TP4-Path Planning using RRT

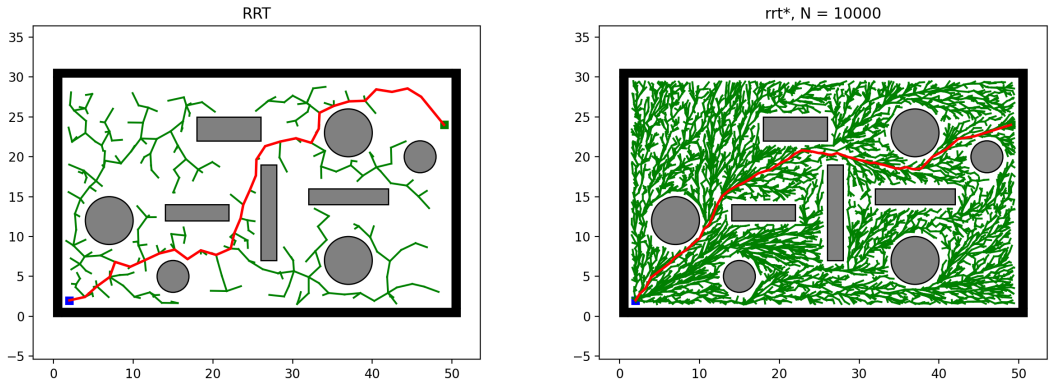
Yan CHEN

December 2020

1 Introduction

In this TP, we discuss these two algorithms RRT and RRT* with comparing their performance. And the planification in narrow corridors is considered.

2 RRT vs RRT*



(a) (b)

FIGURE 1 – RRT and RRT* with step length=2

	RRT		RRT*	
	Computation Time / s	Length Of Path	Computation Time / s	Length Of Path
1	0.32	80.52	58.90	841.51
2	0.14	64.95	57.25	896.70
3	0.53	99.91	58.72	863.39
4	0.18	73.91		
5	0.16	63.93		
aveg	0.27	76.64	58.29	867.20

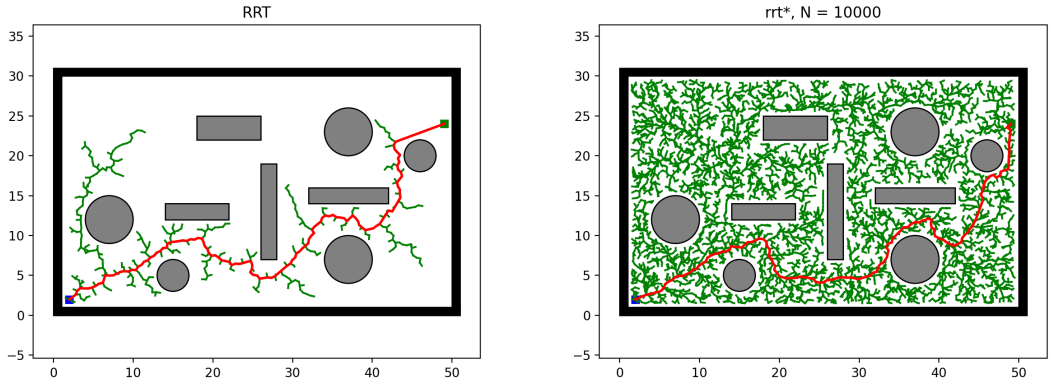
FIGURE 2 – Test of RRT and RRT*

We can see that the average length of the paths is **76.64** and the average computation time is **0.27s** of the RRT algorithm. The average length of the paths is **58.29** and the average computation time is **867.20s** for the RRT* after 10000 iterations. Actually, the RRT* can find goal points after several hundreds iterations. And the average length of the paths of RRT* is just same with RRT at the first time to find the path. But its' computation time is longer than that of RRT. However, with the more iteration, RRT* can find the shorter path. After 10000 iterations, the average shortest length of path is for RRT*. So if we don't consider the shortest path and less computation time, RRT algorithm is a better choice. If we consider the shortest path and don't care about more computation time, RRT* algorithm is a better choice.

We also consider the influence of step length on their performance. We test the two algorithms with **0.5 step length** and **4 step length**. And we compare them with the **2 step length**.

Step length = 0.5

We can see that the average length of the paths is **74.24** and the average computation time is **0.663s** for the RRT algorithm. We can see that the average shortest length of the paths is **72.23** and the average number of iteration that find the goal point at first time is about **2600** for the RRT* algorithm.

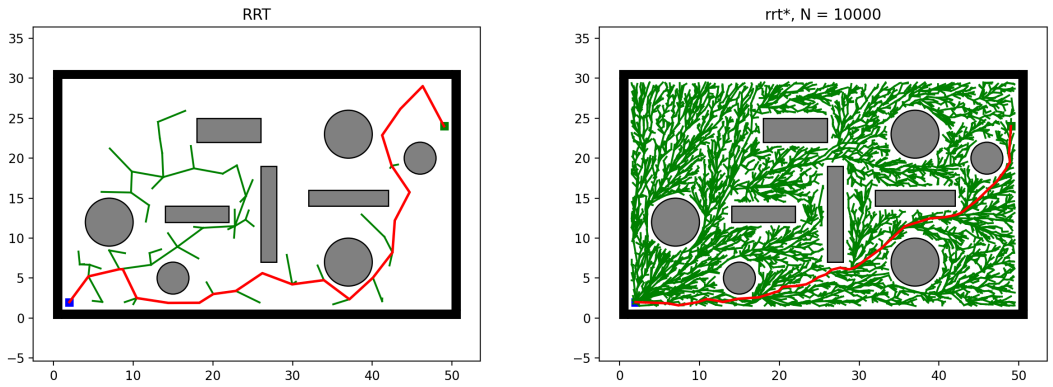


(a) (b)

FIGURE 3 – RRT and RRT* with step length=0.5

Step length = 4

We can see that the average length of the paths is **73,98**and the average computation time is **0.111s** for the RRT algorithm. We can see that the average shortest length of the paths is **57.43** and the average number of iteration that find the goal point at first time is about **350** for the RRT* algorithm.



(a) (b)

FIGURE 4 – RRT and RRT* with step length=4

According to the result above, we can see that smaller step length don't improve the performance of RRT and RRT*. The computation time is longer, the length of path isn't shorter. After 10000 iterations, the length of path is even much longer than that of step length=2 for RRT*. However, bigger step length has reduce the computation time for RRT and less numbers of iteration that find the goal point at the first time for RRT*. And it can have almost shortest path with step length=2 after 10000 iterations. For better performance of RRT*, we obtain the result of RRT* after 20000 iterations. We can see that it finds shorter path after 20000 iterations. It means that more iterations, better performance (length of path) for RRT*.

Test result :

1. Found path, length : 66.91
   Computation time : 4746.18

The aim of RRT is to find the path in less time, so much longer proper step length can improve the performance of RRT. The main aim of RRT* is to find the shortest path. Theoretically, RRT* algorithm can find the shortest path with smaller step length after enough iterations, but needs more computation time.

3 Planification in narrow corridors

In this section, We test the performance of RRT for planification in narrow corridors. We can see that it is difficult to find the goal point. It means that it needs more time to find that.

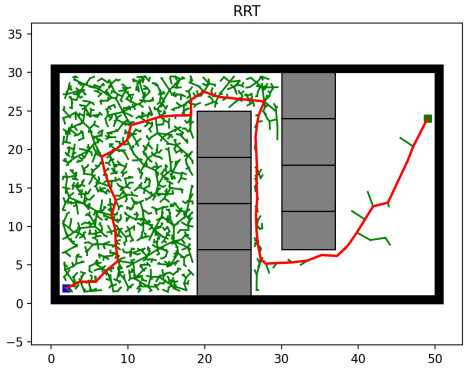


FIGURE 5 – RRT Planification in narrow corridors

Test result :

1. Computation time : 3.53
   Found path in 5317 iterations
   length : 101.10

Actually, it needs lots of time to pass the narrow corridors and can't even find the goal point in 10000 iterations. Because the area of passed zone nearby the narrow corridors is small. These new sampling nodes found nearby the narrow corridors are discarded due to the collision with the obstacles. And we use the random to generate the sampling nodes. In terms of probability, the probability of sampling points that can be sampled without colliding with these obstacles is very small. So it is difficult to find a proper sampling point and even can't find them in short time.

Based on the result above, we sample a part of the points randomly in the obstacle free area around the corners of the obstacles for increasing the probability of passing narrow corridors (a simple variant of the OBRRT algorithm). The zone of sampling is shown in Figure 6. We modify the function ***generate_random_node(self, goal_sample_rate)*** in ***rrt.py***.

```
def generate_random_node(self, goal_sample_rate):
    if np.random.random() < goal_sample_rate:
        return self.s_goal
    delta = self.utils.delta
    node = Node((np.random.uniform(self.x_range[0] + delta, self.x_range[1] -
        delta),
        np.random.uniform(self.y_range[0] + delta, self.y_range[1] - delta)))

    # sample the points around the corners of the obstacle
    if self.utils.is_inside_obs(node) == True and np.random.random() < 0.5:
        return Node((np.random.uniform(self.env.obs_rectangle[4][0], self.env.
            obs_rectangle[4][0]-delta),
            np.random.uniform(self.env.obs_rectangle[4][1], self.env.
                obs_rectangle[4][1]-delta)))
    return node
```

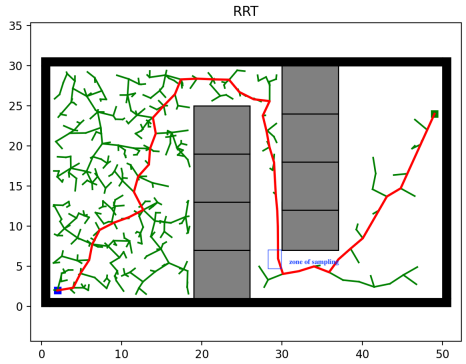


FIGURE 6 – Zone of sampling

According to the result of Figure 7, the probability of passing the narrow corridors to find the goal point by RRT is not much high. But it can pass some times the the narrow corridors to find the goal point by OBRRT [1] with the augmentation of probability of sampling points around the corner of obstacles. When it is **50%**, it has a better performance. Almost it can pass the narrow corridor and find the goal point during 10000 iterations. Some times it needs much less computation time.

	0%		25%		50%		75%		100%	
	Computation Time	Length Of Path	Computation Time	Length Of Path	Computation Time	Length Of Path	Computation Time	Length Of Path	Computation Time	Length Of Path
1	No Path		No Path		7.28	103.66	No Path		6.84	109.93
2	No Path		No Path		0.37	102.39	No Path		0.90	105.73
3	0.53	99.91	5.82	98.43	2.04	106.94	5.34	100.06	No Path	
4	No Path		No Path		2.75	103.98	2.74	101.01	7.61	99.62
5	7.88	106.34	No Path		2.90	101.17	0.46	100.73	3.74	104.68
aveg	4.21	103.13	5.82	98.43	3.07	103.63	2.85	100.60	4.77	104.99

FIGURE 7 – Test of OBRRT

Références

[1] Samuel RODRIGUEZ et al. "An obstacle-based rapidly-exploring random tree". In : *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE. 2006, p. 895-900.