Proceedings of the 38th Chinese Control Conference
July 27-30, 2019, Guangzhou, China

A Probability Adaptive Sampling-based Algorithm for Obstacle Avoidance Motion Planning Problems

Kai Mi[1,2], Jun Zheng[1], Yunkuan Wang[1]

1. Institute of Automation, Chinese Academy of Sciences, Beijing 100190
E-mail: jun.zheng @ia.ac.cn

2. University of Chinese Academy of Sciences, Beijing 100049, China
E-mail: mikai2015@ia.ac.cn

Abstract: In recent years, with the increasingly complex robot application environments, the research of robotic autonomous obstacle avoidance motion planning is one of the key technologies to improve its intelligence. In this paper, we present an improved algorithm based on Rapidly-exploring Random Tree (RRT) algorithm, which is called Probability Adaptive RRT (PARRT). We analyze the defects of the original algorithm in detail from the perspective of Voronoi diagram. For some complicated occasions, especially when the robot is surrounded by obstacles, the efficiency of the original algorithm will be greatly reduced. For this problem, we proposed an expansion probability adaptive method. By dynamically adjusting the extending probability of nodes near obstacles, the invalid node expansion is reduced effectively and the planning speed is greatly improved. Finally, we experiment in both two-dimensional space and the joint space of manipulators. The planning results show the improved effect of the proposed algorithm.

Key Words: Rapidly-exploring Random Tree, Voronoi diagram, motion planning, extending probability, probability adaptive

1 Introduction

The traditional path planning algorithms include artificial potential field method [1], D*-search algorithm [2] and so on. Because of the need to establish a complete configuration space model of obstacles, the computational complexity will increase exponentially with the increase of the configuration space dimension. Especially for the artificial potential field method, in [3], a multi-point potential field (MPPF) method is proposed for real-time obstacle avoidance. However, it will easily fall into local extremum. Some intelligent algorithms have also been applied to robot motion planning. For example, genetic algorithm has been used in [4] which could plan a high smooth and time-optimal trajectory for manipulators. Long calculation time is the biggest flaw for such algorithms. Sampling-based motion planning is essentially different. The obstacle information can be obtained by sampling the configuration space with a collision detector [5] and there is no need to establish a complete obstacle model in configuration space.

Sampling-based planning algorithms are usually divided into two contexts: single-query planning and multiple-query planning. The multiple-query planner is represented by Probabilistic Roadmap (PRM) planner [6], and the map information needs to be preprocessed and stored in a specific data structure. The single-query planner is represented by Rapidly-exploring Random Tree (RRT) planner [7, 8] and there is no need for preprocessing. It is suitable for solving the problem of redundant path planning with kinematic and dynamic constraints in complex environment. But the algorithm itself still has some shortcomings. For robots with differential constraints in high-dimensional configuration space, Lavalle et al. proposed a control-based RRT algorithm (Kinodynamic RRT) [9]. They applied robot kinematics to path generation and generated an effective motion trajectory directly. In order to improve search efficiency, the RRT-connect algorithm is developed in [10], where two trees are generated in parallel from the initial state and the target state. Then in order to improve the quality of the generated path, the heuristically-guided RRT (hRRT) has been proposed in [11]. The cost value for each state point in the C-space should be set and the heuristic searching is aimed at the minimum cumulative cost. Then in [12], the Transition-based RRT planner (T-RRT) is proposed, which is designed to combine the rapid exploration properties of RRT with stochastic global optimization methods. In order to obtain the asymptotic optimality, Karaman et al. proposed RRT* algorithm [13], which introduced an optimization structure based on the RRT algorithm. When a new node is added to the tree, it will optimize the structure of the tree to ensure that the path from the initial node to the arbitrary node is optimal in the current tree. As the iteration continues, the final path tends to be optimal. For special applications such as complex environments with narrow channels, RRT algorithms based on obstacle boundaries [14], and RRT algorithms which are biased to narrow spaces [15, 16] have been proposed.

The RRT planner is essentially a random search algorithm, and is lack of the ability to adjust the sampling domain of nodes. In some cases, especially when the robot is surrounded by obstacles, the sampling domain is not well suitable for the problem. To solve it, an improved algorithm called Dynamic-Domain RRTs have been proposed in [17]. The method is to limit the sampling domain of the nodes near obstacles and set it to a circle or a sphere with a specific radius. However the radius is the same for every nodes near obstacles, it is not conducive to the expansion of effective boundary nodes. Based on the basic RRT algorithm, this paper proposes a method to dynamically adjust the extending probability of boundary nodes and has applied it to the basic RRT and RRT-Connect algorithm.

The paper is organized as follows. Firstly, we introduce the concept of Voronoi region and present the problem we

* This work is supported by Major Science and Technology Project of Henan Province under Grant No. 161100210300.

Authorized licensed use limited to: University of Chinese Academy of SciencesCAS. Downloaded on June 10,2020 at 02:43:14 UTC from IEEE Xplore. Restrictions apply.

plan to solve in Section 2. The Section 3 is divided into two parts. The first part mainly elaborates the improved algorithm, including the principle of algorithm. Then a simple proof of probability completeness of the proposed improved algorithm is given in the second part. In Section 4, we experimented in both two-dimensional space and the joint space of manipulator. The planning results show the improved effect of the proposed algorithm. Finally, conclusions and some improved ideas are outlined in Section 5.

2 Preliminary Material

2.1 Voronoi Diagram and Motivating Example

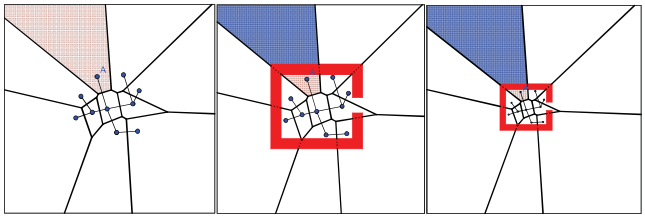


Fig. 1: This is the Voronoi diagram of the nodes in random tree. The image (a) is a situation without obstacles. The image (b) and (c) have different environment sizes and the red area in them is obstacles.

The Voronoi diagram consists of many continuous polygons which are composed of vertical bisectors for two adjacent dots. The characterization of Voronoi diagram is that for any point within the polygon, the distance from the point to the sample point of corresponding polygon is closest, and each polygon only contains one sample point. In Fig. 1(a), the point A is the sample point for the yellow area polygon and the yellow area is its Voronoi region. If a random node falls into the yellow area, node A will be expanded. So the area of Voronoi region determines the probability that a node can be chosen for an extension. For the basic RRT algorithm, only the node in the tree that is closest to the random node can be selected for expansion. So each node in the random tree *T* has a corresponding Voronoi region. The probability that a node can be chosen for extension depends on the area size of its Voronoi region. In Fig. 1(b), the red area is the obstacles. Before discussion, we give some key definitions.

Definition 1: Given a *C*-space with obstacles and a random tree, define the *boundary node* as a node which has been failed to extend at least one time because of the obstacles (e.g. the node A in Fig. 1(b)).

Definition 2: The *invalid region* of a *boundary node* is the area belonging to the Voronoi region of the *boundary node*. When a random node falls into the *invalid region*, the corresponding *boundary node* cannot be extended because of obstacles (e.g. the blue area in Fig. 1(b)).

Definition 3: Different from the *invalid region*, the *visibility region* of a *boundary node* is the area where when a random node falls into, the corresponding *boundary node* can be extended (e.g. the yellow area in Fig. 1(b)).

In Fig. 1(b), the task is to move the robot outside the obstruction of obstacles. It is obvious that most of the points around the obstacles have a larger area of Voronoi region. However because of the existence of the obstacles, the extension for *boundary nodes* is often invalid. Take point A as an example, it is clear that the *invalid region* occupies the vast majority of the sampling area. When random nodes fall into the *invalid region*, they will consume a lot of invalid collision detection time.

It will be worse when the sampling region is enlarged in Fig. 1(c). The area of Voronoi region for the points around the obstacles becomes larger and the points near the opening have a smaller probability of being selected for extension. Meanwhile, the size and position of the obstacles are the same, so only the *invalid area* is expanded for the *boundary nodes* .These make the problem worse. To solve the problem, we propose a method to dynamically adjust the extending probability of *boundary nodes*. It will be described in detail in the next section.

2.2 Problem Formulation

Let C be an n-dimensional configuration space and n will be six for 6-DOF arm robot system. Let C_{obs} be the set of obstacles in the space. Correspondingly, $C \setminus C_{obs}$ will be the obstacle-free space which is called C_{free} . Let q_{init} and q_{goal} be two elements of C_{free} and they represent the initial state and goal state respectively.

Problem: For an arm robot, given a n-dimensional configuration space C , an obstacle space $C_{obs} \subset C$, an initial state $q_{init} \in C_{free}$, and a goal state $q_{goal} \in C_{free}$, compute a continuous path σ from q_{init} to q_{goal} in C_{free} ($\sigma \subset C_{free}$), especially for the environment where the sampling domain is not well suitable .

3 Algorithm Description

In this section, it will be divided into two parts to present. Firstly, an improved algorithm based on the basic RRT will be proposed to solve the problem which is raised above. Then, a simple proof of probability completeness of the proposed improved algorithm is given.

Considering the intuition of narration, we explain the algorithm in two-dimensional space. The principle of the algorithm is also suitable for the high-dimensional configuration space of arm robots.

3.1 Main Algorithm

The pseudocode of Probability Adaptive RRT algorithm is shown in Fig. 2. Compared to the basic RRT algorithm, we have added statistics on the number of successes and failures of node expansion. When the parameter *nFail* is not zero for a node, this means the closest distance between the node to obstacles is less than e . Then we call this node *boundary node*. In general, *boundary nodes* have their Voronoi region growing together with the size of the environment (e.g. the *boundary nodes* in Fig. 1(c)) and they will introduce a large number of invalid extensions. The process of node extension is very time consuming, the reasons are as following. First, it needs to interpolate between q_{near} and q_{new} . Then, collision detection is required for each interpolation point. Finally, for the arm robot, forward kinematics together with interpolation should be performed.

Authorized licensed use limited to: University of Chinese Academy of SciencesCAS. Downloaded on June 10,2020 at 02:43:14 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Probability Adaptive RRT(q_{init}, K, e)

T.init(q_{init})
for $i = 1$ to K **do**
  repeat
    $q_{rand} \leftarrow RANDOM\_STATE()$
    $q_{near} \leftarrow NEAREST\_NEIGHBOR(q_{rand}, T)$
  until $Domain\_Checking(T, q_{near})$
  $q_{new} \leftarrow NEW\_STATE(q_{rand}, q_{new}, e)$
  if $CollisionFreePath(q_{new}, q_{near})$ **then**
    T.Add_Node(q_{new})
    T.Add_Edge(q_{near},q_{new})
    $q_{near}.nSuccess = q_{near}.nSuccess + 1$
  else
    $q_{near}.nFail = q_{near}.nFail + 1$
  end if
end for

Fig. 2: The pseudocode of Probability Adaptive RRT algorithm.

In order to reduce the invalid expansion, we add the judgment function *Domain Checking*. Only when the output of the function *Domain Checking* is true, the program will jump out of the loop to extend a new node.

3.2 Domain Checking

Algorithm 2 *Domain Checking*(T, q_{near})

$(nSuccess, nFail, h, chooseFail) \leftarrow GetCurrentPara(T, q_{near})$
$P_1 = \frac{nFail}{nFail+nSuccess+1}$
$P_2 = \frac{1}{(1+e^{-h})}$
$P = P_1 * P_2$
if $Rand(0,1) > P$ **then**
  $h = h * \alpha$
  return true
else
  if $chooseFail > Failmax$ **then**
    $h = h/\alpha$
    $chooseFail = 0$
  else
    $chooseFail = chooseFail + 1$
  end if
  return false
end if

Fig. 3: The pseudocode of function *Domain Checking*.

The function *Domain Checking* is presented in Algorithm 2. Here we use a method similar to Monte Carlo method. The probability P_1 is the main factor that the corresponding node in the tree is denied expansion and it is defined as:

$$P_1 = \frac{nFail}{nFail + nSuccess + 1} \tag{1}$$

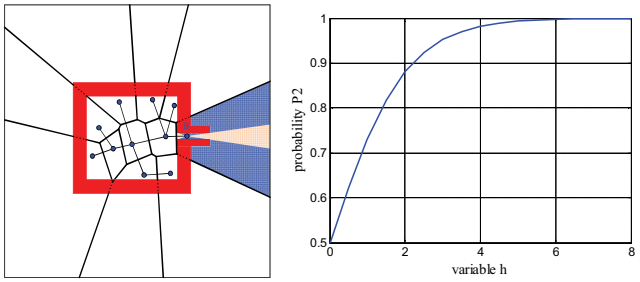
Where *nSuccess* represents the number of times the current node q_{near} has been successfully expanded and *nFail* represents the number of failures. Some cases are presented as following:

- When q_{near} is not *boundary node*, *nFail* will be zero. Then, the probability P_1 will be zero. So q_{near} will always be allowed to extend.
- When q_{near} is a *boundary node*, for the case where the *invalid region* of the node is much larger than *visibility region* (e.g. the node A in Fig. 1(c)), the parameter *nSuccess* may be small or even 0. With the increase of the parameter *nFail*, the probability P_1 will gradually increase from 0 to 1/2, 2/3, etc. P_1 will approach to 1 finally. So q_{near} will have a small probability to be expanded.

In general, with the increase in the number of sampling, P_1 will gradually approach to a ratio ω which is defined as:

$$\omega = \frac{S_{invalid\ region}}{S_{Voronoi\ region}} \tag{2}$$

Where $S_{invalid\ region}$ represents the area of invalid sampling region and $S_{Voronoi\ region}$ represents the total sampling area of current node q_{near}.



(a) (b)

Fig. 4: The image (a) lists a situation where a *boundary node* is a necessary node for the problem, and the image (b) shows the relationship between the auxiliary probability and the parameter h.

Considering some situations with narrow channels, as is shown in Fig. 4(a), the node B at the narrow exit is a *boundary node*. The yellow area is *visibility region* and the blue area is invalid. Obviously the ratio ω for node B is very large, so the probability P_1 will gradually increase. In order to avoid the situation that robot cannot pass through the narrow channel, we add an auxiliary probability P_2. It is defined as:

$$P_2 = \frac{1}{1+e^{-h}} \tag{3}$$

In equation (3), the parameter h is a variable greater than 0. As is shown in Fig. 4(b), with the increase of parameter h, the auxiliary probability P_2 will approach to 1. When h decreases to 0, the probability P_2 becomes 0.5. At the initialization, h is set to a large value in order to reduce the probability of extension for *boundary nodes*. Then, during the exploration, the number *chooseFail* of the *boundary node* will increase when the node is failed to be selected for extension. When the number reaches a maximal number *Failmax*, the parameter h will be divided by a given factor α. Each time the *boundary node* is succeed to be selected for extension, the parameter h is multiplied by the same factor α. The auxiliary probability can effectively improve the fault tolerance of the algorithm and ensure the probability completeness.

Authorized licensed use limited to: University of Chinese Academy of SciencesCAS. Downloaded on June 10,2020 at 02:43:14 UTC from IEEE Xplore. Restrictions apply.

3.3 Probabilistic Completeness

The probabilistic completeness for the basic RRT planner has been confirmed in the section 4 of [6]. The new algorithm presented above is based on the basic RRT planner, and it is also a probabilistically complete planner. This property is directly inherited from the basic RRT planner. The only difference is that in the new algorithm, the probability that a node is extended is dynamically adjusted because of the function *Domain checking*. However, for both the main probability P_1 and the auxiliary probability P_2, the values of them are always between 0 and 1 strictly. So the probability of success for function *Domain Checking* is strictly positive. The same as basic RRT planner, the new planner will converge to an entire coverage of the reachable free configuration space.

4 Experimental Results

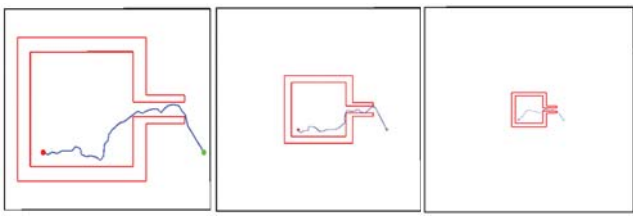
We have implemented our algorithm in C++ and incorporated it into an open source movement planning platform Open Motion Planning Library (OMPL) in MoveIt which is based on ROS. All simulations are run on a personal computer with 3.7 GHz Intel CPU and 8 GB memory under the Windows 7 operating system.

In order to prove the validity and versatility of the algorithm, we have experimented in both 2D space and arm robot space. Some given parameters shown in Table 1 are the same for both occasions. We apply our method not only to the basic RRT planner but also to the RRT-Connect planner. So, PARRT and PARRT-Connect represent the probability adaptive algorithm based on RRT and RRT-Connect respectively. For each of the experiments, we show the running times, the number of nodes in the final path and the number of times the collision detection is requested during the searching process.

Table 1: The given parameter of the improved algorithm for simulation

Given parameter	K	α	h(init)	*Failmax*
Value	50000	2	1024	10

4.1 Simulation in 2D Space



(1) (2) (3)

Fig. 5: The goal is to move a point robot out of the trap, and different environment sizes are shown in (1), (2) and (3).

In order to reflect the effectiveness of the algorithm, we experiment in different sizes of the environment which is shown in Fig. 5. The red point is initial node and the green is target node. The sides of the environment in Fig. 5(1) (2) and (3) are 160, 300 and 600, respectively. Experiments for each algorithm in each environment have been carried out 100 times. The step length for each experiment is 4 and the probability of target bias for RRT and PARRT planner is 0.05.

Table 2: The results for different environment sizes

	Basic RRT	Our method for RRT	RRT-Connect	Our method for RRT-Connect
time(1)	2.896 s	**2.039 s**	1.295 s	**1.065 s**
num.nodes (1)	48	49	51	52
num.cd(1)	1649	**1198**	735	**543**
time(2)	3.566 s	**1.027 s**	2.367 s	**1.338 s**
num.nodes (2)	50	50	54	57
num.cd(2)	2119	**510**	1287	**753**
time(3)	7.302 s	**1.962 s**	6.341 s	**2.058 s**
num.nodes (3)	52	49	53	56
num.cd(3)	4660	**920**	3120	**1043**

In Table 2, the numbers (1) (2) and (3) denote the experimental results in three different environments. In comparison, we can get the following conclusions. First, the number of nodes in the final path for different algorithm varies little. Then, compared to the basic RRT planner, the RRT-Connect algorithm has a better performance. Last, due to the reduction in the number of collision detection, the probability adaptive algorithms have less running time than the old algorithms and the effect is more obvious with the increase of the environment size.

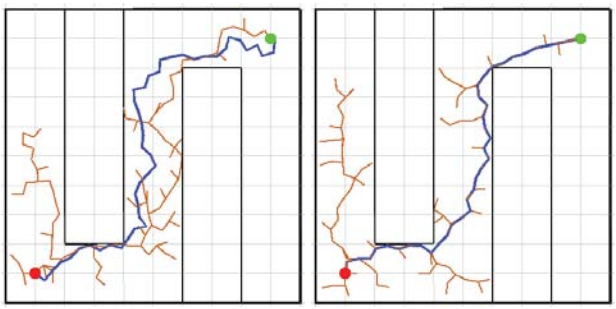


Fig. 6: The goal is to move a point robot from the red point to the green point. The left image is planned by basic RRT algorithm and the right is planned by the proposed PARRT algorithm.

Table 3: The performance data for different algorithms

	Time /s	Num. nodes	num.cd	Tree size
Basic RRT	0.48	42	279	148
PARRT	**0.444**	41	255	**176**
RRT-Connect	0.267	39	136	68
PARRT-Connect	**0.221**	41	130	**84**

Authorized licensed use limited to: University of Chinese Academy of SciencesCAS. Downloaded on June 10,2020 at 02:43:14 UTC from IEEE Xplore. Restrictions apply.

Considering path planning in narrow obstacle environment, there are many boundary nodes and the planned path must pass through most of them. In order to avoid the expansion probability of some necessary boundary points being too low, we introduce an auxiliary probability as mentioned in the section 3.2. Compared with the basic RRT algorithm, the proposed algorithm only increases the time of probability judgment and it is rarely negligible compared with the time-consuming of collision detection. Now we add an experiment to prove it.

For a typical labyrinth environment, the path planned by the proposed algorithm is not much different from that of the basic RRT algorithm as shown in Figure 6. We have performed 100 tests on each algorithm and the average performance data is shown in Table 3. Compared with the basic RRT algorithm, the proposed improved algorithm has little difference in the performance data under the narrow obstacle environment. No matter the planning time or the number of collision detections, there is little difference. Therefore, this mechanism of probability adaptation does not adversely affect path planning in a narrow obstacle environment.

At the same time, we found an important phenomenon that with fewer collision detection times, the improved algorithm can extend more nodes in the rapidly-exploring random tree as shown in the last column of Table 3. Whether for RRT or RRT-Connect, the probabilistic adaptive approach extends more nodes in the tree. This means the proposed probabilistic adaptive algorithm is faster for the exploration of obstacle-free space C_{free} compared to the basic RRT algorithm.

4.2 Simulation in Arm Robot Space

Table 3: The constraints of each joint

	J1	J2	J3	J4	J5	J6
θ (rad)	-2.967 ,2.967	-2.094 ,2.094	-2.182 ,2.705	-4.712 ,4.712	-2.094 ,2.094	-6.283 ,6.283
$\dot{\theta}$ (rad/ s)	0.7	0.7	0.7	0.4	0.7	0.7
$\ddot{\theta}$ (rad/ s2)	0.7	0.7	0.7	0.5	0.7	0.7

In order to make the simulation model more real, we use the Japanese DENSO robot as a simulation object. It is an arm robot with six joints. Table 3 shows several constraints of robot joints including the joint movement range, the maximum joint angle speed and the maximum joint angle acceleration.

In Fig. 7(1), we build a simulation environment in MoveIt using the URDF file of DENSO robot and then add some obstacles. The task is to take something from the green shelf as is shown in Fig. 7(2). We have added the new algorithms PARRT and PARRT-Connect to the OMPL framework. Considering the existence of obstacles, the excess step size will reduce the success rate of the motion planning. So we set the step size to 1 and the probability of target bias for RRT and PARRT planner is 0.05. In high-dimensional space, collision detection is very challenging, so we use an open source library called Flexible Collision Library (FCL) [18] which is supported by OMPL.

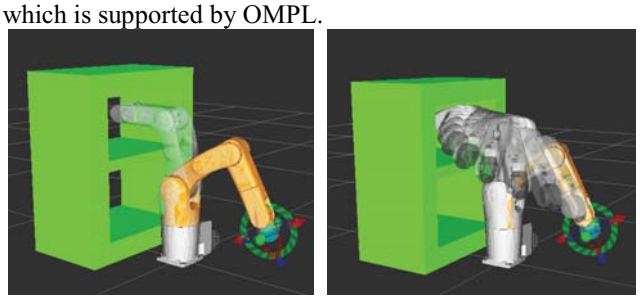


Fig. 7: The task is to take something from the green shelf. The left image shows the initial pose and goal pose for the problem and the right shows the motion process.

Table 4: The results for different algorithms

	Basic RRT	PARRT	RRT-Connect	PARRT-Connect
T /ms	105.5	**61.2**	96.6	**49.5**
num. nodes	21	18	24	21
num.cd	1242	**722**	1068	**572**

Compared with the original algorithms, the specific experimental results are shown in Table 4. It is clear that the probability adaptive algorithm can effectively reduce the planning time. Compared to the basic RRT algorithm, the proposed PARRT algorithm reduces the planning time by nearly 42 percent for the current special environment.

There is something we need to explain. The time for collision detection in 4.1 is longer. The reason is that the algorithm for collision detection in 4.1 is written by ourselves and here we use the mature collision detection library FCL directly. So the efficiency in 4.1 is relatively lower.

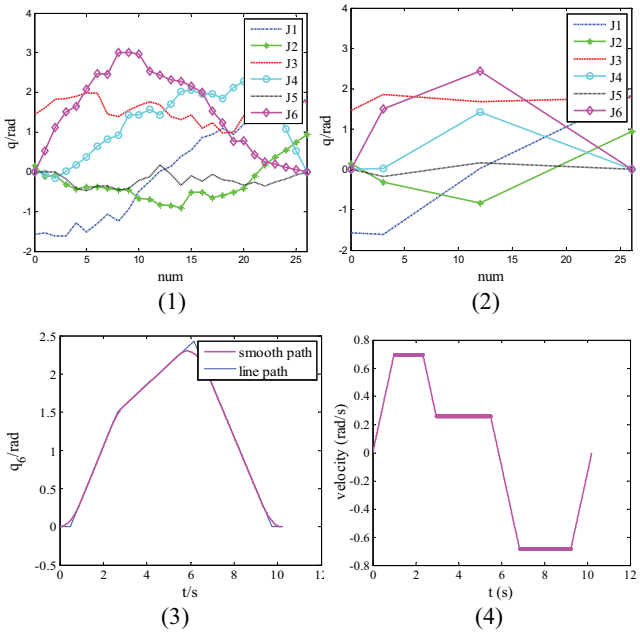


Fig. 8: This is the results of post processing. Different stages of post-processing are represented in image (1) (2) and (3).

Authorized licensed use limited to: University of Chinese Academy of SciencesCAS. Downloaded on June 10,2020 at 02:43:14 UTC from IEEE Xplore. Restrictions apply.

The currently obtained path is only a series of polylines for each joint without any time information. In order to facilitate simulation verification and achieve stable control of the robot, the path needs to be post-processed, including path pruning and path smoothing.

Fig. 8(1) shows the path generated by PARRT in joint space and it has 27 nodes. Fig. 8(2) shows the path after pruning with only four nodes left. Fig. 8(3) and (4) show the trajectories and velocity curves of the robot's sixth joint after smoothing. The trajectories have been planned to meet the following three requirements as much as possible.

- Continuous speed planning. By interpolating parabolic blends near the inflection points, it can achieve a continuous speed planning of each joint alone the path (e.g. Fig. 8(4)).
- Short running time as far as possible. The maximum speed and acceleration which are shown in Table 3 are used as reasonable as possible during each time period.
- Free of collision. To keep the path free of collision, we use the method of combining parabola and linear polynomial to make the path consistent with the original polyline path as much as possible as is shown in Fig. 8(3).

5 Conclusion

In this paper, we have considered the influence of obstacle distribution on RRT algorithm from the perspective of Voronoi region. For some occasions, especially when robots are surrounded by obstacles, the sampling region is inappropriately to be chosen. Then we proposed an expansion probability adaptive method. By dynamically adjusting the expansion probability of the *boundary nodes* in the sampling process, the invalid expansion of the *boundary nodes* is effectively reduced, and the planning speed is greatly improved. Finally, our experimental results show that the path generated by our method is free of obstacles and can be applied to arm robot successfully.

There are several directions to improve the current work. First, the algorithm proposed in this paper can greatly improve the search efficiency for special occasions, but the path generated by the algorithm is not necessarily the best path. Then, to keep the path free of collision as much as possible, we adopt the method of combining parabola and linear polynomial. It could not guarantee the continuity of the joint acceleration.

References

[1] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, *Autonomous Robot Vehicles*, 500-505, 1986.

[2] A. Stentz. Optimal and efficient path planning for partially-known environments, *IEEE International Conference on Robotics and Automation,* vol. 4, pp. 3310-3317, 1994.

[3] S. Saravanakumar, G. Thomas and T. Asokan. Real-time obstacle avoidance for an underactuated flat-fish type autonomous underwater vehicle in 3D space, *International Journal of Robotics and Automation,* 29(4), 2014.

[4] Y. Liu, M. Cong, H. Dong, D. Liu, Y. Du. Time-optimal motion planning for robot manipulators based on elitist genetic algorithm, *International Journal of Robotics and Automation,* 32(4), 2017.

[5] S. R. Lindemann, S. M. Lavalle. Current Issues in Sampling-Based Motion Planning, *Springer Tracts in Advanced Robotics,* 15: 36-54, 2005.

[6] L. E. Kavraki, P. Švestka, J. C. Latombe, M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Transactions on Robotics & Automation,* 12(4): 566-580, 1994.

[7] S. M. LaValle, J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects, *4th International Workshop on Algorithmic Foundations of Robotics.* 2000: 293-308.

[8] S. M. Lavalle. Rapidly-Exploring Random Trees: A New Tool for Path Planning, *Algorithmic & Computational Robotics New Directions*, 293-308, 1998.

[9] S. M. LaValle, J. J. Kuffner. Randomized Kinodynamic Planning. *International Journal of Robotics & Research,* 15(5): 378-400, 1999.

[10] J. J. Kuffne, S. M. LaValle. RRT-connect: An efficient approach to single-query path planning, *IEEE International Conference on Robotics and Automation.* 2000: 995-1001.

[11] C. Urmson, R. Simmons. Approaches for heuristically biasing RRT growth, *International Conference on Intelligent Robots and Systems. IEEE*, vol.2, pp.1178-1183, 2003.

[12] L. Jaillet, J. Cortés, T. Siméon. Sampling-Based Path Planning on Configuration-Space Costmaps. *IEEE Transactions on Robotics,* 26(4): 635-646, 2010.

[13] S. Karaman and E. Frazzoli. Incremental Sampling-based Algorithms for Optimal Motion Planning. *Robotics: Science and Systems,* Zaragoza, Spain, 5326-5332, 2010.

[14] S. Rodriguez, X. Y. Tang, J. M. Lien, N. M. Amato. An obstacle-based rapidly exploring random tree, *IEEE International Conference on Robotics and Automation.* 2006: 895-900.

[15] M. B. Du, J. J. Chen, P. Zhao, H. W. Liang, Y. Xin, T. Mei. An improved RRT-based motion planner for autonomous vehicle in cluttered environments, *IEEE International Conference on Robotics and Automation.* 2014: 4674-4679.

[16] J. Lee, O. Kwon, L. J. Zhang, S. Yoon. SR-RRT: Selective retraction based RRT planner, *IEEE International Conference on Robotics and Automation.* 2012: 2543-2550.

[17] A. Yershova, L. Jaillet, T. Simeon, S. M. LaValle. Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain, *IEEE International Conference on Robotics and Automation. IEEE,* 2006: 3856-3861.

[18] J. Pan, S. Chitta and D. Manocha. FCL: A general purpose library for collision and proximity queries. *IEEE international conference on robotics and automation*, 2012: 3859-3866.

Authorized licensed use limited to: University of Chinese Academy of SciencesCAS. Downloaded on June 10,2020 at 02:43:14 UTC from IEEE Xplore. Restrictions apply.