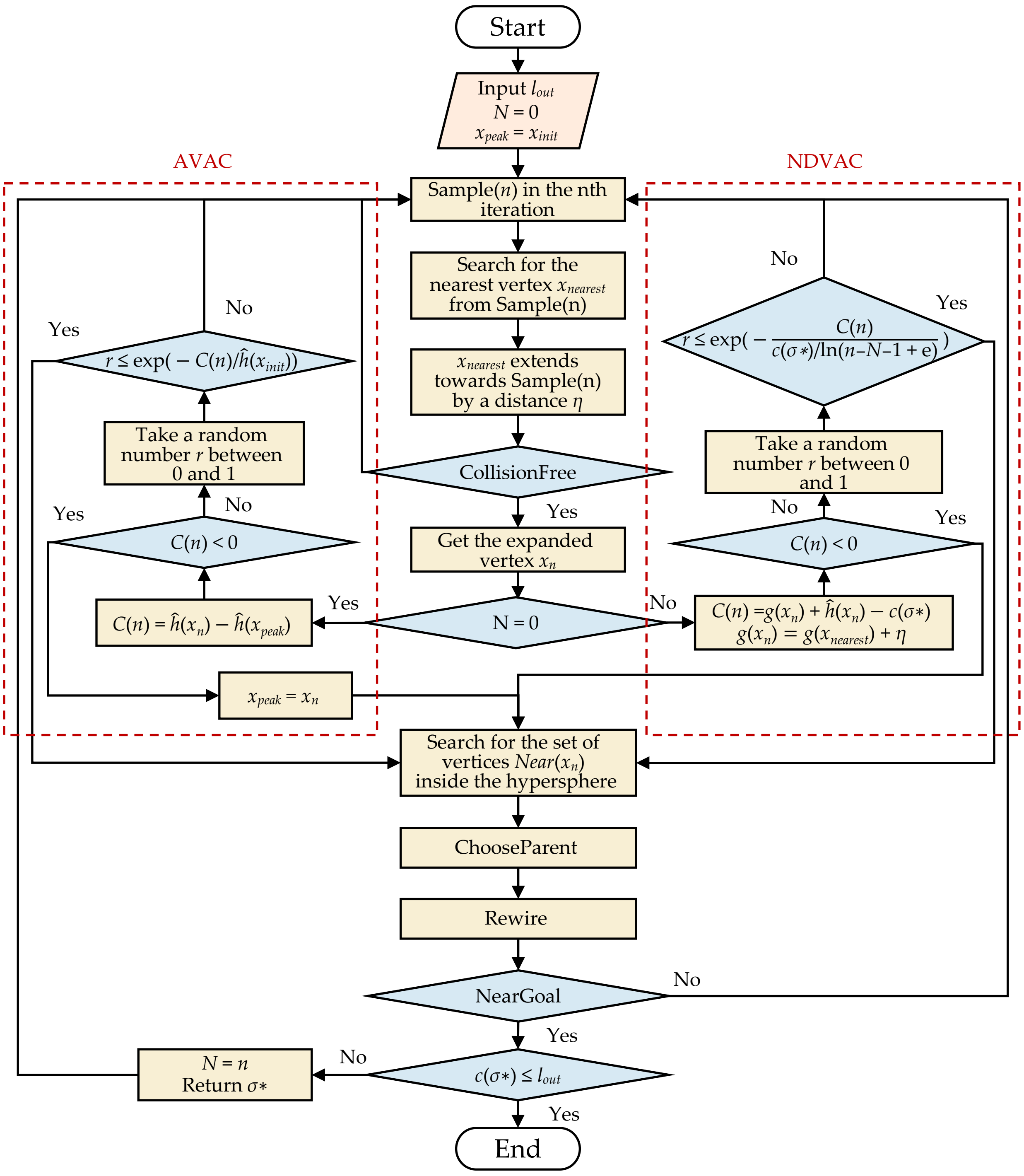
AVAC
NDVAC
Start
Input l_{out}
$N = 0$
$x_{peak} = x_{init}$
Sample(n) in the nth iteration
Search for the nearest vertex $x_{nearest}$ from Sample(n)
$x_{nearest}$ extends towards Sample(n) by a distance η
CollisionFree
Yes
Get the expanded vertex x_n
N = 0
Yes
No
$C(n) = \hat{h}(x_n) - \hat{h}(x_{peak})$
$C(n) < 0$
Yes
No
Take a random number r between 0 and 1
$r \leq \exp(-C(n)/\hat{h}(x_{init}))$
Yes
No
$x_{peak} = x_n$
$C(n) = g(x_n) + \hat{h}(x_n) - c(\sigma*)$
$g(x_n) = g(x_{nearest}) + \eta$
$C(n) < 0$
Yes
No
Take a random number r between 0 and 1
$r \leq \exp(-\frac{C(n)}{c(\sigma*)/\ln(n-N-1+e)})$
Yes
No
Search for the set of vertices $Near(x_n)$ inside the hypersphere
ChooseParent
Rewire
NearGoal
No
Yes
$c(\sigma*) \leq l_{out}$
No
Yes
$N = n$
Return $\sigma*$
End

Figure 7. Flow chart of M-RRT*.

Algorithm 4 M-RRT*

```
1:  $G \leftarrow (V, E)$; $V \leftarrow x_{init}$; $x_{peak} \leftarrow x_{init}$; $N \leftarrow 0$;
2:  $\hat{h}(x_{peak}) \leftarrow Dis(x_{peak}, x_{goal})$;
3:  $\hat{h}(x_{init}) \leftarrow Dis(x_{init}, x_{goal})$;
4:  for $i = 1$ to $n$ do
5:      $x_{rand} \leftarrow Sample(i)$;
6:      $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
7:      $(x_n, \sigma) \leftarrow Steer(x_{nearest}, x_{rand})$;
8:      if $CollisionFree(\sigma)$ then
9:          if $N = 0$ then
10:             $AVAC(x_n, \hat{h}(x_{peak}), \hat{h}(x_{init}))$;
11:         else
12:             $NDVAC(x_n, x_{nearest}, n, N, \sigma^*)$;
13:         $X_{near} \leftarrow Near(V, x_n)$;
14:         $(x_{parent}, \sigma_{parent}) \leftarrow ChooseParent(X_{near}, x_{nearest}, x_n, \sigma)$;
15:         $V \leftarrow AddVertex(x_n)$;
16:         $E \leftarrow AddEdge(x_{parent}, x_n)$;
17:         $G \leftarrow Rewire(G, x_n, X_{near})$;
18:         $(\sigma^*, N) \leftarrow NearGoal(G, x_n, n)$;
19:     end if
20: end for
21: return $G$;
```

4. Simulations

Since RRT* is the basic algorithm of the algorithm proposed in this paper, Informed-RRT* is widely recognized as an efficient RRT-based algorithm, and Q-RRT* is a newly proposed correlation algorithm, this paper uses them for comparative analysis. Q-RRT* has the best efficiency when the depth is 2, so 2 is chosen as the depth in this paper.

4.1. Environment Configuration and Indicator Description

M-RRT* is compared with the above three algorithms in three environments of the same size of 100×100. The simulation environments are shown in Figure 8. Each algorithm was run 100 times because of the randomness of the sampling-based algorithms. The system and resource characteristics used in the simulation implementation are shown in Table 1.

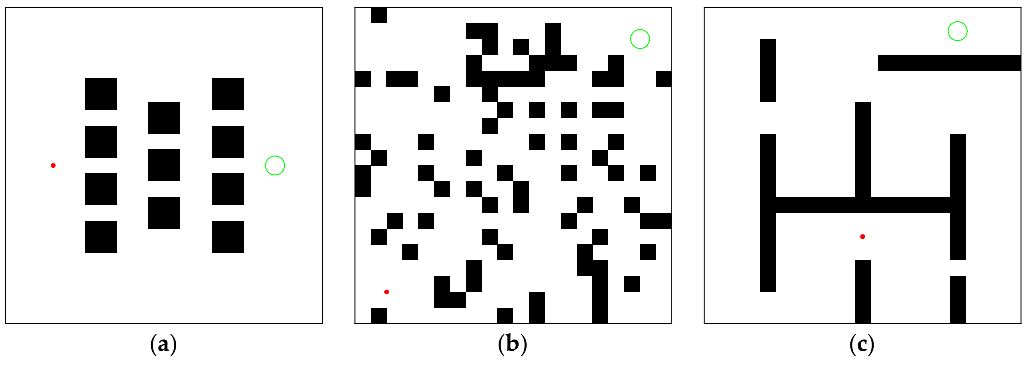

Figure 8. Environments for the simulations. (**a**) Regular environment; (**b**) Cluttered environment; (**c**) Maze. (Red dot: start state, green circle: goal region.).

Table 1. System and resource characteristics.

Simulation Environment	System	CPU	GPU	RAM
Python 3.9	Windows 10 Professional edition	Intel(R) Core(TM) I5-110400F	NVIDIA GeForce GTX 1050	16 G

In this section, all algorithms were simulated with the same parameters. Two evaluation indicators are used to compare the performances of the algorithms. Firstly, some problems require finding a feasible solution for motion planning in a short time; hence, this paper compares the time of the four algorithms to find the initial path. We define this time as the index 't_{init}'. Secondly, four algorithms have asymptotic optimality; thus, they can all find the optimal path as the number of iterations approaches infinity. Due to the randomness of the sampling-based algorithm, this paper first determines the optimal path length in each environment and then compares the time it takes for the algorithms to converge to an approximate optimal path. We define this time as the index '$T_{5\%}$'. The length of the approximate optimal solution is '$1.05{*}l_{optimal}$', where '$l_{optimal}$' is the length of the optimal solution.

Again, because of randomness, the statistical results of 100 runs of each algorithm are described by boxplots, and the specific values are shown in tables. In a boxplot, a line in the middle of the box represents the median of the data. The bottom and top of the box are the upper and lower quartiles of the data, respectively. Therefore, the height of the box reflects the fluctuation of the data to some extent. The median can show the performance of the randomness algorithms. In addition, the paper also describes the mean, maximum, and minimum to prove the efficiency and stability of the algorithms. The difference between the upper and lower quartiles is called the interquartile range (IQR). Values that are 1.5 times IQR larger than the upper quartile, or 1.5 times IQR less than the lower quartile, are classified as outliers and shown as rhombus in the boxplot.

4.2. Regular Environment

The regular environment is shown in Figure 8a. In Figure 9, the paths generated by RRT* (t_{init} = 0.1093, $T_{5\%}$ = 1.093), Informed-RRT* (t_{init} = 0.0781, $T_{5\%}$ = 0.453), Q-RRT* (t_{init} = 0.6876, $T_{5\%}$ = 0.7059), and M-RRT* (t_{init} = 0.0625, $T_{5\%}$ = 0.1406) are shown. In the regular environment, $l_{optimal}$ = 70.9268. The grey lines in each figure are the paths explored by the algorithm through the extended vertices, and the red line is the approximate optimal path. RRT* randomly sampled the entire state space; hence, its vertices grew in any direction. Unlike RRT*, once the initial path was found, Informed-RRT* sampled only inside the hyper-ellipsoid, with the start and target states as the foci. Therefore, most of the vertices of Informed-RRT* appear to be inside an ellipse. Q-RRT* optimizes the structure of the random tree so that the path between its vertices and the initial vertex is as straight as possible. Since M-RRT* preferentially accepted vertices that are conducive to searching for solutions and had a limited acceptance of vertices that provide inferior solutions, it generated fewer vertices than the other two algorithms.

The 100 simulation results of t_{init} and $T_{5\%}$ are described by boxplots, as shown in Figures 10 and 11, respectively. Table 2 counts the specific numerical values of the simulation results. For the convenience of the comparison, the data in the table are generally displayed to the fourth decimal place. If the size of the same indicator for the algorithms is very close, it shows the decimal place where their numbers are different. The two minimum values of the three algorithms are the same because the length of the initial path is less than $1.05{*}l_{optimal}$. Due to the regular distribution of obstacles in this environment and the small distance between the initial and goal states, the above situation is common for sampling-based algorithms. Although RRT* and Informed-RRT* have identical procedures for finding initial solutions, they are less stable, as shown in Figure 10. Since Q-RRT* consumes more computing resources on the optimized path, it takes a long time to find the initial path. In addition, M-RRT* takes less time to find the initial solution. It is clear from Figure 11 that M-RRT* converges faster than the other three algorithms. Based on the above analysis, M-RRT* outperforms RRT*, Informed-RRT*, and Q-RRT* in the regular environment.

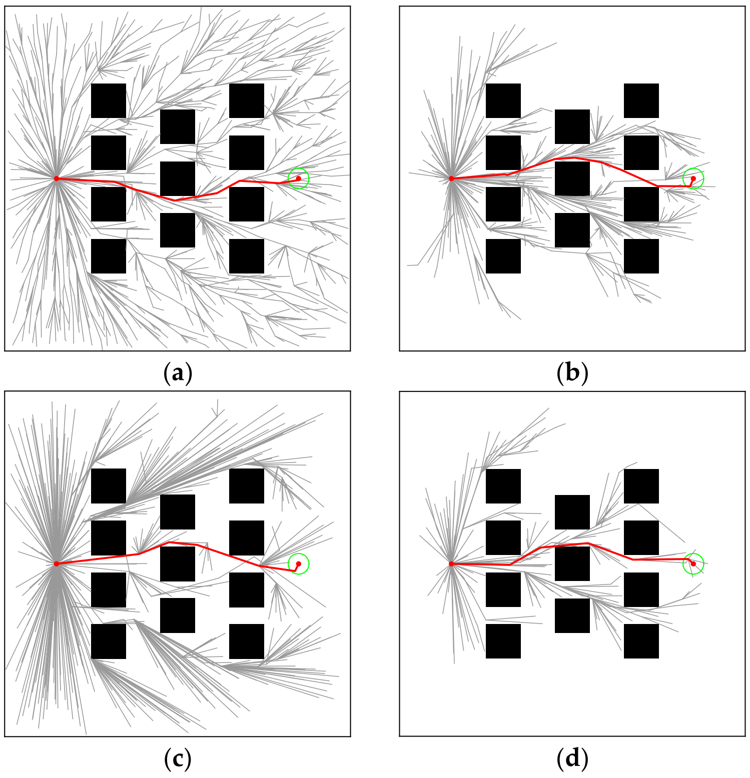

Figure 9. Performance of the four algorithms in the regular environment. (**a**) RRT*. (**b**) Informed-RRT*. (**c**) Q-RRT*. (**d**) M-RRT*. (Red line: approximate optimal path, green circle: goal region.).

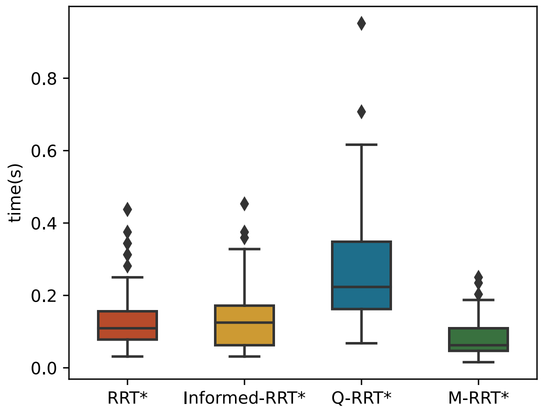

Figure 10. t_{init} in the regular environment.

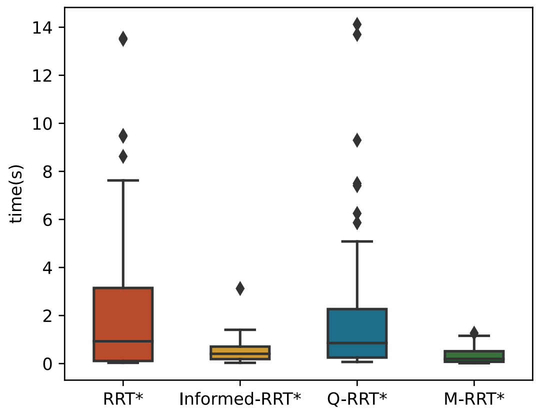

Figure 11. $T_{5\%}$ in the regular environment.

Table 2. t_{init} and $T_{5\%}$ of four algorithms in the regular environment.

Algorithm		Mean	Median	Min	Max
RRT*	t_{init}	0.1273	0.1093	0.031243	0.4374
	$T_{5\%}$	2.1116	0.9294	0.031243	13.5426
Informed-RRT*	t_{init}	0.1385	0.1249	0.031241	0.453
	$T_{5\%}$	0.488	0.4061	0.031241	3.1242
Q-RRT*	t_{init}	0.2739	0.2233	0.0678	0.9514
	$T_{5\%}$	1.8275	0.8547	0.0678	14.1172
M-RRT*	t_{init}	0.0799	0.0624	0.0156	0.2499
	$T_{5\%}$	0.3159	0.1952	0.0156	1.2653

4.3. Cluttered Environment

The cluttered environment is shown in Figure 8b. In Figure 12, the paths generated by RRT* (t_{init} = 0.337, $T_{5\%}$ = 3.1296), Informed-RRT* (t_{init} = 0.346, $T_{5\%}$ = 1.6216), Q-RRT* (t_{init} = 1.0265, $T_{5\%}$ = 4.6131), and M-RRT* (t_{init} = 0.2852, $T_{5\%}$ = 0.8666) are shown. In the cluttered environment, $l_{optimal}$ = 114.8326. RRT* required many vertices to explore the state space fully, while M-RRT* generated the fewest vertices to converge to the approximate optimal solution.

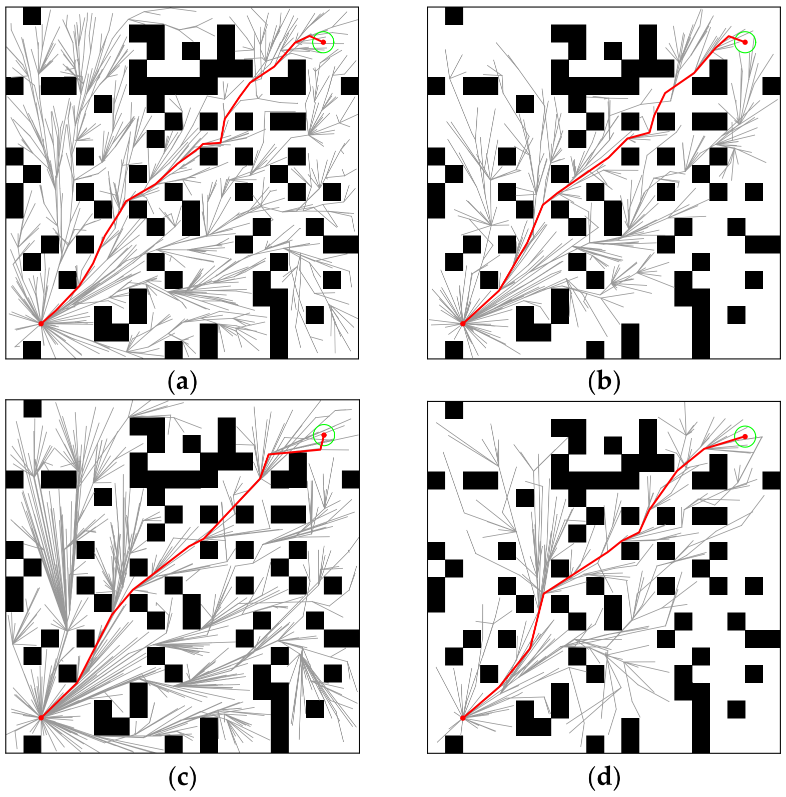

Figure 12. Performance of the four algorithms in the cluttered environment. (**a**) RRT*. (**b**) Informed-RRT*. (**c**) Q-RRT*. (**d**) M-RRT*. (Red line: approximate optimal path, green circle: goal region.).

The 100 simulation results of t_{init} and $T_{5\%}$ are described by boxplots, respectively. Table 3 counts the specific numerical values of the simulation results. As shown in Figure 13, in the cluttered environment, the median of M-RRT* is still the smallest, but the median of RRT* is larger than that of Informed-RRT*. Therefore, M-RRT* can obtain the initial solution earlier, while the other two algorithms are not stable. Q-RRT* still takes the longest time to find the initial path. As can be seen from Figure 14, although the efficiency of

Informed-RRT* searching for the optimal solution has been improved compared to RRT*, M-RRT* is the most effective algorithm. Q-RRT* can continuously search ancestor vertices to optimize paths, but these paths often collide in the cluttered environments. Thus, Q-RRT* wastes a lot of time trying to change the structure of the tree instead, and it performs the worst in this environment. According to the running results of two performance indicators, M-RRT* has an outstanding performance in the cluttered environment.

Table 3. t_{init} and $T_{5\%}$ of four algorithms in the cluttered environment.

Algorithm		Mean	Median	Min	Max
RRT*	t_{init}	0.3999	0.3514	0.1405	1.1716
	$T_{5\%}$	2.5054	2.1088	0.1874	11.1693
Informed-RRT*	t_{init}	0.3716	0.3124	0.1093497	1.3278
	$T_{5\%}$	1.6417	1.6089	0.1562	4.9363
Q-RRT*	t_{init}	1.0989	0.9808	0.3127	2.3687
	$T_{5\%}$	4.0272	3.7631	0.3127	11.7356
M-RRT*	t_{init}	0.3002	0.2811	0.1093492	0.8123
	$T_{5\%}$	1.1838	1.0075	0.1093492	4.7174

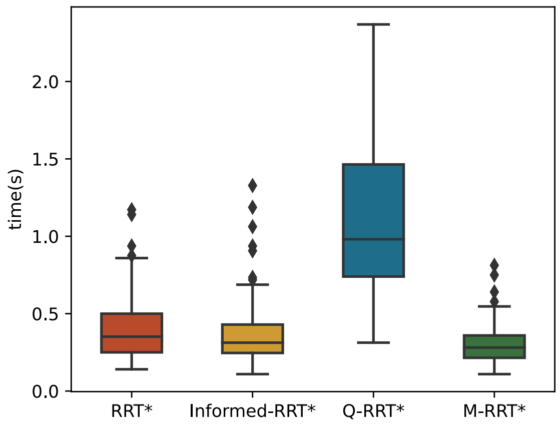

Figure 13. t_{init} in the cluttered environment.

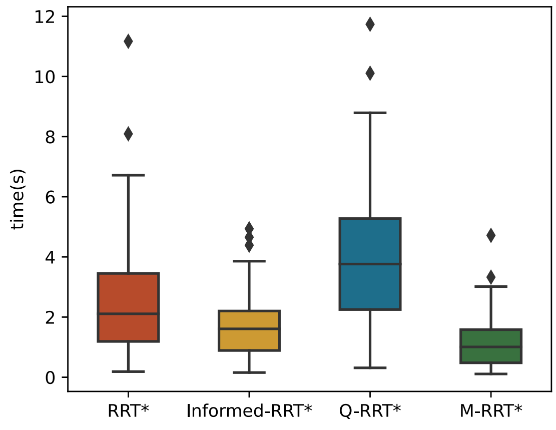

Figure 14. $T_{5\%}$ in the cluttered environment.

4.4. Maze

The maze is shown in Figure 8c. In Figure 15, the paths generated by RRT* (t_{init} = 0.2163, $T_{5\%}$ = 4.1688), Informed-RRT* (t_{init} = 0.375, $T_{5\%}$ = 14.4523), Q-RRT* (t_{init} = 0.8908, $T_{5\%}$ = 1.5375), and M-RRT* (t_{init} = 0.0647, $T_{5\%}$ = 1.3433) are shown. In the maze, $l_{optimal}$ = 138.6089. It can be seen that Informed-RRT* always performed global sampling like RRT*. Thus, Informed-RRT* has no advantage in this environment. Q-RRT* optimized paths more efficiently in the maze. On the contrary, due to the two vertex acceptance criteria proposed in this paper, M-RRT* rejected a lot of inferior vertices and obtained the solution with fewer vertices.

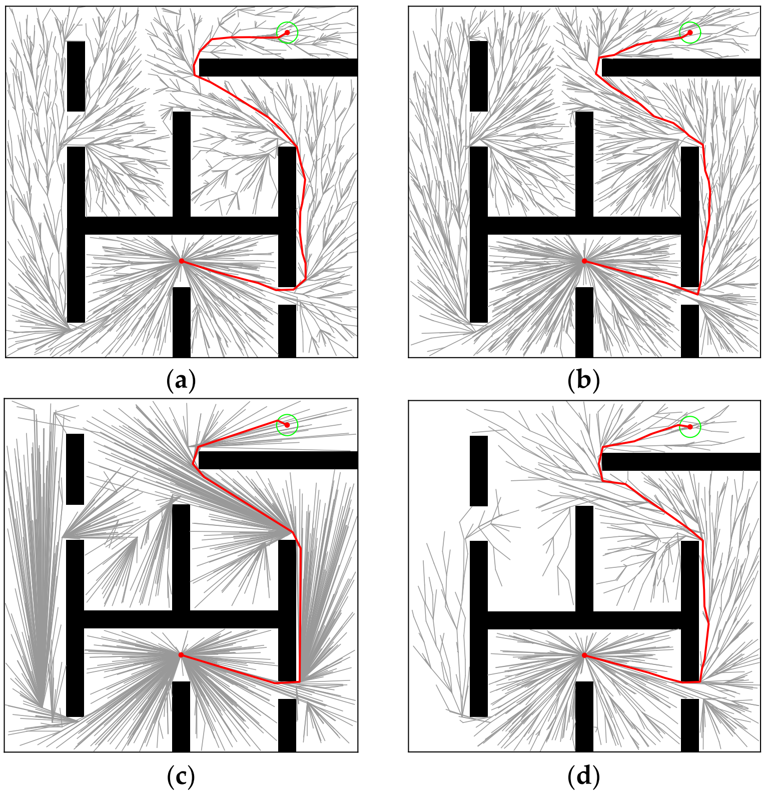

Figure 15. Performance of the four algorithms in the Maze. (**a**) RRT*. (**b**) Informed-RRT*. (**c**) Q-RRT*. (**d**) M-RRT*. (Red line: approximate optimal path, green circle: goal region.).

Figures 16 and 17 show 100 simulation results of t_{init} and $T_{5\%}$. Table 4 counts the specific numerical values of the simulation results. Since maze-type environments tend to trap the algorithm in a local optimum, M-RRT* requires more extensive sampling when estimating the initial solution. It can be seen from Figure 16 that Q-RRT* takes the longest time, and although the medians of the other three algorithms are relatively close, the time needed for M-RRT* to obtain the initial solution is slightly less. As shown in Figure 17, M-RRT* obviously has the fastest convergence rate, while Informed-RRT* has the worst performance. The region of the hyper-ellipsoid set by Informed-RRT* in searching for the optimal solution in this environment is larger than the state space, which causes it to converge even slower than RRT*. The convergence speed of Q-RRT* is also very fast, so this paper indirectly proves that when Q-RRT* has enough space to search for ancestor vertices without collision, its efficiency will be greatly improved. Although RRT* performs worse, the minimum values of its two indicators are the lowest. This is because, in one of the experiments on RRT*, the random vertices of the algorithm found the path very quickly, which is accidental. By combining Figures 16 and 17 and Table 4, it can be seen that M-RRT* performs significantly better in the maze.

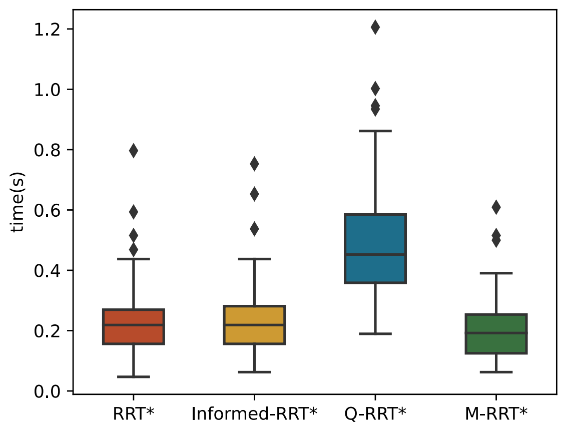

Figure 16. t_{init} in the Maze.

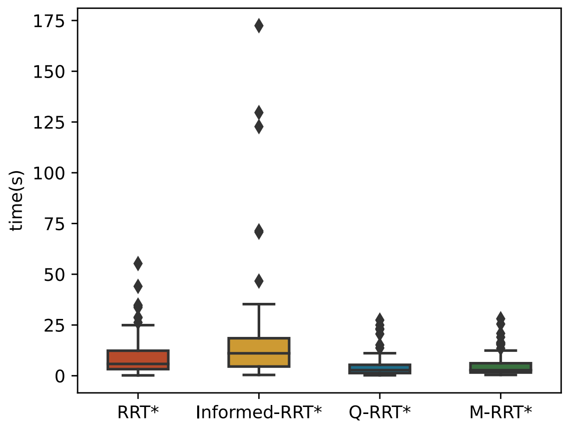

Figure 17. $T_{5\%}$ in the Maze.

Table 4. t_{init} and $T_{5\%}$ of four algorithms in the Maze.

Algorithm		Mean	Median	Min	Max
RRT*	t_{init}	0.2365	0.2186	0.0468	0.7966
	$T_{5\%}$	9.445	5.8267	0.1874	55.2755
Informed-RRT*	t_{init}	0.2375	0.2030	0.06248	0.753
	$T_{5\%}$	17.0028	11.0589	0.406157	172.453
Q-RRT*	t_{init}	0.4876	0.4523	0.1894	1.206
	$T_{5\%}$	4.5829	2.7853	0.2269	27.4085
M-RRT*	t_{init}	0.2062	0.1921	0.06247	0.6092
	$T_{5\%}$	5.091	2.7571	0.406155	28.0715

5. Conclusions

There has been an increase in research on sampling-based motion planning algorithms in recent years. RRT* is a commonly used optimal algorithm, but this method has the problems of a low search efficiency and a slow convergence speed. M-RRT* is proposed to solve the motion planning problem of mobile robots. First, this research proposes an asymptotic vertex acceptance criterion in the initial path estimation phase of M-RRT*, which can effectively reduce the time of finding the initial path and make the algorithm start searching for the optimal path earlier. Secondly, this research proposes a nonlinear dynamic vertices acceptance criterion in the optimal path search phase of M-RRT*. This criterion preferentially accepts vertices that may improve the current path so as to rapidly converge to the optimal path.

Although M-RRT* is a promising algorithm, it can only rely on global sampling to jump out of the local optimum when finding initial solutions in some special environments. Moreover, this paper mainly studies static motion planning, but the actual environment is dynamic, so we should take into account the real-time nature of motion planning in a further work. Since the motion planning of the manipulators has complex constraints, a small number of path points can reduce its calculation. We apply the algorithm proposed in this paper to the motion planning of the manipulator as the next research content.

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

Funding: This work is supported by the National Key R&D Program of China (Grant 2021YFB3301400), the LiaoNing Revitalization Talents Program (Grant XLYC1907057), the Nature Science Foundation of Liaoning province (Grant 2021-MS-030), and the State Key Laboratory of Robotics [2022-Z03].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Fang, T.; Ding, Y. A sampling-based motion planning method for active visual measurement with an industrial robot. *Robot. Comput.-Integr. Manuf.* **2022**, *76*, 102322. [CrossRef]
2. Lu, Y.; Tang, K.; Wang, C. Collision-free and smooth joint motion planning for six-axis industrial robots by redundancy optimization. *Robot. Comput-Integr. Manuf.* **2021**, *68*, 102091. [CrossRef]
3. Zhang, H.; Zhu, Z.; Yuan, J. Non-inverse kinematics of free-floating space robot based on motion planning of sampling. *J. Northwest. Polytech. Univ.* **2021**, *39*, 1005–1011. [CrossRef]
4. Deng, H.; Xin, G.; Zhong, G.; Mistry, M. Gait and trajectory rolling planning and control of hexapod robots for disaster rescue applications. *Robot. Auton. Syst.* **2017**, *95*, 13–24. [CrossRef]
5. Niu, G.; Pan, B.; Fu, Y.; Qu, C. Development of a New Medical Robot System for Minimally Invasive Surgery. *IEEE Access* **2020**, *8*, 144136–144155. [CrossRef]
6. Wang, N.; Zhang, C.; Vahidi, A. Probabilistic anticipation and control in autonomous car following. *IEEE Trans. Control Syst. Technol.* **2017**, *27*, 30–38.
7. 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]
8. Dorigo, M.; Caro, G.D.; Gambardella, L.M. Ant algorithms for discrete optimization. *Artif. Life* **1999**, *5*, 137–172. [CrossRef]
9. Wang, Y.; Han, T.; Jiang, X.; Yan, Y.; Liu, H. Path Planning of Pattern Transfer Based on Dual-Operator and a Dual-Population Ant Colony Algorithm for Digital Mask Projection Lithography. *Entropy* **2020**, *22*, 295. [CrossRef]
10. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]
11. Koren, Y.; Borenstein, J. Potential field methods and their inherent limitations for mobile robot navigation. In Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, CA, USA, 9–11 April 1991.
12. 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]
13. Koenig, S.; Likhachev, M.; Furcy, D. Lifelong planning A*. *Artif. Intell.* **2004**, *155*, 93–146. [CrossRef]
14. LaValle, S.M.; Kuffner, J.J. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
15. 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]
16. Wei, K.; Ren, B. A Method on Dynamic Path Planning for Robotic Manipulator Autonomous Obstacle Avoidance Based on an Improved RRT Algorithm. *Sensors* **2018**, *18*, 71. [CrossRef]
17. Chiang, H.T.L.; Tapia, L. COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2024–2031. [CrossRef]
18. Tang, X.; Chen, F. Robot path planning algorithm based on Bi-RRT and potential field. In Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA), Beijing, China, 13–16 October 2020.

19. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), San Francisco, CA, USA, 24–28 April 2000.
20. Kang, J.G.; Lim, D.W.; Choi, Y.S.; Jang, W.J.; Jung, J.W. Improved RRT-Connect Algorithm Based on Triangular Inequality for Robot Path Planning. *Sensors* **2021**, *21*, 333. [CrossRef]
21. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
22. Xu, J.; Song, K.; Dong, H.; Yan, Y. A batch informed sampling-based algorithm for fast anytime asymptotically-optimal motion planning in cluttered environments. *Expert Syst. Appl.* **2020**, *144*, 113124. [CrossRef]
23. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014.
24. 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]
25. Qi, J.; Yang, H.; Sun, H. MOD-RRT*: A Sampling-Based Algorithm for Robot Path Planning in Dynamic Environment. *IEEE Trans. Ind. Electron.* **2021**, *68*, 7244–7251. [CrossRef]
26. Wang, J.; Li, T.; Li, B.; Meng, M.Q.-H. GMR-RRT*: Sampling-Based Path Planning Using Gaussian Mixture Regression. *IEEE Trans. Intell. Veh.* **2022**, *7*, 690–700. [CrossRef]
27. Guo, J.; Xia, W.; Hu, X.; Ma, H. Feedback RRT* algorithm for UAV path planning in a hostile environment. *Comput. Ind. Eng.* **2022**, *174*, 108771. [CrossRef]