An Adaptive Rapidly-Exploring Random Tree

Binghui Li and Badong Chen, *Senior Member, IEEE*

***Abstract*—Sampling-based planning algorithms play an important role in high degree-of-freedom motion planning (MP) problems, in which rapidly-exploring random tree (RRT) and the faster bidirectional RRT (named RRT-Connect) algorithms have achieved good results in many planning tasks. However, sampling-based methods have the inherent defect of having difficultly in solving planning problems with narrow passages. Therefore, several algorithms have been proposed to overcome these drawbacks. As one of the improved algorithms, Rapidly-exploring random vines (RRV) can achieve better results, but it may perform worse in cluttered environments and has a certain environmental selectivity. In this paper, we present a new improved planning method based on RRT-Connect and RRV, named adaptive RRT-Connect (ARRT-Connect), which deals well with the narrow passage environments while retaining the ability of RRT algorithms to plan paths in other environments. The proposed planner is shown to be adaptable to a variety of environments and can accomplish path planning in a short time.**

***Index Terms*—Narrow passage, path planning, rapidly-exploring random tree (RRT)-Connect, sampling-based algorithm.**

I. Introduction

As a class of motion planning algorithms, sampling-based planning methods represented by probabilistic road map (PRM) [1] and rapidly-exploring random tree (RRT) [2] have achieved a lot of results in practical planning. Among them, algorithms based on RRT have been widely used in robotic arm path planning, unmanned aerial vehicle (UAV) motion planning and other robot motion planning problems [3]–[6]. These kinds of algorithms can not only solve high-dimensional planning problems which are difficult to be solved by deterministic methods, but also solve motion planning problems with differential constraints. The basic feature of sampling-based planning algorithms is to extract spatial information by sampling in an unknown space with a sampler. As the number of sampling points increases, the structure of the unknown space can be described at a higher resolution, making it possible to quickly find a path connecting two state points that meets the constraints. As an important sampling-based algorithm, RRT is based on the incremental growth of the search tree to explore the spatial state. In the planning process of RRT, after each sampling, the nearest neighbor to the sampling point in the tree will grow towards the sampling point. Since the sampling points are evenly distributed in space, the tree can gradually fill the space with the sampling process until the end of the planning.

Due to the probability characteristics of the sampler, the RRT algorithm has different results for different runs of the same task. Moreover, because it is not easy to sample in a narrow region, the number of sampling points in the narrow region is generally small. As a result, the planner cannot find the narrow passage quickly and pass through it. The randomness of RRT planning results has little effect on tasks that only need to quickly find a feasible path. However, the lack of planning capabilities for narrow passages limits the application of such algorithms in difficult environments. Because of this defect, some improved RRT algorithms have been proposed, among which rapidly-exploring random vines (RRV) [7] can achieve good results by using principal component analysis (PCA) to determine the type of local environments to increase the number of sampling points in the narrow passages. However, compared with RRT, RRV's planning time for cluttered environments may increase significantly, showing that the performance of RRV has environmental dependence.

In this paper, we propose a new improved algorithm called adaptive RRT-Connect (ARRT-Connect), which is based on the improvement of RRV and a faster bidirectional RRT named RRT-Connect [8]. Specifically, the new algorithm includes the following new features. First, a new sampler is designed for the sampling process, which is more greedy in sampling behavior. At the beginning of the planning process, the search tree will be expanded to explore the space as much as possible, thus speeding up the planning. Second, ARRT-Connect simplifies the process of judging the type of local environments, which allows the new algorithm to capture the positional relationship between sampling points and narrow passages with only a few simple calculations, and retain the ability to perform well in planning tasks with narrow passages. Finally, ARRT-Connect uses bidirectional search like RRT-Connect. To overcome the performance instability caused by greedy bidirectional search, we modify the exchange strategy of bidirectional search, so that ARRT-Connect can better complete various planning tasks and further shorten the planning time. The proposed algorithm can achieve good results in most environments, which can not only handle planning tasks with narrow regions but also retain planning abilities for general environments.

II. Related Works

To deal with the problem where the RRT algorithm is inherently inadequate in planning for difficult environments

Manuscript received March 29, 2021; revised June 3, 2021; accepted July 11, 2021. This work was supported in part by the National Science Foundation of China (61976175, 91648208) and the Key Project of Natural Science Basic Research Plan in Shaanxi Province of China (2019JZ-05). Recommended by Associate Editor Jun Zhang. *(Corresponding author: Badong Chen.)*

Citation: B. H. Li and B. D. Chen, "An adaptive rapidly-exploring random tree," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 2, pp. 283–294, Feb. 2022.

The authors are with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: l372084406@stu.xjtu.edu.cn; chenbd@mail.xjtu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2021.1004252

with narrow regions, many improved algorithms have been proposed. These algorithms can be divided into algorithms based on improved sampling distribution and algorithms based on improved tree extension.

The algorithms based on improved sampling distribution aim to increase the number of sampling points in difficult regions such as narrow passages by changing the distribution of sampling points and increasing the probability of passing through narrow regions. In [9], the authors use bridge-test to increase the sampling points in the narrow passages while reducing the points in the non-interested areas, so that the planner's exploration opportunities for narrow passages increase. In [10], the authors add a radius for each node in the tree to indicate that the region around the node is a free area that can be sampled. If an obstacle is encountered in the expansion process of a node in the tree, the radius of the node will be updated. This method increases the number of valid sampling points and improves the planning ability for the bug environment. In [11], the authors add a parameter to describe the density of surrounding points for each node of the tree. The sampler then collects new points around the lowest density point to expand the tree into unexplored areas. In [12], if there is an obstacle on the line between the sampling point and the nearest neighbor point in the tree, local optimization will be performed. First, the sampling point retreats to the boundary of the obstacle, and then the boundary point closest to the original sampling point is found. All the involved points are finally added to the tree in a normal manner to increase the probability of the tree entering the narrow passages. In [13], the authors change the sampling behavior by dividing the space into different regions. This method calculates the entropy of the sampling points and divides the space into the obstacle, free, boundary and difficulty areas according to the value of entropy. After that, the number of sampling points in the difficult area will be increased so that the planner can better deal with planning problems in a difficult area. In [14], the authors record a radius value R for each node in the tree, which means that the hypersphere with this node as the center and R as the radius is a free space. Moreover, only the sampling points outside the hypersphere are accepted by the planner, so the tree can be expanded to unexplored areas as much as possible.

The improved algorithms above achieve the desired performance in the face of an environment containing narrow passages. However, many algorithms introduce additional variables that need to be updated for each node in the tree, which will increase the amount of calculation after sampling. Furthermore, during the planning period, most of the planners are not aware of the spatial structure. Therefore, the drawback of the above algorithms is that it is unable to compute a more suitable expansion direction for the tree in difficult environments.

The algorithms based on the improved tree extension focus on the aspect of how to calculate an appropriate extension direction for the search tree to increase expansion ability in narrow passages. In [15], if a sampling point cannot connect to the search tree in the extension step, a local tree will be generated at the sampling point. The planner will then try to connect these trees. If a tree is in a narrow passage, the planning time of such difficult environments can be reduced. In [16], the planner uses the information of obstacles in the workspace to calculate the appropriate expansion direction. But a lot of calculations is needed to obtain the geometric information of obstacles. In [17], in view of a situation where the nodes in the tree are already in a narrow passage, the authors use PCA to calculate the extension direction for the tree, so that the tree can more easily pass through the narrow passage. However, it does not solve the problem of how the tree enters the narrow passage. In [18], the authors use bridge-test and PCA to find narrow passages, then adjust the direction of tree expansion. At the same time, to speed up the planning process, only the sampling points outside the free hypersphere area of each node are accepted by the planner. In [19], the authors propose a method to find narrow passages by clustering the sampling points. After clustering, a tree will be generated in each center and the reinforcement learning method will be used to expand these trees. In [20] and [21], the authors also use multiple local trees to adjust the global search state. Moreover, the authors in [21] try to use Bayesian estimation to adjust the local sampling direction to enhance the algorithm's processing ability in difficult environments. These improved algorithms generally compute and select the appropriate tree extension direction according to the spatial information. However, most of these methods have high computational complexity and strong dependence on narrow environments.

The context-dependent algorithm [22] may solve the environmental dependency that appears in previously mentioned algorithms. Nevertheless, that algorithm is implemented by combining several methods. The algorithm proposed in this paper attempts to solve the problems of narrow passages and environmental adaptability in a simple process.

III. Proposed Algorithm

The algorithm proposed in this paper aims to improve the planning ability of traditional RRT algorithms for an environment containing narrow passages. Meanwhile, in order to reduce the planner's dependence on such specific environments, the new algorithm retains the planning capabilities for common environments (e.g., cluttered) as much as possible. The new algorithm modifies three important parts of RRT-Connect to achieve the desired performance. Details of the proposed algorithm are given below.

A. Algorithm Framework

The sampling space used in the proposed algorithm is the configuration space (C-space). The configuration space is a space composed of parameters describing the state of the robot. The states that the robot can reach form a free space, while the set of states that the robot can not reach or that the robot will collide with obstacles is obstacle space. The concept of configuration space has been widely used in various planning algorithms since it was proposed. Common configuration spaces include the space composed of joint variables of the manipulator, the space composed of the coordinates of the planar robot, etc. In the configuration space,

the robot is simplified to a point, so no matter what kind of robot system, we are ultimately planning a path for a point in the space. If an algorithm can solve a certain configuration space planning problem, then the robot system similar to this configuration space can use this algorithm for motion planning. From another perspective, the configuration space is equivalent to a higher level of abstraction of the planning problem, which makes the planning algorithm separate from the specific robot system and allows for application in more motion planning problems.

The framework of the new algorithm is identical to RRT-Connect in [8] and is shown in Algorithms 1 and 2. The modified parts are Random_Config, Extend and Swap respectively. First, we design a new sampler to get random configurations in the Random_Config function. The parameter R_T is used to describe the size of the tree in C-space. The P_{goal} and $P_{outside}$ control the probability of the sampler sampling in different regions. Second, when the extension fails, we will determine the type of local environment to reboot the tree's extension in the Extend function. Last, the density information of the tree is used to achieve a more flexible exchange in the Swap part. The complete description of these three modifications is given in subsequent sections.

Algorithm 1 ARRT_Connect(q_{init}, q_{goal})

Require: starting point q_{init} and target point q_{goal}
1: T_a.init(q_{init})
2: T_b.init(q_{goal})
3: **for** k = 1 to K **do**
4: q_{rand} = Random_Config(R_T_a, q_{goal}, P_{goal}, $P_{outside}$)
5: **if** (Extend(T_a, q_{rand}) ≠ $Trapped$) **then**
6: **if** (Connect(T_b, q_{new}) = $Reached$) **then**
7: **return** Path(T_a, T_b)
8: **end if**
9: **end if**
10: Swap(T_a, T_b)
11: **end for**
12: **return** $Fail$

Algorithm 2 Connect(T, q)

Require: tree T and point q
1: **repeat**
2: S = Extend(T, q)
3: **until** $S \neq Advanced$
4: **return** S

B. Greedy Sampler

As the first step of the algorithm, the behavior of the sampler directly affects the exploration process and the final planning performance. The goal of the sampler proposed in this paper is to explore the space more quickly to shorten the overall planning time. In [23] and [24], the planners sample multiple points in each sampling process, then they select appropriate tree nodes and corresponding sampling points to expand the tree according to the established criteria. Essentially, these two methods reduce invalid sampling points and planning time by selecting as many unexplored areas as possible in the sampling process. The basic idea of the sampler used in ARRT-Connect is the same, but the acceleration is achieved by constantly changing the sampling area rather than adding additional sampling points. That means the sampler is limited to sample points in the selected unexplored region with a higher probability and the selected area may change in each sampling procedure. The details are shown in Algorithm 3.

Algorithm 3 Random_Config(R_T, q_{goal}, P_{goal}, $P_{outside}$)

Require: region of tree R_T, target point q_{goal}, and two threshold P_{goal}, $P_{outside}$
1: **if** all dimensions of the R_T reach the corresponding boundary of C-space **then**
2: q_{rand} = random configuration in the space
3: **else**
4: p_{rand} = Random_Number()
5: **if** $p_{rand} \leq P_{goal}$ **then**
6: $q_{rand} = q_{goal}$
7: **else**
8: **if** $p_{rand} \geq P_{outside}$ **then**
9: q_{rand} = random configuration in R_T
10: **else**
11: **for** i = 1 to D **do**
12: ur_i = the length of unexplored intervals in dimension i
13: **end for**
14: $d_{\max}$ = the dimension with the largest ur value
15: $P_{\max} = ur_{d_{\max}} / \sum_{i=1}^{D} ur_i$
16: **if** Random_Number() ≤ $P_{\max}$ **then**
17: $dim = d_{\max}$
18: **else**
19: dim = one of the remaining dimensions
20: **end if**
21: q_{rand} = Config_In_Range(one of the unexplored intervals of dim and full range of other dimensions)
22: **end if**
23: **end if**
24: update $P_{outside}$
25: **end if**
26: **return** q_{rand}

In Algorithm 3, two parameters, P_{goal} and $P_{outside}$, divide [0, 1] into three intervals, which in turn represent the sampling probability of the sampler in the target region, the unexplored region and the area within the tree. Sampling in these three different areas has different meanings. Goal-biased sampling can speed up the connection of the tree to the target point. Sampling in the unexplored area can accelerate the expansion of the tree, while the uniform sampling in the tree area can increase the spatial resolution of the explored region. The value of P_{goal} is generally set to be relatively small and does not need to be changed in the whole process, while the value of $P_{outside}$ is set to be larger at the beginning (e.g., 0.95). This is done to sample in the unexplored area as much as possible at the beginning. Afterwards, as the number of points increases during the process of the algorithm, the value of $P_{outside}$ gradually decreases, so that the sampler tends to

sample in the explored region.

If the target area is selected for sampling, the sampler returns the goal point as the sampling point directly. But, in the other two cases, the sampler needs to use the parameter R_T to select the unexplored region or the region within the tree. R_T is a parameter added to the search tree, which describes the minimum external hypercube of the tree, named the tree region. Therefore, the sampler sample uniformly in the region described by R_T can get a random point in the explored area (Lines $8-9$ in Algorithm 3). R_T records the range of intervals the tree occupies in each dimension of the space. The original range of each dimension minus the interval of R_T in that dimension constitutes two independent unexplored intervals (Fig. 1). These unexplored intervals are the basis for the sampler to select the sampling region. When the unexplored area is selected to be sampled (Lines $11-21$ in Algorithm 3), the length of each unexplored interval dimension is first calculated, then the dimension with the longest unexplored interval will be recorded. After that, the sampler will select the recorded dimension with a probability $P_{\max}$, where $P_{\max}$ is the ratio of the unexplored interval lengths of the recorded dimension to the sum of the unexplored interval lengths of all dimensions. Otherwise, it will choose one of the remaining dimensions that contain unexplored intervals. Finally, an unexplored region consisting of the full range of the remaining dimensions and the larger of the two unexplored intervals of the selected dimension is selected. The sampler will sample uniformly in this unexplored area to return a random point. In the whole sampling process, if R_T overlaps with the C-space, i.e., the search tree touches all the boundaries of the configuration space, then the sampler will degenerate to global uniform sampling to increase the number of sampling points in the tree area.

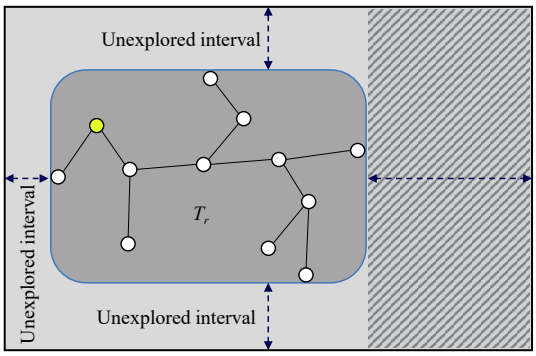


Fig. 1. R_T divides the C-space into two regions: inside the tree and outside the tree, the sampler will sample in the unexplored regions with a higher probability. The shaded region on the right of the figure is one of the unexplored regions that might be selected.

Such a design enables the planner to rapidly expand the tree to all parts of the C-space in the early stage of the planning process, and then gradually increase the spatial resolution within the tree. Moreover, because of the rapid exploration of configuration space, this strategy also helps the planner to accomplish the planning tasks with fewer sampling points.

C. Environmental Judgment

After obtaining a sampling point from the sampler, the normal RRT-Connect extension method is first tried. If it fails, it indicates that there is an obstacle between the sampling point and the nearest point in the tree, then the environmental type judgment process is entered and an appropriate direction will be selected to retry the expansion operation. The whole process is summarized as Algorithm 4.

Algorithm 4 Extend(T, q_{rand})

Require: tree T and sample point q_{rand}
1: q_{near} = Nearest_Neighbor(T, q_{rand})
2: **if** New_Config($q_{near}, q_{rand}, q_{new}$) **then**
3: T.add_vertex(q_{new})
4: T.add_edge(q_{near}, q_{new})
5: **if** $q_{new} = q_{rand}$ **then**
6: **return** *Reached*
7: **else**
8: **return** *Advanced*
9: **end if**
10: **else**
11: S_{points} = Local_Sampling(q_{near})
12: $S_{obstacle}, S_{free}$ = Divided_BasedOn_Free(S_{points})
13: ave_{obs} = Mean($S_{obstacle}$)
14: **if** InObstacle(ave_{obs}) **then**
15: q_{obs1}, q_{obs2} = Furthest_Point_Pair($S_{obstacle}$)
16: Expand q_{near} along the line between q_{obs1}, q_{obs2}
17: **else**
18: **if** Distance(ave_{obs}, q_{near}) $\geq \delta$ **then**
19: Expand q_{near} towards ave_{obs}
20: **else**
21: q_{free1}, q_{free2} = Furthest_Point_Pair(S_{free})
22: Expand q_{near} along the line between q_{free1}, q_{free2}
23: **end if**
24: **end if**
25: **return** $Trapped$
26: **end if**

At the beginning of the environmental type judgment process, the planner needs to collect the local spatial information around the q_{near} that failed to expand, which is accomplished by local sampling. In this paper, local sampling refers to selecting points at a certain distance from the center point along each dimensional axis. To obtain enough sampling points to describe the local structure, the planner selects the first batch of points with q_{near} as the center and then takes the first batch of sampling points as the center to obtain the second batch of points. Moreover, the expansion distance used in the second batch is smaller than that in the first batch to gain more proper points in local space. Fig. 2 describes the entire procedure of local sampling. The advantage of this local sampling method is that the number of sampling points obtained by local sampling will automatically increase with an increase in spatial dimension, so it is unnecessary to artificially adjust the number of local sampling points every time the dimension changes.

After we get points S_{points} in local sampling, we divide the points into $S_{obstacle}$ and S_{free} according to whether the

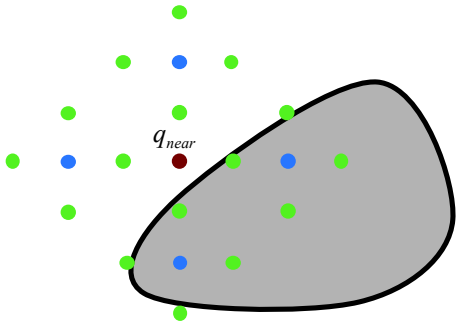


Fig. 2. Process of local sampling. The red point q_{near} represents the point in the tree that failed to expand, while the blue points and the green points respectively represent the first batch of local sampling points centered on q_{near} and the second batch of sampling points centered on the first batch of points.

sampling point is located in free regions, then the mean point ave_{obs} of the non-free set $S_{obstacle}$ is calculated. If ave_{obs} is in an obstacle region, it is considered to encounter a wall-type obstacle, after which the distance between any two points in $S_{obstacle}$ is calculated to get the two points farthest apart. The line of these two points is the new expansion direction. This direction generally represents the boundary direction of the wall, so the tree shows the characteristics of growing along the surface of the obstacles. However, if ave_{obs} is in free space, it is further judged whether the distance between q_{near} and ave_{obs} is greater than δ, which is a threshold smaller than the extension step size. If it is, the q_{near} is considered to be outside a narrow passage, the search tree will try to expand towards ave_{obs}. Otherwise, q_{near} is likely to be located in a narrow passage after which the two points farthest apart in S_{free} will be calculated. The new expansion direction will be the connection direction of these two points. The tree will try to expand several times, which makes it easier for the tree to pass through the narrow passage. If one of them fails, the algorithm will immediately enter the final exchange process. Fig. 3 shows the three scenarios. When using the above strategy to determine the type of local environment, concave obstacles may be judged as narrow passages, resulting in additional time consumption. However, as the extension fails, the algorithm will enter the exchange procedure immediately, so the algorithm will not spend too much time in the case of an incorrect judgment.

D. Bidirectional Exchange Strategy

The last step in each round of the algorithm is to select one of the search trees to expand in the next iteration according to the policy. In general bidirectional search, whether two trees need to exchange depends on the comparison result of the number of nodes between them. This is to balance the size of the two trees. However, in difficult environments, this strategy can take a lot of time to expand a tree with few nodes and hindered growth. Besides, the new sampler and the bidirectional search are greedy, which leads to the problem where the algorithm may sample too many points in the local space. Therefore, a more flexible exchange method is needed to balance the search process of the two trees.

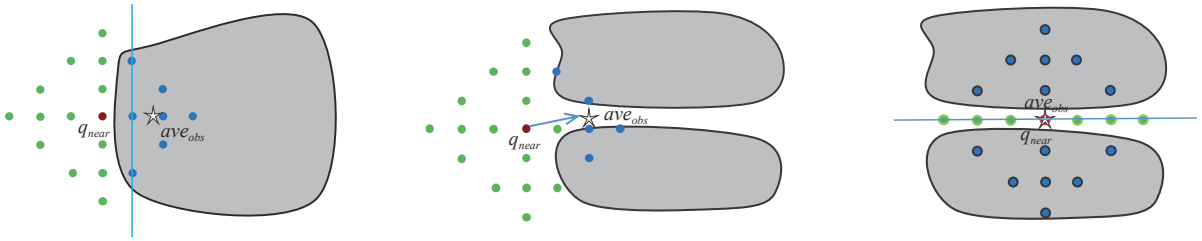
Algorithm 5 Swap(T_a, T_b)

Require: tree T_a and T_b
1: **if** Nodes(T_a) < Nodes(T_b) **then**
2: swap T_a and T_b
3: **else**
4: $failure = failgure + 1$
5: **if** $failure \geq threshold$ **then**
6: $density_a$ = Density(T_a)
7: $density_b$ = Density(T_b)
8: **if** $density_a > density_b$ **then**
9: expand the T_b
10: **else**
11: expand the T_a
12: **end if**
13: swap T_a and T_b
14: $failure = 0$
15: **end if**
16: **end if**

The exchange strategy of the new algorithm is consistent with the strategy of swapping according to the number of nodes under normal conditions, but when it fails, the new strategy will record the number of failures. If the recorded value is greater than the *threshold*, the planner will enter the forced exchange process (Lines 6 – 13 in Algorithm 5). In the forced exchange process, R_T and the number of nodes of each tree are used to calculate the current point density of the tree, and then the tree with lower point density is selected to carry out the sampling and expansion procedures described in Sections III-B and III-C to increase the number of nodes in the tree area. Finally, regardless of whether the expansion of the previous step is successful, two trees will be forced to swap. The new strategy can better balance the expansion of the two trees to increase the stability of the algorithm.

IV. Experiments

In this section, we will compare the planning performance of RRT, RRT-Connect, RRV and the proposed ARRT-Connect. The experiments include two types: A basic environmental test and integrated environmental test. Each type contains a variety of tests to compare the adaptability of each planner in different environments. All experiments are conducted on a machine with an Intel I7-7700 3.6 GHz processor and 4 GB of memory. All algorithms have been written in Python and use an efficient multi-dimensional space search structure, *R*-tree [25], to quickly find the nearest neighbor. The expansion step of the tree is selected in {1, 3} in the basic environmental test, and in {20, 22} in the integrated environmental test. For local sampling after expansion failure in all tests, the sampling radius of RRV is 1.5 times the tree step size, while the two local expansion steps of ARRT-Connect are 1.5 times and 0.75 times the tree step size respectively. To speed up the planning, all algorithms are set to sample in the goal area with a probability $P_{goal} = 0.01$ [26] and the parameter P_{ouside} in ARRT-Connect



(a) (b) (c)

Fig. 3. Three types of local environments. (a) Wall; (b) Entrance of the narrow passage; (c) Narrow passage. The red points are q_{near} and the star points are ave_{obs}. The ave_{obs} point in the obstacle area indicates that a wall-type obstacle is encountered. Otherwise, if ave_{obs} and q_{near} are far apart, q_{near} is likely to be at the entrance of the narrow passage, and if they are close together, q_{near} may be in the narrow passage. Arrows or lines are the new extension directions for q_{near}.

is set to 0.95 initially. The maximum number of sampling points is set to 25 000 for the basic environmental test and 20 000 for the integrated environmental test. If the path is not found within this limit, the planning is considered as a failure. Each experiment is tested 50 times to collect relevant data. The execution time, the total number of nodes in the tree and the success rate will be given for each result to evaluate the overall performance of the algorithms. To clearly show the time-consuming differences of different algorithms, all the time units are milliseconds. The execution time and the number of nodes are calculated using only data from successful planning in 50 tests, and the success rate represents the ratio of the number of successful path planning in 50 tests.

A. Basic Environmental Test

In the basic environmental test, we will test different environments in two-dimensional and three-dimensional space. For two-dimensional tests, we test algorithms in four different environments and Fig. 4 shows the planning results of four algorithms in these environments, where triangle and square points represent the starting and the target points respectively. The Simple environment is used to test the performance of the algorithms under easy obstacles. The purpose of the Cluttered environment is to test whether the performance of the new algorithm will not be significantly reduced due to additional calculations when dense obstacles are encountered. The planning paths in the Narrow and Bug Trap environments must pass through narrow passages. Therefore, in these two difficult environments, whether the algorithm can quickly find and pass through a narrow passage will be very important. From Figs. 4 (a) and 4(b), we can find that the results of ARRT-Connect are similar to RRT-Connect and the tree nodes of these two algorithms are less than those of RRT and RRV. In Figs. 4 (c) and 4(d), ARRT-Connect uses the fewest nodes to find the path, showing the best performance among the four algorithms. These results show that the new algorithm has strong planning ability and good adaptability in a normal environment and an environment with narrow passages.

The specific statistical results of the four algorithms in two-dimensional environments are shown in Table I. From the column of average time in Table I, we can find that ARRT-Connect gives the best results for Narrow and Bug Trap environments. In the test of the Narrow environment, the average planning time of RRT and RRT-Connect is 16 and 9 times that of ARRT-Connect respectively, while the planning time of RRV algorithm, which is good at handling with narrow passages, is 4 times that of the new algorithm. For the Bug Trap environment, the planning time of RRT and RRT-Connect is about 57 times that of ARRT-Connect, and the time consumption of RRV is about 28 times that of ARRT-Connect. Moreover, neither RRT nor RRT-Connect is able to plan a path in all tests in these two environments, while both RRV and ARRT-Connect have a 100% success rate. For the other two environments that do not contain narrow passages, RRT-Connect takes the least time, and the new algorithm follows closely behind. Although the time consumption of ARRT-Connect in these two environments is not the lowest, its performance is 2–5 times faster than RRT and 6–10 times faster than RRV, so it can efficiently complete planning tasks in the multi-obstacle environment.

In terms of the variance of time consumption, ARRT-Connect performs well, especially in difficult environments, which reflects the stability of the new algorithm in various environments. It can also be seen from Table I that the new algorithm has the lowest average number of sampling points in all tests, that is, the new algorithm can use fewer points to explore the space and provide feasible planning results, which reflects the efficiency of ARRT-Connect from another aspect.

To test the influence of the sampler and the exchange strategy on the planning results, we changed the sampler and exchange strategy respectively and carried out the test in the above two-dimensional environments. Table II shows the effect of the new sampler on the performance of the ARRT-Connect algorithm. In Table II, the Free sampler means that only the sampling points in the free area are accepted, while the Greedy sampler is the new sampler described in Section III-B. The results show that ARRT-Connect with Greedy sampler performs best in all tests, where not only the average planning time of the algorithm is lower, but also the standard deviation is smaller and the performance is more stable. In the test of the Simple environment, the results of the two samplers are similar. This is because the Simple environment is easy and only a few sampling points are needed to complete the planning, so they are all less time-consuming.

Table III compares the performance of ARRT-Connect with three different exchange strategies, namely swapping at each

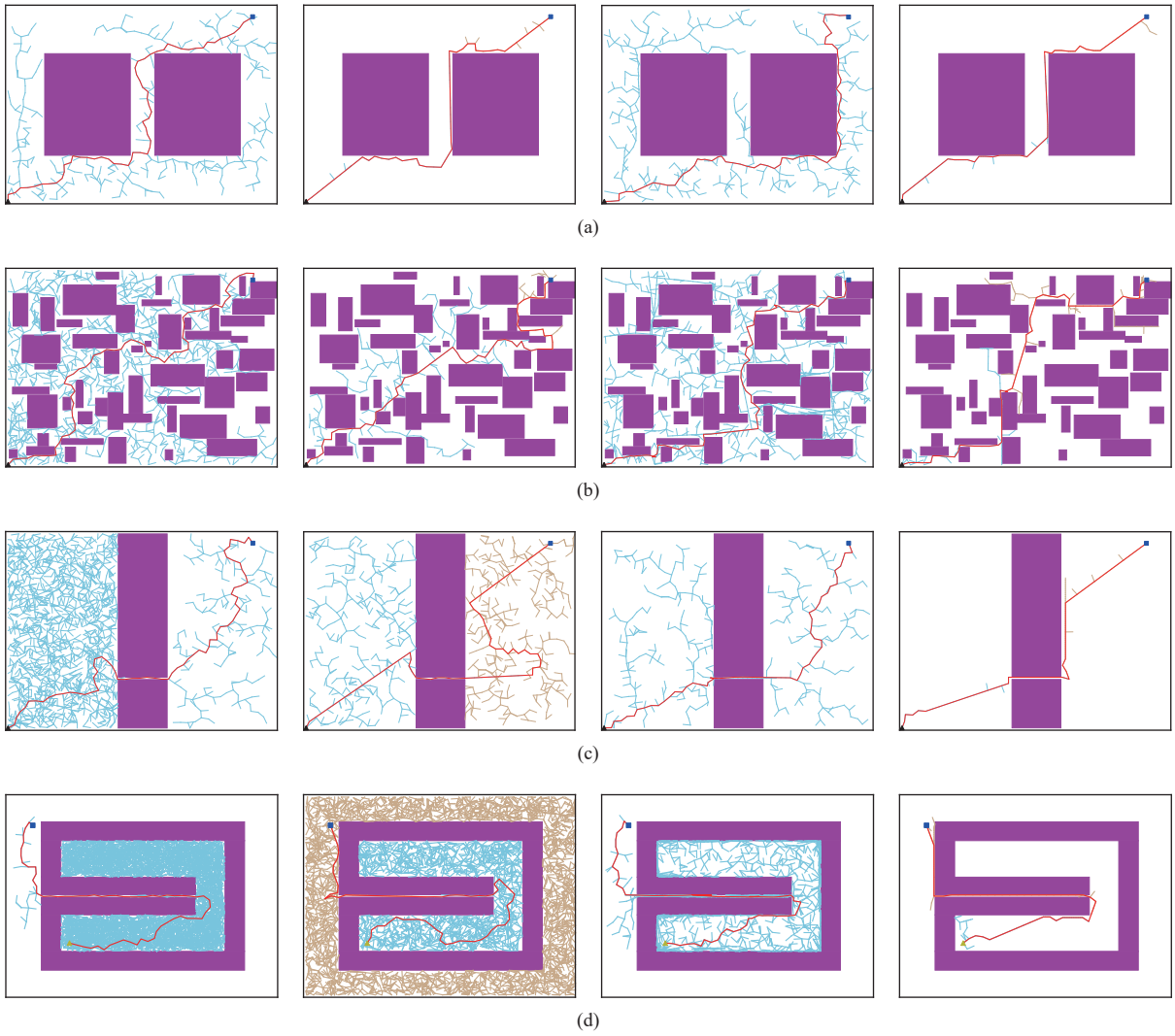


Fig. 4. Planning results of four algorithms for two-dimensional environments, in which the triangle points are the starting points and the square points are the target points. From left to right in each row are the planning results of RRT, RRT-Connect, RRV and ARRT-Connect algorithms. (a) Simple; (b) Cluttered; (c) Narrow; (d) Bug Trap.

iteration, swapping according to the number of nodes of the two trees, and the proposed adaptive swapping. Similar to Table II, the new swap strategy performs best on most tests, and the variance of the algorithm's time consumption with the newly proposed exchange strategy is always the lowest. It indicates that the proposed exchange strategy can better solve the problem of search direction replacement, thus reducing the time consumption of the algorithm when one direction can not be expanded.

For three-dimensional tests, the algorithms are tested in three different environments (Fig. 5), among which the Simple3d and Narrow3d environments are the three-dimensional extensions of the corresponding environments in two-dimensional space respectively, while the Multi-Narrow3d is a more difficult environment similar to the Narrow3d. Moreover, in Narrow3d and Multi-Narrow3d environments, the feasible path must pass through narrow passages. The statistical results are given in Table IV. Results on the Simple3d environment show that bidirectional search is far more efficient than one-way search, with average time consumption only about one-tenth of that of one-way search. Meanwhile, the performance of ARRT-Connect and RRT-Connect is basically the same. In the other two tests, ARRT-Connect is the algorithm with the best performance. The average planning time of RRT-Connect is 2–5 times that of ARRT-Connect, and the performance of the other two algorithms are worse. Furthermore, ARRT-Connect has the least number of sampling points in all tests, and the variance is small, showing the same stability as the two-dimensional space tests. Fig. 6 shows clearly that the ARRT-Connect can complete the planning task with fewer points in the Narrow3d environment, which intuitively reflects the efficiency of the proposed algorithm.

B. Integrated Environmental Test

In this experiment, we will test the algorithms on a simulated office map. It is more convenient to place different obstacles by using the simulated office map. The size of the map is 2413 × 1268 and there are four rooms A, B, C and D on the office map (Fig. 7). Room A is a grocery store with a

TABLE I
RESULTS FOR PLANNNING IN DIFFERENT TWO-DIMENSIONAL ENVIRONMENTS

Environment	Algorithm	Average time	Min time	Max time	Standard deviation	Mean nodes	Success rate
Simple	RRT	296.08	155.56	431.23	70.38	499	1.00
	RRT-Connect	**46.62**	37.71	**63.09**	**7.36**	134	1.00
	RRV	573.04	234.11	1 148.44	206.18	504	1.00
	ARRT-Connect	50.42	**35.76**	94.33	12.52	**76**	1.00
Cluttered	RRT	1232.08	571.67	2726.06	490.12	2004	1.00
	RRT-Connect	**294.77**	**72.86**	926.36	204.15	796	1.00
	RRV	2798.66	1024.45	7770.14	1239.57	1453	1.00
	ARRT-Connect	439.26	178.72	**867.84**	**176.73**	**465**	1.00
Narrow	RRT	1843.09	345.43	9154.26	1794.75	3720	0.97
	RRT-Connect	1090.83	91.07	8627.04	1760.80	2512	0.90
	RRV	416.96	136.77	891.72	185.09	519	1.00
	ARRT-Connect	**112.94**	**37.29**	**388.43**	**70.98**	**138**	1.00
Bug Trap	RRT	8386.37	4181.81	12 057.08	1825.24	20 710	0.97
	RRT-Connect	8764.26	88.37	19 484.30	5394.85	17 566	0.93
	RRV	4108.97	718.02	9319.44	2033.42	2429	1.00
	ARRT-Connect	**146.32**	**57.67**	**839.52**	**151.87**	**164**	1.00

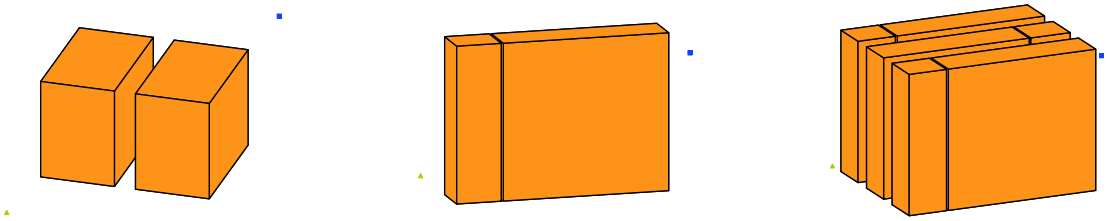
TABLE II
RESULTS FOR NEW ALGORITHM WITH DIFFERENT SAMPLERS

Environment	Sampler	Average time	Min time	Max time	Standard deviation	Mean nodes	Success rate
Simple	Free	67.18	38.67	257.96	46.60	**89**	1.00
	Greedy	**61.05**	**27.47**	**152.23**	**30.85**	90	1.00
Cluttered	Free	453.87	123.84	2578.88	487.76	479	1.00
	Greedy	**442.11**	**111.59**	**965.02**	**221.99**	**472**	1.00
Narrow	Free	117.21	65.36	737.23	119.71	149	1.00
	Greedy	**104.81**	**43.14**	**249.69**	**53.27**	**132**	1.00
Bug Trap	Free	234.33	**54.37**	1416.70	360.26	270	1.00
	Greedy	**105.77**	63.43	**184.51**	**33.02**	**125**	1.00

TABLE III
RESULTS FOR NEW ALGORITHM WITH DIFFERENT EXCHANGE STRATEGIES

Environment	Exchange strategy	Average time	Min time	Max time	Standard deviation	Mean nodes	Success rate
Simple	Every turn	41.16	**27.82**	143.04	21.14	**68**	1.00
	Depend on nodes	86.08	30.28	188.46	40.57	106	1.00
	Adaptive	**39.84**	28.40	**125.29**	**13.93**	78	1.00
Cluttered	Every turn	409.39	153.45	852.47	178.43	508	1.00
	Depend on nodes	582.75	186.64	1742.29	297.58	651	1.00
	Adaptive	**352.89**	**150.49**	**589.65**	**122.89**	**458**	1.00
Narrow	Every turn	120.30	30.87	1728.53	240.01	171	1.00
	Depend on nodes	155.20	**26.26**	1623.06	290.85	210	1.00
	Adaptive	**105.19**	35.09	**267.75**	**64.46**	**149**	1.00
Bug Trap	Every turn	**109.62**	**41.17**	970.93	166.05	**134**	1.00
	Depend on nodes	242.78	55.84	2887.44	439.80	300	1.00
	Adaptive	139.19	64.04	**517.75**	**94.43**	139	1.00

lot of obstacles. Rooms B and C are similar to meeting rooms or waiting rooms, so there are only a few simple obstacles, while room D is equivalent to a room with large equipment and contains simple narrow passages. Four scenarios will be



(a) (b) (c)

Fig. 5. Experimental environments in three-dimensional space, in which the triangle points are the starting points and the square points are the target points. (a) Simple3d; (b) Narrow3d; (c) Multi-Narrow3d.

TABLE IV
RESULTS FOR PLANNING IN DIFFERENT THREE-DIMENSIONAL ENVIRONMENTS

Environment	Algorithm	Average time	Min time	Max time	Standard deviation	Mean nodes	Success rate
Simple3d	RRT	906.45	529.89	1326.18	183.49	1464	1.00
	RRT-Connect	**77.14**	56.39	**133.20**	**15.19**	235	1.00
	RRV	1832.37	1051.63	3384.89	429.37	1496	1.00
	ARRT-Connect	122.49	**55.63**	223.66	46.42	**136**	1.00
Narrow3d	RRT	2874.61	1126.40	8080.04	1880.34	4861	1.00
	RRT-Connect	1606.15	302.74	7478.29	1815.60	3889	1.00
	RRV	1907.68	1268.61	2548.49	377.26	1452	1.00
	ARRT-Connect	**267.10**	**78.75**	**782.61**	**142.37**	**260**	1.00
Multi-Narrow3d	RRT	5693.90	3022.54	8624.82	1400.78	8277	1.00
	RRT-Connect	2262.42	865.59	5598.76	1179.91	5003	1.00
	RRV	5497.75	3617.57	8975.81	1059.47	4421	1.00
	ARRT-Connect	**1027.41**	**344.43**	**3081.75**	**631.92**	**713**	1.00

tested in the experiment. The first three scenarios are the planning from room A, B, C to room D, that is, the path planning from the ordinary obstacle environments to a narrow passage environment. The fourth scenario is the planning from room B to room C to test the performance of the algorithm in a normal environment. The statistics are given in Table V. The path results of Scenario I to IV are shown in Fig. 8. Table V shows that the proposed algorithm performs best in all tests. Furthermore, in the first three scenarios, RRT takes the longest time and has the lowest success rate, which shows that RRT can not deal with the planning problem of an environment with narrow passages well. Because the narrow passages in the map are relatively simple, RRT-Connect can complete all the planning, and the time consumption is lower than the RRV algorithm. This indicates the high efficiency of dual-tree search, and also reflects the problem where even though RRV can cope with narrow passages, its efficiency will be reduced when encountering common obstacles. ARRT-Connect has obvious advantages in these three scenarios. Besides low time consumption, the variance of time consumption is also small, which shows the stability of the proposed algorithm. For the fourth scenario, the performance of ARRT-Connect and RRT-Connect is basically the same, and both of them are better than the other two algorithms. This once again proves the adaptability of the new algorithm, that is, the planning performance of the new algorithm under a normal environment is not significantly affected by the additional computation added to cope with narrow passages.

V. DISCUSSIONS

There are two important parts in ARRT-Connect, one is the sampling probability of the new sampler in the area outside the tree at the beginning, and the other is the sampling range during local sampling. The sampling probability outside the tree influences the overall behavior of the sampler, thereby influencing the way the tree explores the space. In ARRT-Connect, this probability is determined by $P_{outside}$. If the initial value of $P_{outside}$ is small, the search tree will tend to sample within the tree, and the expansion speed of the tree will slow down. Gradually, the point density within the tree will increase, which can better describe the spatial structure within the tree. If $P_{outside}$ is large, the search tree will tend to expand outwards, so that the tree can explore more space with fewer points, thereby increasing search efficiency. ARRT-Connect initially sets the value of $P_{outside}$ to be large, and gradually decreases the value of $P_{outside}$ as the number of sampling points increases, so that the algorithm can better explore the space. The range of local sampling affects the quality of the local information obtained. If the sampling range is too large, the sampling points can not describe the local spatial information near the nearest neighbor point. On the contrary, the sampling points will not give the structure of local space accurately, so a moderate local sampling range can better guide the expansion of the tree.

In addition to the aforementioned key points, there is a small defect in the environmental judgment process, that is, the concave obstacles may be judged as the narrow passages

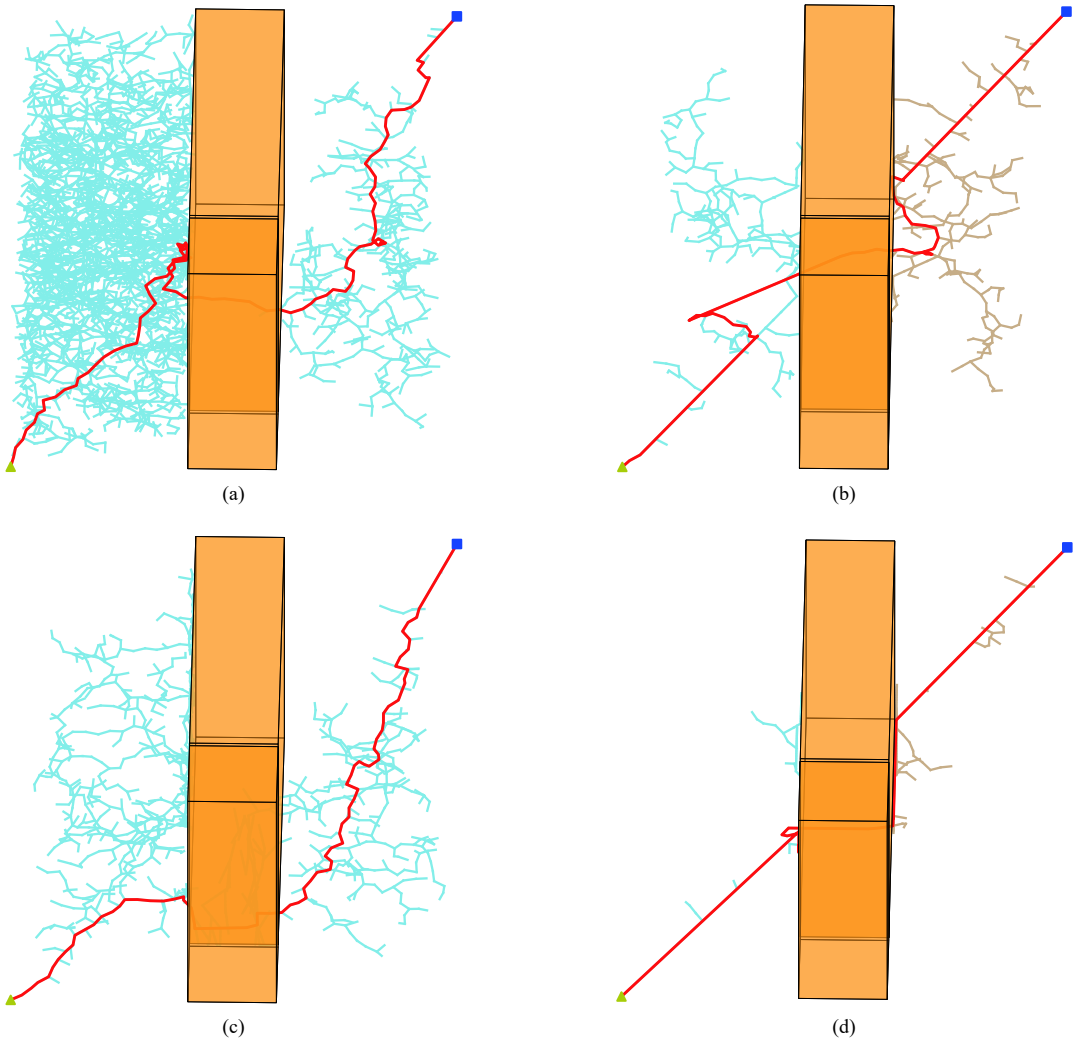


Fig. 6. Planning results of four algorithms for Narrow3d environment. (a) RRT; (b) RRT-Connect; (c) RRV; (d) ARRT-Connect.

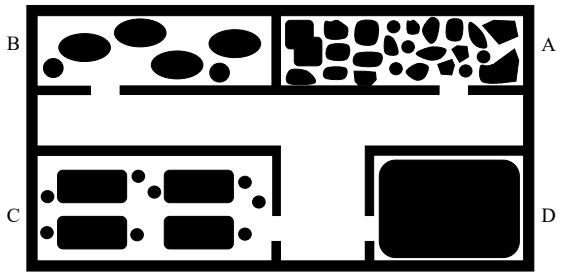


Fig. 7. Office map used in the integrated environmental test. There are four rooms A, B, C and D in anticlockwise order from the upper right corner.

or the entrance of the narrow passages, which will increase the time consumption of the algorithm. However, because the proposed algorithm uses bidirectional search and improves the exchange strategy, the planner will try to expand in the other direction when the tree falls into a situation where it cannot be expanded in one direction, thereby reducing the impact of incorrect judgments.

VI. Conclusions

In this paper, a new algorithm based on RRT-Connect and RRV is proposed to deal with the motion planning problems with narrow passages. The proposed algorithm (ARRT-Connect) combines a new greedy sampler, an environmental judgment process, and a flexible two-way exchange strategy to achieve the best results compared with RRT, RRT-Connect and RRV in an environment with narrow passages. Among them, the environmental judgment process can use information obtained from local sampling to calculate a new feasible direction (e.g., along the surface of an obstacle or along a narrow passage) for the nearest neighbor point that fails to expand. Due to the extra calculation time required in the judgment process, the performance of ARRT-Connect in a cluttered environment has weakened. Bidirectional search and the proposed sampler reduce this influence and accelerate the planning speed. Therefore, the performance of the new algorithm in a cluttered environment is closer to RRT-Connect, and the new algorithm can adapt to various environments. We also discuss the key factors that have an impact on the efficiency of the algorithm, and their influence on the behavior of ARRT-Connect is analyzed. This gives the direction for further research on this kind of algorithm.

TABLE V
RESULTS FOR PLANNING IN INTEGRATED ENVIRONMENT

Environment	Algorithm	Average time	Min time	Max time	Standard deviation	Mean nodes	Success rate
Scenario I	RRT	13 598.91	10 447.97	16 374.88	1788.21	17 508	0.20
	RRT-Connect	1630.00	624.71	4158.41	827.61	4623	1.00
	RRV	5616.84	1437.86	10 459.30	1760.08	2096	1.00
	ARRT-Connect	**912.58**	**239.03**	**1655.38**	**523.01**	**1046**	1.00
Scenario II	RRT	14 290.64	11 728.30	16 539.44	1322.01	18 147	0.20
	RRT-Connect	1745.12	982.04	3049.26	472.41	4676	1.00
	RRV	4473.92	1310.90	8630.67	1252.99	2252	1.00
	ARRT-Connect	**523.22**	**299.48**	**718.76**	**105.97**	**563**	1.00
Scenario III	RRT	13 623.22	10 853.42	15 920.13	1532.23	17 295	0.54
	RRT-Connect	1331.29	367.55	3060.15	617.98	3676	1.00
	RRV	2261.75	1101.41	5513.61	866.79	1325	1.00
	ARRT-Connect	**316.49**	**119.14**	**556.88**	**127.04**	**367**	1.00
Scenario IV	RRT	4524.04	2764.41	7781.14	1168.70	7016	1.00
	RRT-Connect	1219.01	561.50	2635.08	666.94	2788	1.00
	RRV	7441.39	3783.31	11 326.01	1510.79	3739	1.00
	ARRT-Connect	**1006.42**	**412.27**	**1996.61**	**439.88**	**1161**	1.00

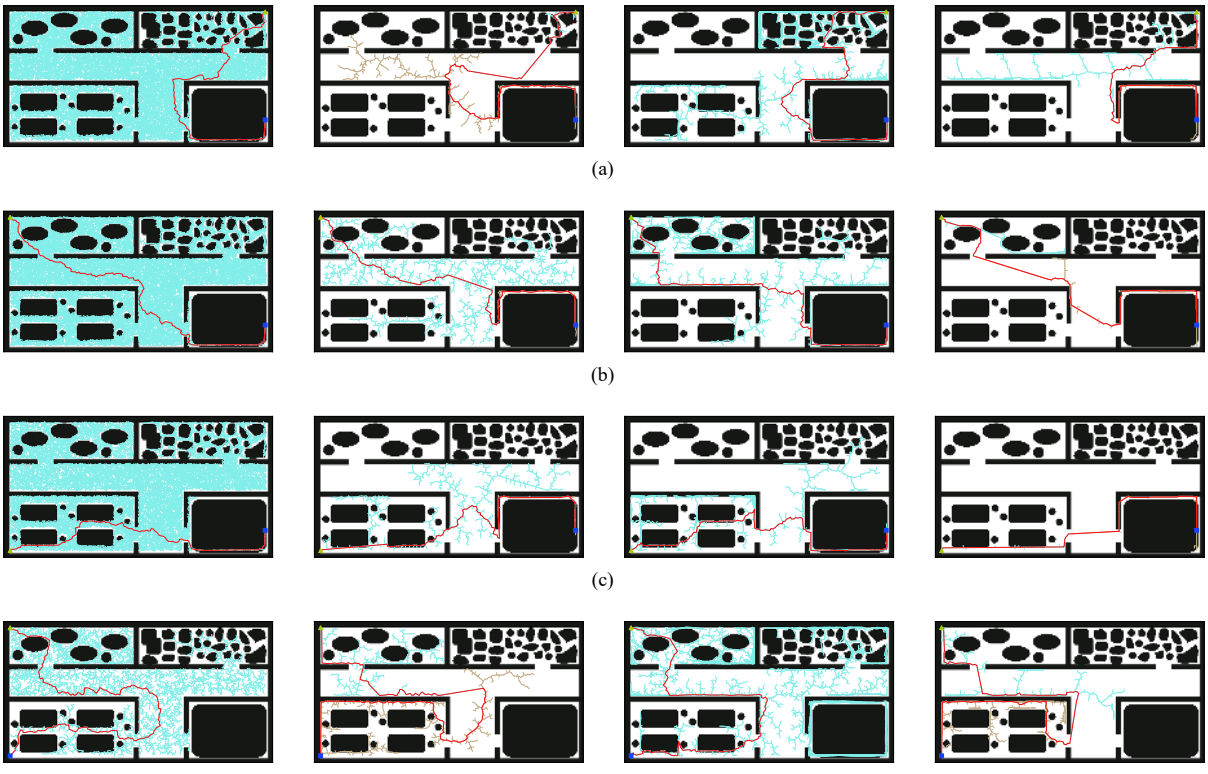


Fig. 8. Planning results of four algorithms for office map. From left to right in each row are the planning results of RRT, RRT-Connect, RRV and ARRT-Connect algorithms. (a) Room A to D; (b) Room B to D; (c) Room C to D; (d) Room B to C.

REFERENCES

[1] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[2] S. M. Lavalle, "Rapidly-exploring random trees : A new tool for path planning," Iowa State University, Tech. Rep. 98-11, Oct. 1998.

[3] X. Tang and F. Chen, "Robot path planning algorithm based on bi-rrt and potential field," in *Proc. IEEE Int. Conf. Mechatronics and Automation (ICMA)*, 2020, pp. 1251–1256.

[4] C. Yuan, W. Zhang, G. Liu, X. Pan, and X. Liu, "A heuristic rapidlyexploring random trees method for manipulator motion planning," *IEEE Access*, vol. 8, pp. 900–910, 2020.

[5] W. Xinyu, L. Xiaojuan, G. Yong, S. Jiadong, and W. Rui, "Bidirectional potential guided RRT* for motion planning," *IEEE*

Access, vol. 7, pp. 95046–95057, 2019.

[6] H. Zhang, Y. Wang, J. Zheng, and J. Yu, "Path planning of industrial robot based on improved RRT algorithm in complex environments," *IEEE Access*, vol. 6, pp. 53296–53306, 2018.

[7] A. Tahirovic and M. Ferizbegovic, "Rapidly-exploring random vines (RRV) for motion planning in configuration spaces with narrow passages," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018, pp. 7055–7062.

[8] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. ICRA Millennium Conf. IEEE Int. Conf. Robotics and Automation*, 2000, vol. 2, pp. 995–1001.

[9] D. Hsu, Tingting Jiang, J. Reif, and Zheng Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2003, vol. 3, pp. 4420–4426.

[10] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle, "Dynamicdomain RRTs: Efficient exploration by controlling the sampling domain," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2005, pp. 3856–3861.

[11] S. Khanmohammadi and A. Mahdizadeh, "Density avoided sampling: An intelligent sampling technique for rapidly-exploring random trees," in *Proc. Eighth Int. Conf. Hybrid Intelligent Systems*, 2008, pp. 672–677.

[12] Liangjun Zhang and D. Manocha, "An efficient retraction-based RRT planner," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2008, pp. 3743–3750.

[13] S. Rodriguez, S. Thomas, R. Pearce, and N. M. Amato, "RESAMPL: A region-sensitive adaptive motion planner", Algorithmic Foundation of Robotics VII. Berlin, Heidelberg: Springer, 2008, pp. 285–300.

[14] A. Shkolnik and R. Tedrake, "Sample-based planning with volumes in configuration space," *arXiv preprint arXiv:1109.3145*, 2011.

[15] M. Strandberg, "Augmenting RRT-planners with local trees," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2004, vol. 4, pp. 3258–3262.

[16] Rodriguez, Xinyu Tang, Jyh-Ming Lien, and N. M. Amato, "An obstaclebased rapidly-exploring random tree," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2006, pp. 895–900.

[17] S. Dalibard and J.-P. Laumond, "Linear dimensionality reduction in random motion planning," *Int. J. Robotics Research*, vol. 30, no. 12, pp. 1461–1476, 2011.

[18] J. Lee, O. Kwon, L. Zhang, and SE. Yoon, "SR-RRT: Selective retraction-based RRT planner," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 2543–2550.

[19] W. Wang, L. Zuo, and X. Xu, "A learning-based multi-RRT approach for robot path planning in narrow passages," *J. Intelligent & Robotic Systems*, vol. 90, no. 1−2, pp. 81–100, 2018.

[20] T. Lai, F. Ramos, and G. Francis, "Balancing global exploration and local-connectivity exploitation with rapidly-exploring random disjointedtrees," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 5537–5543.

[21] T. Lai, P. Morere, F. Ramos, and G. Francis, "Bayesian local samplingbased planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1954–1961, 2020.

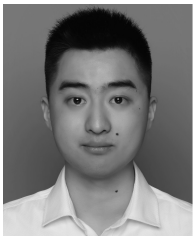
[22] P. Rajendran, S. Thakar, A. M. Kabir, B. C. Shah, and S. K. Gupta, "Context-dependent search for generating paths for redundant manipulators in cluttered environments," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2019, pp. 5573– 5579.

[23] S. R. Lindemann and S. M. LaValle, "Incrementally reducing dispersion by increasing voronoi bias in RRTs," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2004, vol. 4, pp. 3251–3257.

[24] S. R. Lindemann and S. M. LaValle, "Steps toward derandomizing RRTs," in *Proc. 24th Int. Workshop on Robot Motion and Control (IEEE)*, 2004, pp. 271–277.

[25] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD International Conf. Management of Data*, 1984, pp. 47–57.

[26] L. G. D. O. Véras, F. L. L. Medeiros, and L. N. F. Guimaráes, "Systematic literature review of sampling process in rapidly-exploring random trees," *IEEE Access*, vol. 7, pp. 50933–50953, 2019.



Binghui Li received the B.S. degree in automation and the M.S. degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2018 and 2021, respectively. His research focuses mainly on intelligent systems and path planning.

Badong Chen (*Senior Member, IEEE*) received the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2008. He was a Post-Doctoral Associate with the Computational NeuroEngineering Laboratory, University of Florida, Gainesville, FL, USA, from 2010 to 2012. He is currently a Professor with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China. His research interests include machine learning, artificial intelligence, neural engineering and robotics. He has authored or coauthored over 200 articles in various journals and conference proceedings. He serves (or has served) as a Technical Committee Member of IEEE Signal Processing Society Machine Learning for Signal Processing (MLSP), and a Technical Committee Member of IEEE Computational Intelligence Society Cognitive and Developmental Systems (CDS), and an Associate Editor (or Editor Board Member) for 7 international journals including *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Cognitive and Developmental Systems*, *Neural Networks*, *Journal of The Franklin Institute*, *and Entropy*.