ROB316-TP4

Pan Mengyu

December 2020

1 Introduction

In this TP, we will compare the difference of the Rapidly Exploring Random Trees (RRT) algorithm [3] and RRT*[1] in the different situation. Then we will apply the obstacle-based rapidly-exploring random tree (OBRRT) [2]for some special environment.

2 RRT vs RRT*

2.1 Q1

As we can see in the figure 1, the RRT algorithm chooses node randomly to find a path to the destination, while the RRT* algorithm firstly does the same thing as RRT but it continues to compare the path length to find the best one. In the table 1, the iteration doesn't change much as the RRT algorithm chooses random node. In the table 2, I have tested the iteration that the RRT* algorithm first finds the path and the iteration that path is stable and doesn't change for 5 displays.

Start	End	Iteration	Length
(2,2)	(49, 24)	459	73.73883208
(20,2)	(49, 24)	632	60.02708574
(2,20)	(49, 24)	447	59.22872923

Table 1: Result of RRT for Q1

Start	End	First path Iteration	First path Length	Stable Path Iteration	Stable Path Length
(2,2)	(49, 24)	700	68.39497059	1350	65.81371715
(20,2)	(49, 24)	250	58.74468852	650	55.81492259
(2,20)	(49, 24)	500	48.86548891	1050	45.56361126

Table 2: Result of RRT* for Q1

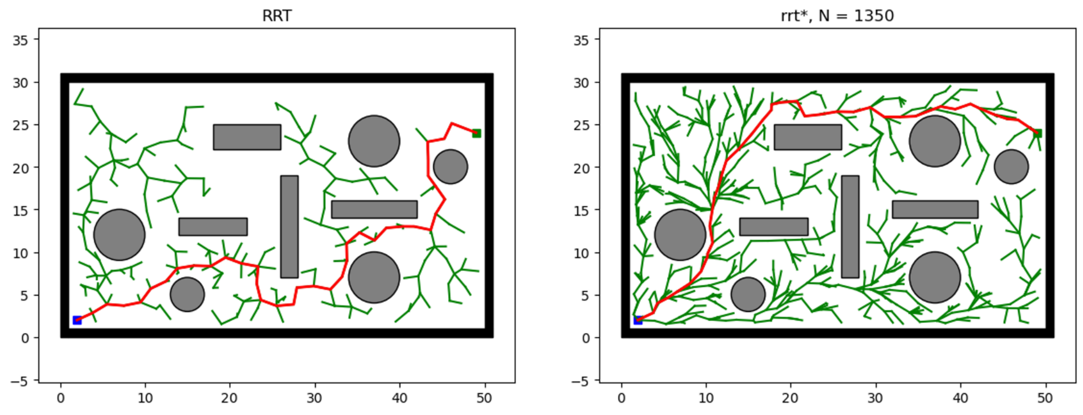

Figure 1: Q1 RRT in the left and RRT* in the right

2.2 Q2

In this question, I remain the start point and end point same as the initial condition but I change the step length in each experiment. For RRT algorithm, in the table 3 big step length can reduce hugely the number of iteration but there is no effect in the path length as this algorithm randomly chooses the node. For RRT* algorithm, step lengths help reduce the iteration both for the first path and for the stable path. However, I cannot say that step length can reduce the stable path length even if in the table 4 stable path length reduces with the increase of step length. The reason is that the stable path may not be the shortest path.

Step Length	Iteration	Length
1	1377	79.20657399
2	459	73.73883208
3	366	68.85983800
5	183	75.13099526
10	30	63.80250779

Table 3: Result of RRT for Q2

3 Planification in narrow corridors

3.1 Q3

The result is showed in the figure 4. In this figure, there is the narrow corridor and the RRT algorithm chooses the node randomly so in the narrow corridor

Step Length	First path Iteration	First path Length	Stable Path Iteration	Stable Path Length
1	1350	70.18982062	2000	69.59809705
2	700	68.39497059	1350	65.81371715
3	650	61.67015504	1400	60.97418566
5	200	69.15282013	700	63.82112585
10	200	70.53657910	500	57.01090296

Table 4: Result of RRT* for Q2

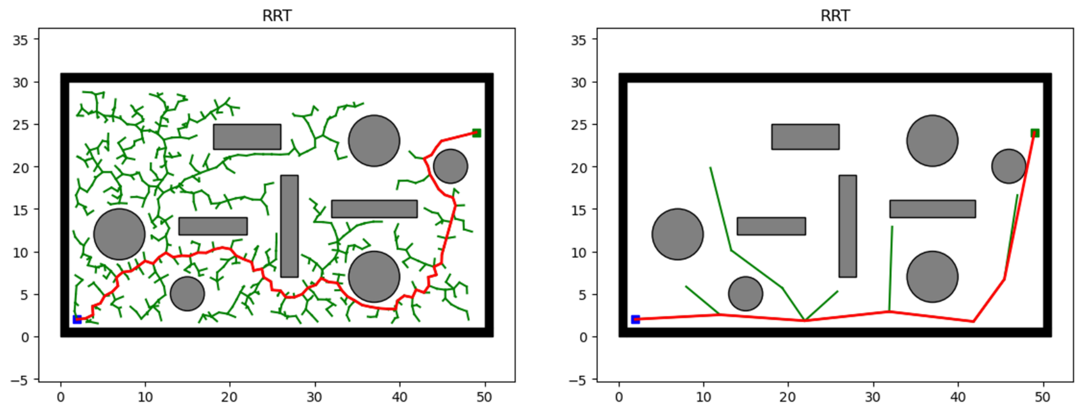

Figure 2: Q2 RRT, step length = 1 in the left and step length =10 in the right

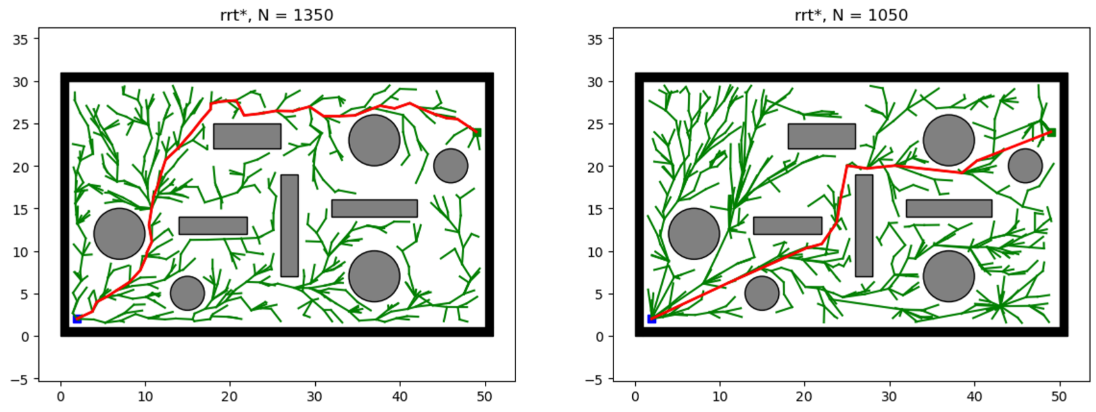

Figure 3: Q2 RRT*, step length = 2 in the left and step length =10 in the right

area it chooses much invalid node thus it has difficulty to pass this area.

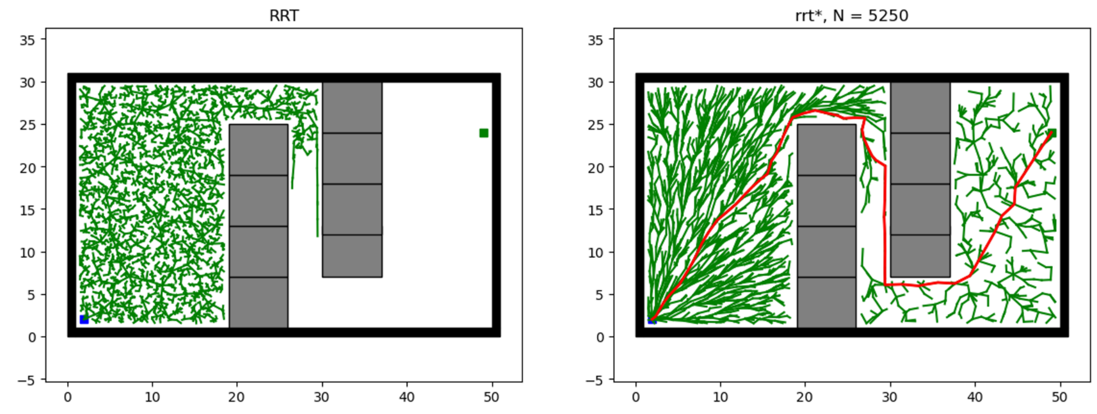

Figure 4: Q3 RRT in the left and RRT* in the right

3.2 Q4

In this case, I apply the OBRRT algorithm, which chooses those nodes who aren't in the obstacle area as the code shows. The result is showed in the figure 5.

```
flag = True
List = self.env.obs_rectangle

while flag:
    x = np.random.uniform(self.x_range[0] + delta,
        self.x_range[1] - delta)
    y = np.random.uniform(self.y_range[0] + delta,
        self.y_range[1] - delta)
    if not self.utils.is_inside_obs(Node((x,y))):
        flag = False
return Node((x,y))
```

References

[1] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *Int.J.Rob.Res*, 30(7), 2011.

[2] Jyh-Ming Lien S. Rodriguez, Xinyu Tang and N. M. Amato. An obstacle-based rapidly-exploring random tree. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2006.

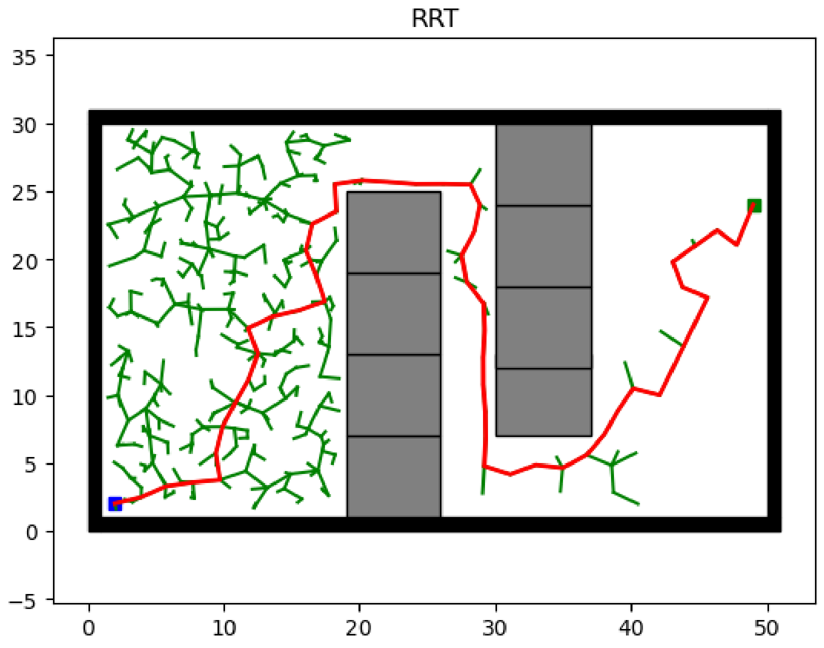

Figure 5: Q4 OBRRT Algorithm

[3] James J. Kuffner Steven M. Lavalle and Jr. Rapidly-exploring random trees : Progress and prospects. *Algorithmic and Computational Robotics : New Directions*, 2000.