MDPI

Article

RJ-RRT: Improved RRT for Path Planning in Narrow Passages

Qisen Chai [1], Yujun Wang [1]*

[1] No. 2, Tiansheng Road, Beibei District,Southwest University, Chongqing, 400715, China
* Correspondence: wangyjun@swu.edu.cn;

Abstract: As a representative of sampling-based planning algorithms, Rapid Random Exploration Tree (RRT), is extensively welcomed in solving robot path planning problems due to its wide application range and easy addition of nonholonomic constraints. However, it is still challenging for RRT to plan the path for configuration space with narrow passages. As a variant algorithm of RRT, Rapid Random Discovery Vine (RRV) gives a better solution, but when configuration space contains more obstacles instead of narrow passages, RRV performs slightly worse than RRT. In order to solve these problems, this paper reexamines the role of sampling points in RRT. Firstly, according to the state of the random tree expanding towards the current sampling point, a greedy sampling space reduction strategy is proposed, which decreases the redundant expansion of the random tree in space by dynamically changing the sampling space. Secondly, a new narrow passage judgment method is proposed according to the environment around of sampling point. After the narrow passage is identified, the narrow passage is explored by generating multiple subtrees inside the passage. The subtrees can be merged into the main tree that expands in a larger area by subsequent sampling. These improvements further enhance the value of sampling points. Compared with the existing RRT algorithms, the adaptability for different environments is improved, and the planning time and memory usage are saved.

Keywords: path planning; RRT ; narrow passage ;complex enviornment

1. Introduction

With fields like the rise of unmanned aerial vehicles (UAV), autonomous driving, and mobile robots, path planning as an essential part of their technology have long been a research hotspot of scholars [1,2]. The main task of path planning is to find an optimal or sub-optimal path from the initial configuration to the goal configuration in the configuration space according to specific performance criteria, such as path length, planning time, and whether the vehicle kinematics are satisfied.

Among many types of algorithms, the graph-based search method represented by A* [3] and Dijkstra's algorithm [4] divides the configuration space mapped from the real environment into multiple grids according to the size of the resolution and then uses the graph search method to find feasible paths. The resolution size has a decisive impact on this type of algorithm's performance. An overly complex and sizeable real environment often causes an exponential increase in computing time and memory usage [5]. As another important class of planning algorithms, sampling-based planning algorithms represented by RRT [6] and PRM [7] simplify the configuration space by random sampling. Therefore, they do not require complex geometric modelling and perform better in high dimensions or large environments. Probabilistic completeness guarantees that if at least one feasible path exists, the probability of this class of methods finding the path will be 1[8,9].

Although sampling-based methods have many advantages for path planning, the performance of such algorithms still has challenges when the configuration space contains multiple narrow passages[10,11]. As a single-query, RRT plans a path by constructing a random tree extending from the initial configuration to the goal configuration by continuous random sampling. As a multi-query, the PRM needs to generate several random sampling

Citation: Chai, Q.; Wang, Y.RJ-RRT: Improved RRT for Path Planning in Narrow Passages Title. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Submitted to *Journal Not Specified* for possible open access publication under the terms and conditions of the Creative Commons Attri-bution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

points at the preprocess stage and then connect these points to each other to form a connectivity graph, thus capturing the configuration space's connectivity. Due to the special property of random sampling, both RRT and PRM are affected by the volume limitation of free space in the random sampling process. Specifically, the branches generated when RRT constructs a random tree are almost impossible to expand in narrow passages, and PRM is difficult to generate random sampling points in the collision-free configuration space with small volumes, thus affecting the construction of connected graphs. To solve the problem that the sample-based algorithm has insufficient planning ability for narrow passages, scholars have given many solutions. For PRM, in[12] , the author proposes an extended-retract map that makes the sampling points all fall on the central axis of free space. In[13], the author proposed a strategy based on Gaussian sampling. This strategy will increase the sampling points located at the boundary between the obstacle and the free space and eventually generate many sampling points around the obstacle. However, it is not considered that the sampling point is around the obstacle but not necessarily inside the narrow passage. In[14], the author proposes a bridge connection test method to detect narrow passages. In this method, a bridge is built by sampling points located on two different obstacles. If the midpoint of the bridge is in free space, it means that it has passed the bridge test. Obviously, narrow passages make it easier to build such Bridges. In[15], the author uses an artificial potential field to make the sampling point located on the obstacle move to free space under the action of repulsive force. In[16], the authors coordinately map both the obstacle space and the free space at the same time, increasing the connectivity of the graph inversely through failed connections.

For RRT, since RRT is more goal-oriented than PRM, it is planning a path while constructing a tree, which makes the improvement of RRT have greater potential. In[17], the authors propose a Voronoi bias-based planner and improve the sampling strategy according to the visibility of the regions by nodes in the tree. In[18], the authors regress the sampling points located on the obstacle so that the tree can expand around the edge of the obstacle, thus ensuring the exploration inside the passages. In[19], the authors use a simple heuristic to generate additional subtrees in the configuration space and control the expansion of the subtrees through parameters. In[20], a modified bridge test is used to identify narrow passages and relies on multiple subtrees expand from narrow passages to explore narrow passages. In [21] the author balances the exploration of global and local random trees through Markov chains. In addition, in[22], the authors use PRM globally and rely on special sampling points to activate RRT and use RRT to explore the complex environment. Based on this idea, RRT can be combined with various variants of PRM. In[23], the author subdivided the narrow passage into passage entrance, passage interior, and obstacle edge by principal component analysis (PCA) so random trees can climb obstacles in different environments like vines to expand. This method has achieved good results in the environment with narrow passages. However, due to its dependence on the environment, the performance is slightly worse than RRT in an environment with many obstacles but no narrow passages.

In addition, when solving the problem of narrow passages, the above methods mostly ignore the randomness of whole configuration space sampling, which often leads to part of sampling points in space that are not helpful for the final path construction so that increasing the total planning time and wasting memory resources. In this paper, we propose a new planning algorithm based on the RRT named Reduce-Judge RRT (RJ-RRT). Specifically, the new algorithm includes two improvements: Firstly, based on the greedy idea, a sampling strategy is proposed to reduce the sampling space to the goal area gradually. Secondly, a new method of environment judgment is proposed, when a narrow passage is identified, it can be explored by subtrees expansion inside the passages. The improved environment judgment method only needs a straightforward calculation to determine the environment around the current sampling point. Both strategies improve planning efficiency. On the one hand, random sampling points are used to reduce the sampling space to avoid random tree expansion in space that does not help with path planning. On the other hand, random

sampling points are used to detect the surrounding environment, and the subtree of narrow passages helps the random tree expand in the configuration space that is difficult to explore. The results show that the algorithm has good adaptability to various environments, and can greatly reduce the planning time and memory usage.

The rest of this paper is outlined as follows. Section 2 formally defines the path planning problem and introduces the preconditions of the proposed algorithm. Section 3 explains the RJ-RRT method proposed in this paper. Section 4 provides the results and evaluation of the simulation experiments. Section 5 concludes this paper and looks forward to future work.

2. Background

In this section, we formally define the problem that path planning needs to solve and introduce algorithms such as Basic-RRT.

2.1. Problem Definition

Let $X \in \mathbb{R}^d$ be the the configuration space of the path planning problem,where $d \in \mathbb{N}$, $d \geq 2$. Let $X_{obs} \in X$ be the obstacle space, which is an impassable space, and denotes the obstacle-free space as $X_{free} = X \setminus X_{obs}$, which is a passable space. $x_{init} \in X_{free}$ and $x_{goal} \in X_{free}$ are the initial configuration and the goal configuration, respectively, and the goal region is defined as a circle with radius r: $X_{goal} = \left\{ x \in X \middle| ||x - x_{goal}|| < r \right\}$. A path is defined by continuous function $\sigma : [0, T] \to X_{free}$, and $\sigma(0) = x_{init}$, $\sigma(T) = x_{goal}$. If the path is feasible, then $\sigma(\tau) \in X_{free}$ for all $\tau \in [0, T]$.

The path planning problem is to find a feasible path. Problem 1 defines the feasibility problem of path planning.

Problem 1(Feasible path Planning) Given a configuration space of planning problem $X \in \mathbb{R}^d$, a free space X_{free}, an initial state $x_{init} \in X_{free}$, and a goal region $X_{goal} \in X_{free}$, find a path $\sigma : [0 : T] \to X_{free}$ if one exists. If no such feasible path exists, return failure.

The cost from the initial configuration to the goal region is not necessarily the same each time planning or on each path. Let $c(\cdot)$ be the cost function. The optimization problem of path planning is to find the path with minimum (non-negative) path cost. Its formal expression is given by problem 2.

Problem 2 (Optimal Path Planning) Given a configuration space of planning problem $X \in \mathbb{R}^d$, a free space X_{free}, an initial state $x_{init} \in X_{free}$, a goal region $X_{goal} \in X_{free}$ and cost function c. Find a feasible path σ^* such that $c(\sigma^*) = min\left\{ c(\sigma) : \sigma \in \sum_{feasible} \right\}$.

The path planning algorithm usually takes some time to plan the path. Let $t \in \mathbb{R}$ be the set of times it takes to find a set of paths. Problem 3 defines finding the best path in Problem 2 in the least amount of time possible.

Problem 3 (Fast Path Planning) Minimum time $t \in \mathbb{R}$ required to plan the Optimal Path.

2.2. Basic-RRT

The Basic-RRT algorithm is the basis of RJ-RRT, an underlying tree data structure is maintained in Basic-RRT. As shown in Figure1. Firstly, Basic-RRT creates a tree rooted at the x_{init}. In each subsequent iteration, x_{rand} is obtained by randomly sampling in the configuration space X, traversing the existing nodes in the tree, selecting the node x_{near} closest to x_{rand}, and generating x_{new} through the steering function with a certain expand size. If the edge $\{x_{near}, x_{new}\}$ is not located in X_{obs}, then x_{new} is added to the tree with x_{near} as the parent node, and the edge $\{x_{near}, x_{new}\}$ will be recorded. When x_{new} is within the range of X_{goal}, the planning is ended, and the tree is returned. The return fails when the timeout or the number of iterations is exceeded. The Basic-RRT algorithm is described in Algorithm1.

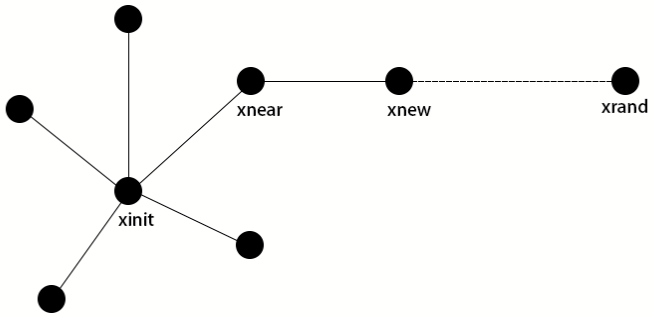


Figure 1. Expand of Basic-RRT.

Algorithm 1 Basie-RRT

$Tree \leftarrow x_{init}$
for $i = 1$ to n **do**
  $x_{rand} \leftarrow Sample()$
  $x_{near} \leftarrow Nearest()$
  $x_{new} \leftarrow Steer(x_{rand}, x_{neare})$
  if $FreeCollision(x_{rand}, x_{near})$ **then**
    $Tree.addEdge(x_{near}, x_{new})$
    $Tree.addNode(x_{new})$
    if $x_{new} \in X_{goal}$ **then**
      return $Tree$
    end if
  end if
end for

RRT-Biased [24] is an efficient method to improve RRT. By the premise of a certain probability, the x_{goal} is set as the sampling point so that the tree can expands faster to the goal area. In Algorithm2, given a brief description of RRT-Biased.

Algorithm 2 Biased-RRT

$prand \leftarrow RandomNumber()$
if $prand \leq K$ **then**
  $x_{rand} = x_{goal}$
else
  $x_{rand} \leftarrow Sample()$
end if

In this paper, the tree is rooted at x_{init} denoted as main tree. The rest of the trees denoted as subtrees.

3. RJ-RRT

In the Basic - RRT algorithm, the selection of random sampling points is in the entire configuration space or can be described as blind, such as in a certain sampling, the current sampling point is likely very close to the goal region, however, due to deviation of the next sampling, the random tree will likely expand to a region that does not have a facilitation effect on the final path generation. RRT-Biased, although through a forced modification make the random tree is biased expand towards the goal region, when encountering a trap or a complex environment, it falls into the same predicament as Basic-RRT. And the role of sampling points for Basic-RRT is to guide the expansion direction of the random tree. If the random tree can quickly approach the goal region, it needs to expand toward the goal region while avoiding obstacles. This is also a requirement for an efficient planner can generate

sampling points. To achieve this efficacy and enhance the value of sampling points,in RJ-RRT, sampling space greedy reduction and environment judgement are performed through sampling points, and subtrees are expand in special environments. The overall framework of the RJ-RRT is given by Algorithm3 and in the subsequent sections detailed description of each improvement.

Algorithm 3 RJ-RRT

$MainTree \leftarrow x_{init}$
for $i = 1$ to n **do**
  $x_{rand} \leftarrow GreedSpaceSampling()$
  $SubTrees \leftarrow EnvironmentalJudgment(x_{rand})$
  if $FreeCollision(x_{rand}, x_{near})$ **then**
    $MainTree.addEdge(x_{near}, x_{new})$
    $MainTree.addNode(x_{new})$
    if $x_{new} \in X_{goal}$ **then**
      return $MainTree$
    end if
    $SubTreeExpand()$
    $SubTreeMerge()$
  end if
end for

3.1. Sampling Space Greedy Reduction Strategy

When there is only the root node x_{init} in the tree, random sampling is selected in the entire configuration space X. For ease of description, the configuration space is denoted as the first $OriginalSpace$. When a node x_{new} is successfully added to the tree, the $OriginalSpace$ will be reduced. The specific process is to take x_{new} as the boundary point of the new sampling space and use the boundary point as the benchmark to divide the $OriginalSpace$ in each dimension to obtain a subspace containing the goal region. This subspace will be used as the space for the next random sampling and denoted as $NextSpace$. In the next sampling, this $NextSpace$ will replaced by $OriginalSpace$. At this time, the sampling space completes a reduction. After each sampling space is reduced, record the $GapSpace$ between $NextSpace$ and the $OriginalSpace$, where $GapSpace$ is defined as Equation(1):

$$GapSpace = OriginalSpace \setminus (OriginalSpace \cap NextSpace) \tag{1}$$

As the number of space reductions increases, will generate N $GapSpace$, denoted as $Gap_1, Gap_2 \ldots Gap_N$. Ideally, the sampling space will gradually approach the goal region as the number of sampling increases, but this situation is often not practical. When the random tree expansion with random sampling fails in the $OriginalSpace$, it means that the current space has failed to be reduced, and the space to sample will fall back to the $GapSpace$ (Gap_N) nearest to the $OriginalSpace$. Sampling in the $GapSpace$ can be multiple times to provide more node expansion options for the next space reduction. When the specified sampling times are reached in the $GapSpace$, forward to the space where the space reduction failed, that is, the $OriginalSpace$ for sampling. If the expansion is successful after sampling, the random tree continues to expand according to the above reduction strategy. If the expansion fails, fall back to the previous $GapSpace$ (Gap_{N-1}) sampling of the $GapSpace$ (Gap_N). After sampling multiple times, forward to the $GapSpace$ (Gap_N), sample again, and finally, forward to the $OriginalSpace$ for sampling. If the reduction still fails, continue to fall back to sampling in more outer $GapSpace$, and then sample in the next $GapSpace$ of each $GapSpace$ in turn. When fall back to the first $GapSpace$ sampling, namely Gap_1, the random tree generated by RJ-RRT in the configuration space is like the Basic- RRT. The process is shown in Algorithm4.

Algorithm 4 GreedSpaceSampling

$OriginalSpace = ConfigurationSpace$
$x_{rand} \leftarrow Sample(OriginalSpace)$
if $FreeCollision(x_{rand}, x_{near})$ **then**
  $NextSpace \leftarrow Reduce(x_{rand}, X_{goal})$
  $GapSpace \leftarrow Reduce(OriginalSpace, NextSpace)$
  $OriginalSpace = NextSpace$
else
  $x_{rand} \leftarrow BackwithForwardSample(GapSpace)$
end if

Take Figure2 as an example. Firstly, A is the configuration space and the first $OriginalSpace$. After the sampling and steering function, the newly added node is $P1$. The sampling space is reduced as the random tree expands, through $P1$ formed new sampling space B. Sampling in B to get node $P2$, the space is further reduced to get sampling space C, sampling in C, random tree expansion is difficult, indicating that the current space cannot be further reduced, fall back to the $GapSpace$ between B and C for sampling, and after multiple samplings, forward to sample in C, the space reduce still fail, come back to sampling in the $GapSpace$ between A and B to sample multiple times, forward again to sampling in the $GapSpace$ between B and C, and finally forward to sampling and expanding in C.

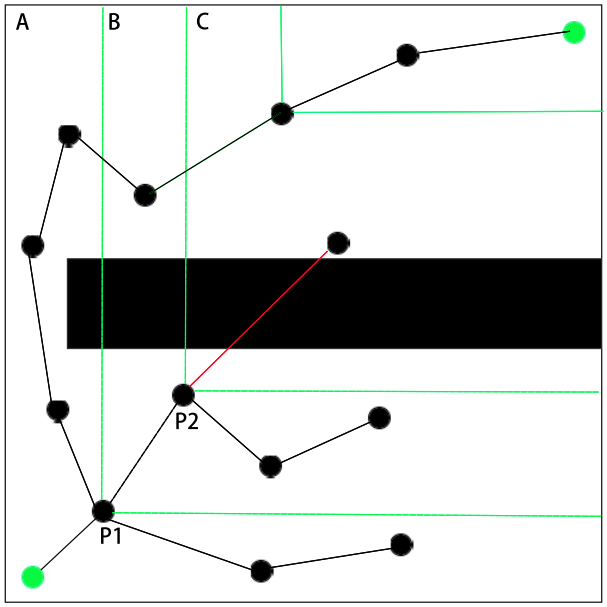


Figure 2. The sampling space A is reduced to B through P1 and then reduced to C through P2. The expansion fails in C. And first fall back sampling and then forward sampling.

3.2. Environmental Judgment

The main purpose of environmental judgment is to identify narrow passages and set the expansion space of subtrees. Environmental judgment will mainly occur during part of the sampling process, and special cases where environmental judgment is not performed will be described in the next section.

When randomly generating a sampling point x_{rand} in the sampling space, the environment around x_{rand} will be judged. As shown in Figure 3.Firstly, if x_{rand} is located in X_{free}, the subsequent judgment will be omitted. If x_{rand} is located in X_{obs}, then in the circle with x_{rand} as the center and $R1$ as the radius, $N1$ sampling points are randomly generated and determine the space where these sampling points are located. If there are sample points lo-

cated in X_{free}, it can be inferred that the x_{rand} is close to the edge of the obstacle. Randomly pick a sampling point in X_{free}, denoted as x_{cir}, x_{cir} will be the root of the subtree. Divide the circle with x_{cir} as the center and $R2$ as the radius into eight equal parts and get eight marked points on the boundary of the circle, denoted as $M1, M2, M3, M4, M5, M6, M7, M8$. Through the location of the marked points, it is further identified whether the current surrounding environment of x_{cir} is the inside of the narrow passage or the entrance of the narrow passage.The process of judging the surrounding environment of the sampling point is given by Algorithm5.

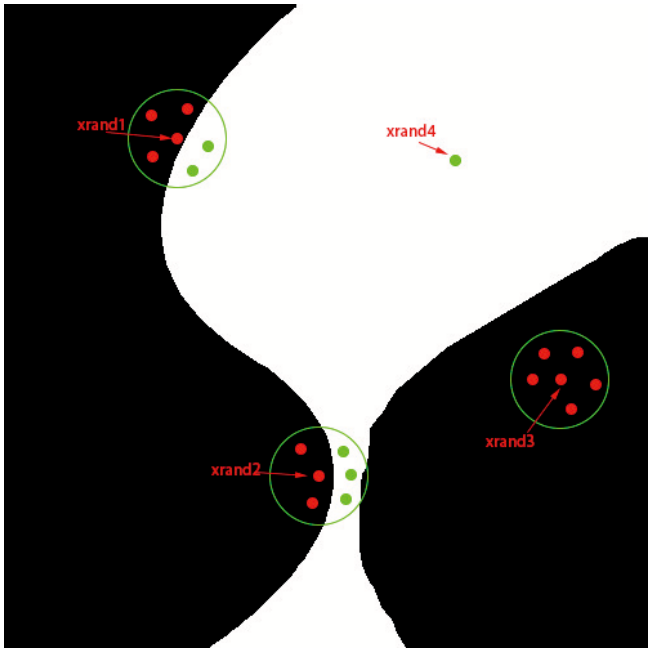


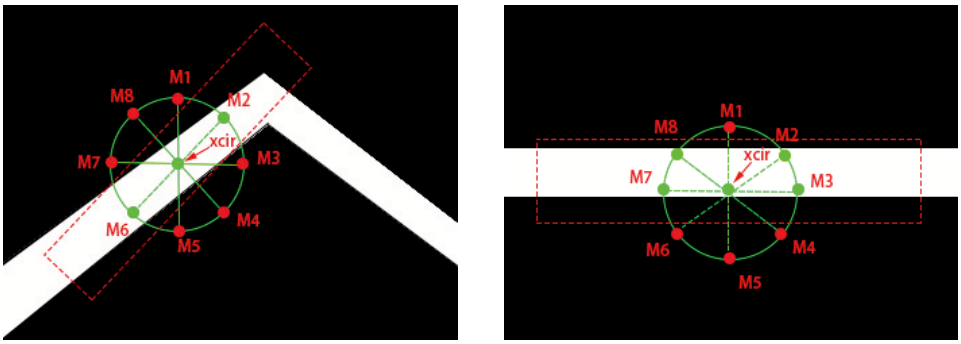
Figure 3. Locally sample around *xrand*. *xrand*1, *xrand*2, *xrand*3 and *xrand*4 show four different cases, respectively. When there are sampling points in the free space in the local sampling, such as *xrand*1, and *xrand*2, then the surrounding environment of *xrand* is further judgments, *xrand*3 and *xrand*4 will not make further judgments

Algorithm 5 EnvironmentalJudgment

$x_{cir} \leftarrow LocalSample(x_{rand})$
$MarkedPoints \leftarrow LocalJudgement(x_{cir})$
$SubTreeExpandSpace \leftarrow LocalJudgement(MarkedPoints)$

Inside the narrow passage: As shown in Figure 4, when two or four marked points are located in X_{free}, the remaining marked points are located in X_{obs}, and where each marked point located in free space is matched with another free point with an angle of 180. Then inferred that x_{cir} is located inside the narrow passage. In this case, the constructed subtree expansion space is a rectangle (If the configuration space is 3D, will be a cube). The center of the rectangle is at x_{cir}, where the long side is parallel to the line formed by the 180 marked points in free space, and the length of the side is $L1$. The other short side of the rectangle is $L2$. This stretched rectangle constitutes the sampling space of the subtree.

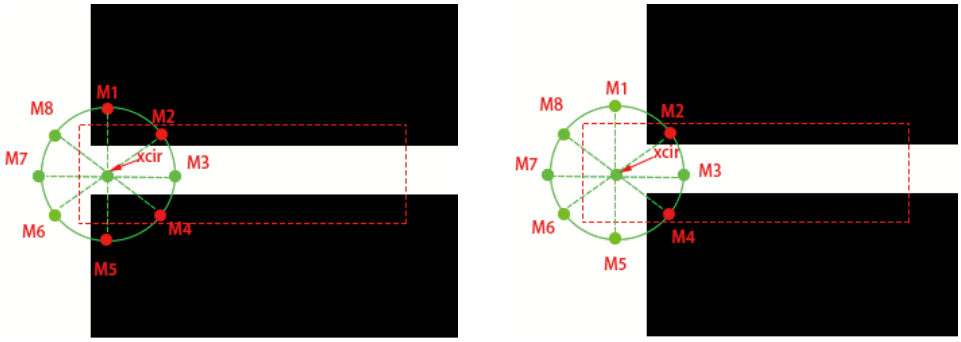
Entrance of the narrow passage: When the marked points are distributed similarly to Figure 5, that is, there are three or four consecutive marked points and an additional isolated marked point in the free space, while the adjacent to the isolated marked point or remaining marked points are located in the obstacle space then it is inferred that x_{cir} is located at the entrance of the narrow passage, and isolated marked point is used to identify narrow passage entrance and guide exploration. The constructed subtree expansion space



(a) (b)

Figure 4. x_{cir} is inside the narrow passage

is like that inside the narrow space. The difference is that the stretching of the subtree expansion space is only carried out in the same direction as the isolated marked point.



(a) (b)

Figure 5. x_{cir} is in the entrance of the narrow passage

The unrestricted generation of subtrees is prevented by judging whether the x_{cir} is already within the expansion interval of some subtree. In addition, the over-exploration phenomenon caused by overlapping sampling Spaces of subtrees is discussed in the next section.

3.3. Subtree Expansion and Merge

3.3.1. Subtree Expansion

When judging that x_{cir} is located inside a narrow passage or at the entrance, take x_{cir} as the root of the subtree, randomly sample and expand in the constructed subtree expansion space, and the expansion method is the same as that of Basic-RRT. Figure6 shows the flow of subtree expansion. In addition, the expansion of the subtree will have a pre-expansion process, firstly sampling N2 times in the subtree expansion space. These sampling points do not make environmental judgments, and the external main tree does not expand with these sampling points. For other subtrees, if the distance between the x_{near} and x_{rand} is less than $D1$, it indicates that these subtrees may be very close, and these subtrees will expand with this sampling. Then when this space sampling ends, it will return to the $OriginalSpace$ for sampling, and both the main tree and the subtrees will expand with each sampling. However the space reduction only happens in the expansion of the main tree.

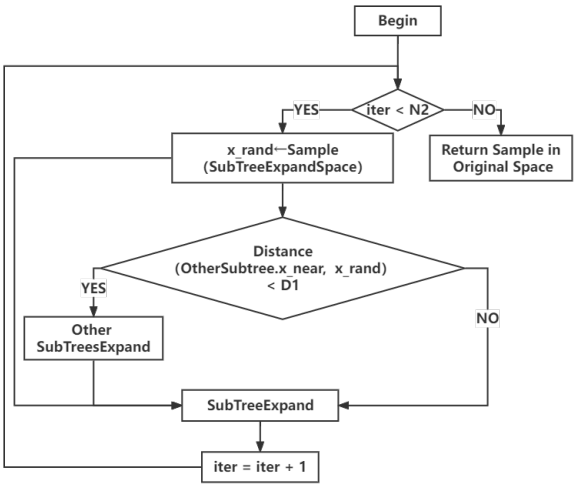


Figure 6. Schematic of SubTree Expansion

3.3.2. Subtree Merge

Subtree merge can be divided into subtree with subtree and subtree with main tree merge.Subtree merging is given by Algorithm6 .

When sampling in $OriginalSpace$, if the distance between the nearest node is less than $D2$ after the main tree and the subtrees expand once, the subtrees are merged into the main tree. When sampling in the $OriginalSpace$ or the subtree expansion space, if there are two or more subtrees after expanding once, the distance between the nearest nodes is less than $D2$, then the multiple subtrees are merged into one subtree. This method not only merges multiple subtrees but also avoids the over-exploration of narrow passages caused by overlapping subtree expansion spaces.

The premise of merging multiple random trees is that there is no collision between the nearest nodes.

Algorithm 6 Subtree Merge

if $x_{rand} \leftarrow Sample()$ **then**
  $MainTreeExpand()$
  $SubTreeExpand()$
  if $Distance(MainTree, SubTrees) < D2$ **then**
    $MainTree = (MainTree \cup SubTree)$
  end if
  if $Distance(SubTree, OtherSubTree) < D2$ **then**
    $SubTree = (SubTree \cup OtherSubTree)$
  end if
end if

4. Simulation and Result

In this section, to evaluate the performance of the proposed algorithm, we compare RJ-RRT with Basic-RRT and RRV through a series of experiments. We conduct simulation experiments in three different sets of environments. As shown in Figure 7, three representative maps are designed in two-dimensional space: complex environment (containing many obstacles), narrow passage, and bug trap environment. The map size of each environment is 10*10. The initial configuration in the first two groups is $[0.5, 0.5]$, the last group's initial configuration is$[2, 3.5]$, and the goal configuration uniformly is$[9.8, 9.8]$. The black area represents obstacles, the green line represents the expansion state of the random tree, and the red line represents the final planned path. Each group of experimental tests 50 times. Compare the average planning time of these planning algorithms, using the number of

collision detections and the final number of nodes generated. And the occupied memory size is reflected by the total number of nodes. The number of collision detections for RRV and RJ-RRT includes additional nodes required for environmental judgment. All-time units are seconds. All algorithms use GJK for collision detection, and the maximum number of iterations is 50,000. To speed up the planning, the three algorithms uniformly set the sampling probability in the goal space as 0.1. The expand size in the first two sets of experiments was 0.1. In our RJ-RRT, R1 is set to 0.5, N1 is set to 15, R2 is set to 0.7, L1 is set to 3, L2 is set to 1.5, and D1 and D2 are set to 0.7 and 0.5 respectively. All experiments were run on Windows 10 with Intel I7-7700, 3.6Ghz 8GB RAM processor, and using MATLAB 2019b as a software platform.

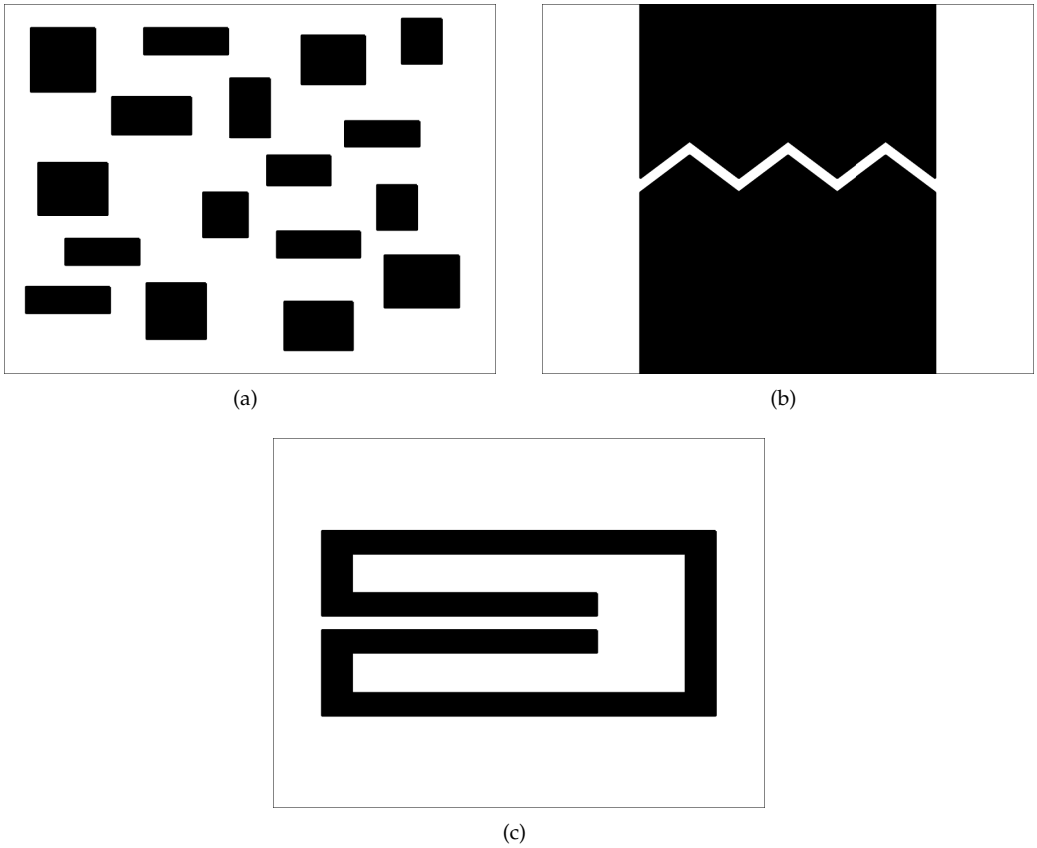


Figure 7. Three different maps in the experiment.(a)Complex;(b)Narrow passage;(c)Bug Trap

4.1. Complex Environment

Figure8 shows the performance of the three algorithms in complex environment. When testing the Basic-RRT algorithm, due to random sampling in the entire configuration space, the final generated random tree occupies almost the entire space. However, although the final tree generated by the RRV algorithm produces fewer nodes than RRT, it still expands in a lot of useless space, and PCA takes a lot of time to calculate. The result is that the average planning time of the RRV algorithm is 58% higher than Basic-RRT. RJ-RRT performs the best in this environment, not only the final random tree has few redundant nodes, but also the planning time is the shortest. The specific statistical results of the three algorithms are shown in Table1. It can be seen that with the setting of the maximum number of iterations, the three algorithms can find the path 100% in the test. In terms of planning time, RJ-RRT saves 87% compared to Basic-RRT, and 92% compared to RRV. The memory consumption of the generating node is 96% lower than that of Basic-RRT and 94% lower than that of RRV.

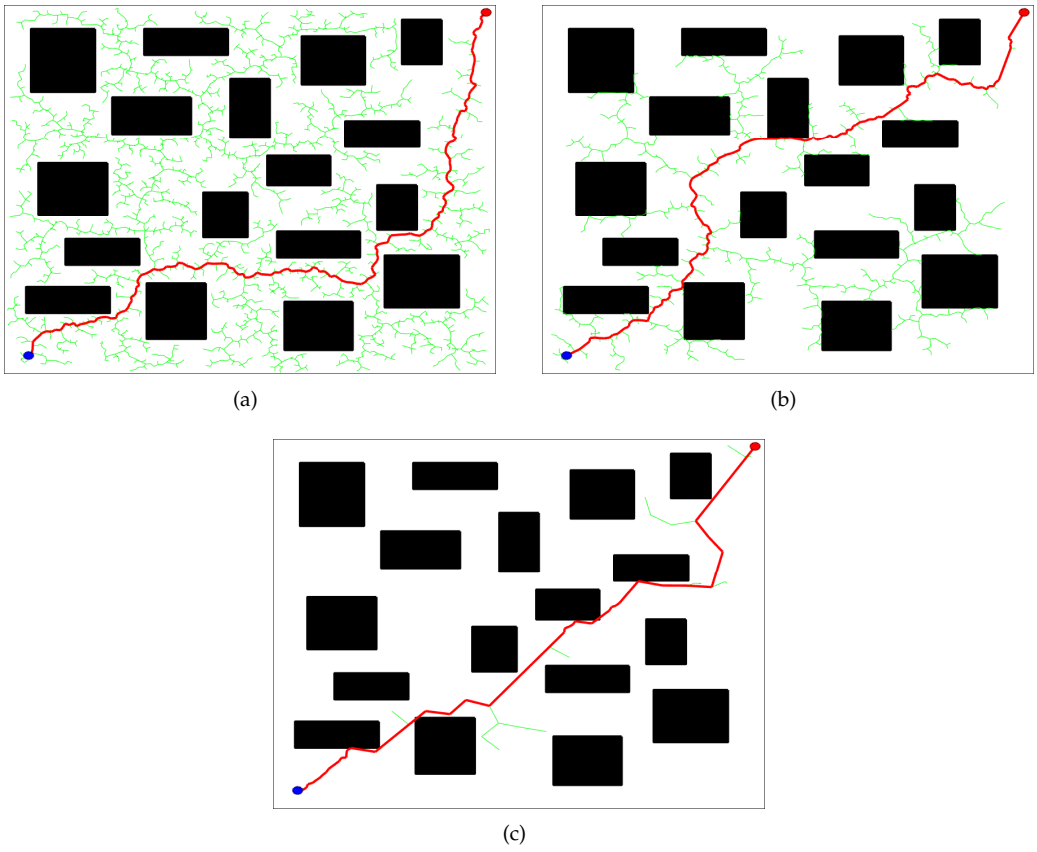


Figure 8. The performance of the three algorithms in complex environments.(a)RRT;(b)RRV;(c)RJ-RRT

Table 1. Results of three algorithms in complex environment.

Algorithm	Average Time	Min Time	Max Time	Average Number of Collisions Detections	Average Number of Nodes
RRT	1.982	0.452	2.741	2338	1542
RRV	3.143	1.310	5.593	2975	1094
RJ-RRT	0.239	0.127	0.874	103	55

4.2. Narrow Passage

Figure9 shows the planning performance of the three algorithms in a narrow passage environment. The experimental results show that when Basic-RRT faces a configuration space containing narrow passages, it is not only difficult to find narrow passage entrances but also difficult to expand inside passages, so successful path planning usually requires a large number of sampling points and more time. From Table2, it can be concluded that in this environment, both RRV and RJ-RRT perform better than RRT, while RJ-RRT requires the least amount of time and takes the least number of samples, on average 1.276s, and generates an average of 227 sampling nodes, while RRV planning takes an average of 2.338s and generates an average of 1158 sampling points. Therefore, RJ-RRT still has a good effect in the environment of narrow passages.

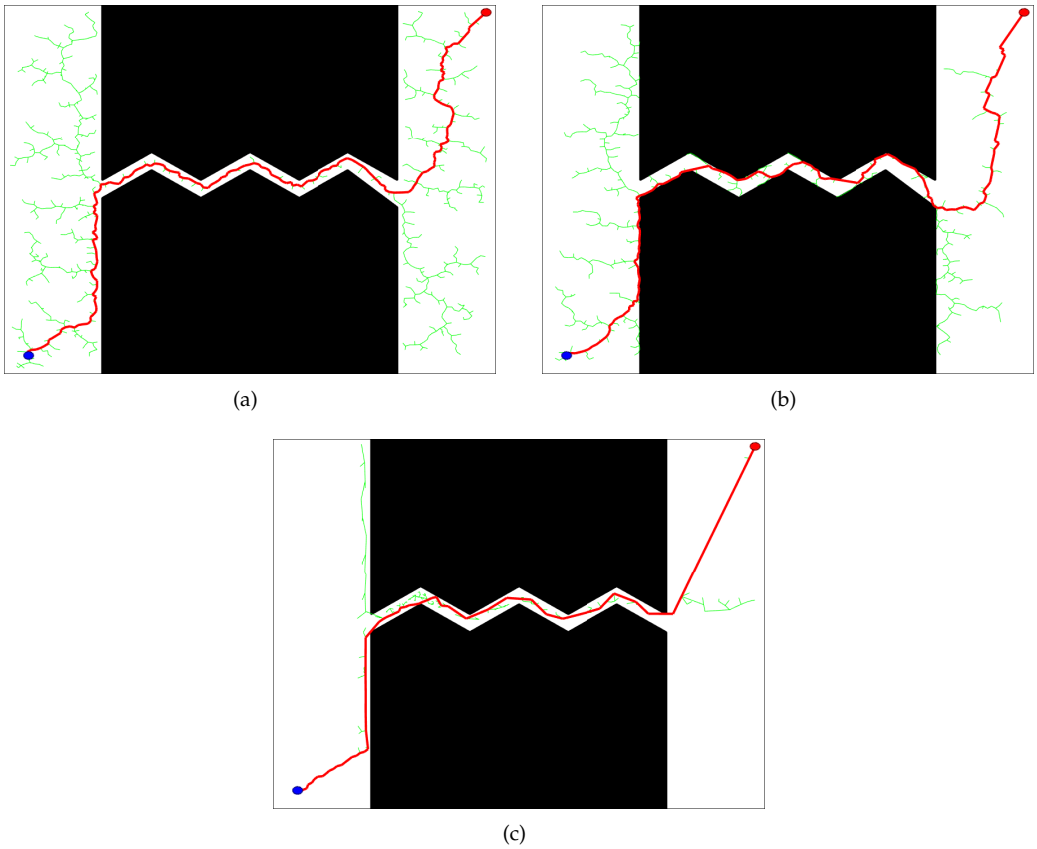


Figure 9. The performance of the three algorithms in narrow passage.(a)RRT;(b)RRV;(c)RJ-RRT

Table 2. Results of three algorithms in narrow passage.

Algorithm	Average Time	Min Time	Max Time	Average Number of Collisions Detections	Average Number of Nodes
RRT	4.937	1.575	10.489	7648	1347
RRV	2.338	1.031	5.944	3327	958
RJ-RRT	1.276	0.475	1.749	678	227

4.3. Bug Trap

The expanded size also affects the number of iterations and memory usage of the RRT and RRT variants. In RJ-RRT, the magnitude of each reduction in the sampling space is also determined by the size of the random tree expansion. Therefore, to analyze the influence of different expand sizes on the performance of different algorithms, we use 0.1/0.3 as the expand size for discussion in this experiment. Table3 and Table4 are the experimental data results when the expand size is 0.1 and 0.3, respectively, and Figure 10 shows the planning performance when the expand size is 0.3. It can be seen from Tables3 and4 that RJ-RRT and RRV perform better than RRT in various performance aspects under different expand size settings. When the expand size is 0.1, RJ-RRT is higher than RRV in generating the total number of nodes. This is because the environment in this experimental, the long and narrow trap passage and short expand size lead to multiple *GapSpaces* in the RJ-RRT planning process, resulting need for multiple fallback and forward operations. These processes increase the number of samples and planning time, so the final performance is not as good as RRV. When the expand size is set to 0.3, since the expand size increase reduces the number of Gap Spaces, RJ-RRT can still plan a path using a smaller number of nodes and a faster time.

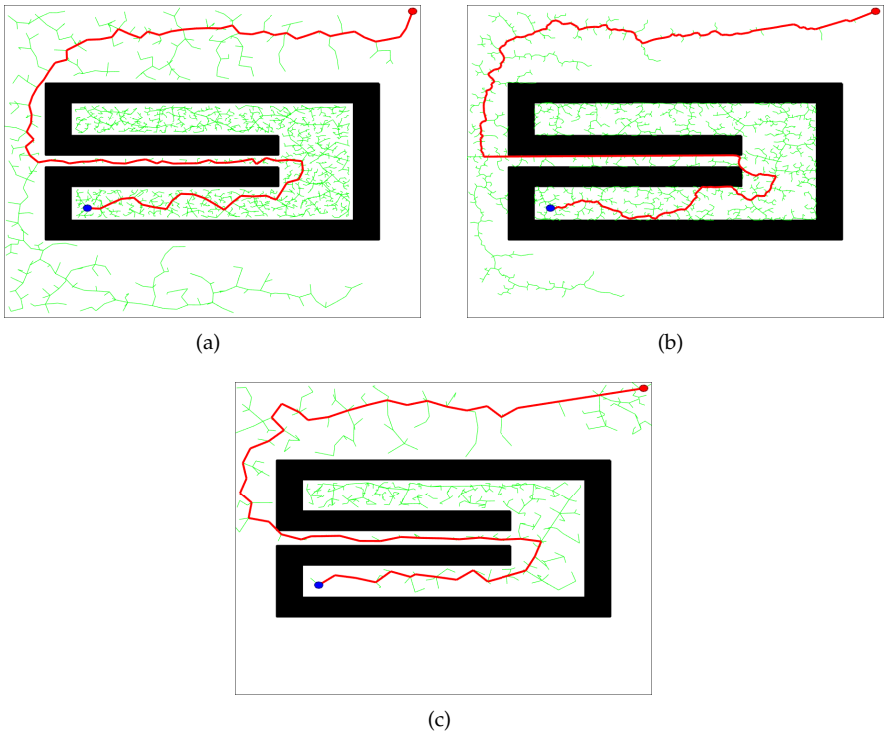


Figure 10. The performance of the three algorithms in bug trap.(a)RRT;(b)RRV;(c)RJ-RRT

Table 3. When extended size is **0.1**, the results of three algorithms in the bug trap.

Algorithm	Average Time	Min Time	Max Time	Average Number of Collisions Detections	Average Number of Nodes
RRT	9.846	6.959	17.707	14596	5579
RRV	5.575	4.945	13.689	5920	2861
RJ-RRT	7.194	5.746	15.145	6218	3347

Table 4. When extended size is **0.3**, the results of three algorithms in the bug trap.

Algorithm	Average Time	Min Time	Max Time	Average Number of Collisions Detections	Average Number of Nodes
RRT	7.912	5.457	14.126	9829	2361
RRV	4.587	3.746	10.578	5124	1741
RJ-RRT	3.348	1.841	6.432	2546	967

5. Conclusions

In this paper, we propose a new planning algorithm RJ-RRT based on the RRT algorithm to solve the planning problem in the narrow passage. A new greedy sampling space reduction and environment judgment strategy are proposed in RJ-RRT. While accelerating the expansion of random tree toward the goal area and reducing redundant nodes, it can identify narrow passages and rely on the subtrees expend inside to explore passages.

Compared with RRT and RRV algorithms, RJ-RRT has better adaptability to different environments and maintains a faster planning speed. In addition, another advantage of RJ-RRT is that easy to combine with other RRT variant algorithms. However, the RJ-RRT still has limitations. For example, is easily affected by the expand size. Therefore, in future research, consider combining adaptive expand size and selectively merging multiple Gap

Spaces, while RJ-RRT should also be extended to 3D environments and incorporate robot kinematics, making our algorithm closer to practical applications.

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

Funding: This research was funded by Chongqing science and Technology Bureau(cstc2017zdcy-zdyfX0042).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

RRT	Rapidly-exploring Random Tree
PRM	Probabilistic Roadmap Method
RRV	Rapidly-exploring Random Vines
RJ-RRT	Reduce-Judge RRT

References

1. Tsardoulias, E.G.; Iliakopoulou, A.; Kargakos, A.; Petrou, L. A review of global path planning methods for occupancy grid maps regardless of obstacle density. *Journal of Intelligent & Robotic Systems* **2016**, *84*, 829–858.
2. Jaillet, L.; Cortés, J.; Siméon, T. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics* **2010**, *26*, 635–646.
3. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* **1968**, *4*, 100–107.
4. Dijkstra, E.W.; et al. A note on two problems in connexion with graphs. *Numerische mathematik* **1959**, *1*, 269–271.
5. Hegedűs, T.; Németh, B.; Gáspár, P. Design of a low-complexity graph-based motion-planning algorithm for autonomous vehicles. *Applied Sciences* **2020**, *10*, 7716.
6. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* **1996**, *12*, 566–580.
7. LaValle, S.M.; et al. Rapidly-exploring random trees: A new tool for path planning **1998**.
8. LaValle, S.M. *Planning algorithms*; Cambridge university press, 2006.
9. Elbanhawi, M.; Simic, M. Sampling-based robot motion planning: A review. *Ieee access* **2014**, *2*, 56–77.
10. Hsu, D.; Latombe, J.C.; Motwani, R. Path planning in expansive configuration spaces. In Proceedings of the Proceedings of International Conference on Robotics and Automation. IEEE, 1997, Vol. 3, pp. 2719–2726.
11. Hsu, D.; Kavraki, L.E.; Latombe, J.C.; Motwani, R.; Sorkin, S.; et al. On finding narrow passages with probabilistic roadmap planners. In Proceedings of the Robotics: the algorithmic perspective: 1998 workshop on the algorithmic foundations of robotics, 1998, pp. 141–154.
12. Wilmarth, S.A.; Amato, N.M.; Stiller, P.F. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In Proceedings of the Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C). IEEE, 1999, Vol. 2, pp. 1024–1031.
13. Boor, V.; Overmars, M.H.; Van Der Stappen, A.F. The Gaussian sampling strategy for probabilistic roadmap planners. In Proceedings of the Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C). IEEE, 1999, Vol. 2, pp. 1018–1023.
14. Hsu, D.; Jiang, T.; Reif, J.; Sun, Z. The bridge test for sampling narrow passages with probabilistic roadmap planners. In Proceedings of the 2003 IEEE international conference on robotics and automation (cat. no. 03CH37422). IEEE, 2003, Vol. 3, pp. 4420–4426.
15. Aarno, D.; Kragic, D.; Christensen, H.I. Artificial potential biased probabilistic roadmap method. In Proceedings of the IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004. IEEE, 2004, Vol. 1, pp. 461–466.
16. Denny, J.; Amato, N.M. Toggle PRM: Simultaneous mapping of C-free and C-obstacle-a study in 2D. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2011, pp. 2632–2639.

17. Yershova, A.; Jaillet, L.; Siméon, T.; LaValle, S.M. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In Proceedings of the Proceedings of the 2005 IEEE international conference on robotics and automation. IEEE, 2005, pp. 3856–3861.
18. Zhang, L.; Manocha, D. An efficient retraction-based RRT planner. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation. IEEE, 2008, pp. 3743–3750.
19. Strandberg, M. Augmenting RRT-planners with local trees. In Proceedings of the IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004. IEEE, 2004, Vol. 4, pp. 3258–3262.
20. Wang, W.; Li, Y.; Xu, X.; Yang, S.X. An adaptive roadmap guided Multi-RRTs strategy for single query path planning. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation. IEEE, 2010, pp. 2871–2876.
21. Lai, T.; Ramos, F.; Francis, G. Balancing global exploration and local-connectivity exploitation with rapidly-exploring random disjointed-trees. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 5537–5543.
22. Shi, K.; Denny, J.; Amato, N.M. Spark PRM: Using RRTs within PRMs to efficiently explore narrow passages. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014, pp. 4659–4666.
23. Li, B.; Chen, B. An adaptive rapidly-exploring random tree. *IEEE/CAA Journal of Automatica Sinica* **2021**, *9*, 283–294.
24. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). IEEE, 2000, Vol. 2, pp. 995–1001.