ROB316-TP3

Pan Mengyu

December 2020

1 Introduction

In this TP, we will exploit the A* path planing algorithm. The code pathfind.py has 3 types of map and we need to find the nearest path with A* path planing algorithm.

2 Heuristic Influence

2.1 Q1

In the figure 1 the heuristic weight is 1.0 and in the figure 2 the heuristic weight is 0.0. Directly we can see the advantage of the graph with heuristic weight = 0 has bigger area to arrive. However,heuristic weight is related to road choosing and if heuristic weight is 0 thus every road weight is the same and it may spend more time to choose the path as the table 1 shows.

Heuristic Weight	Computing time	Path length
0	2.1037847995758057	685.9655121145942
1	1.3332672119140625	685.9655121145943
5	1.4654147624969482	705.0193359837566

Table 1: Relation between Heuristic Weight, Computing time and Path length

2.2 Q2

In the figure 3 the heuristic weight is 5.0. Compared with the figure 1, it is quick to choose the nearest way but as the table 1 shows, it may also choose the detour as the heuristic weight is too high.

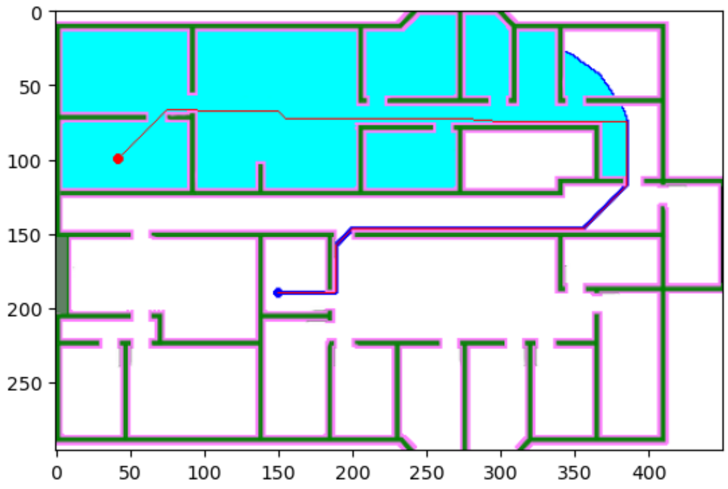

Figure 1: Heuristic Weight = 1.0

3 Environment Influence

3.1 Q3

We compare the influence of three different maps and the results are showed in the figure 4 and figure 5. As we can see in the table2, A* planning algorithm and free choosing (heuristic weight = 0) can both find the best way but A* planning algorithm works very well in the free space with little computing time but it doesn't work well in the labyrinth.

Map	Heuristic Weight	Computing time	Path length
office	0	2.1037847995758057	685.9655121145942
office	1	1.3332672119140625	685.9655121145943
labyrinth	0	1.0833559036254883	785.7472580451152
labyrinth	1	1.1374101638793945	785.7472580451152
free space	0	2.922295570373535	230.37467504308412
free space	1	0.007980585098266602	230.37467504308415

Table 2: Environment Influence

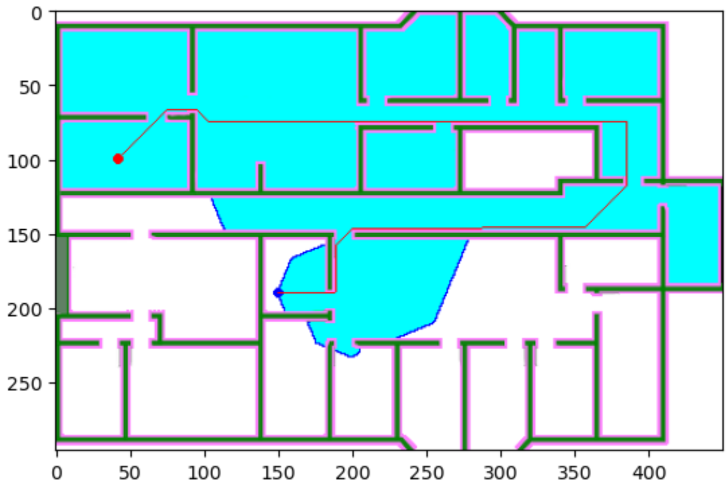

Figure 2: Heuristic Weight = 0.0

4 Weighted nodes

4.1 Q4

In the part, I use the distanceTransform function in opencv to calculate the distance between object and obstacles.

```
distance = cv2.distanceTransform(self.map, cv2.DIST_L2, 3)
```

Then I choose those whose distance is bigger than 3 to get the path and the result is showed in the figure 6.

```
test = (next in closed) or ((next in frontier) and (new_cost >
    cost_so_far[next])) or (distance[next[0],next[1]] < 3)
```

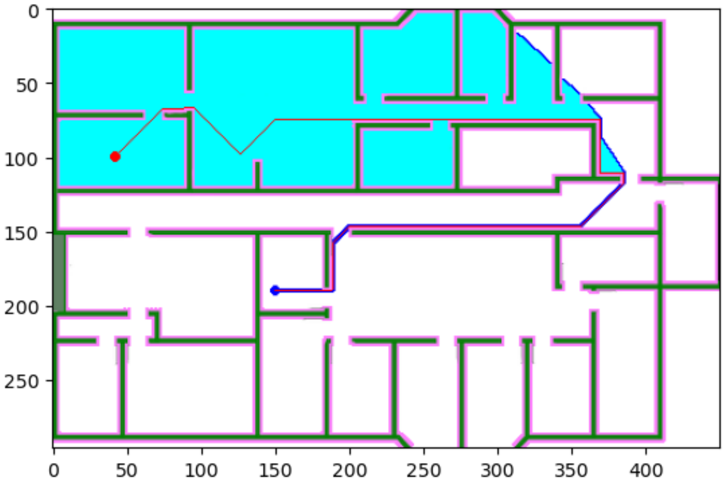

Figure 3: Heuristic Weight = 5.0

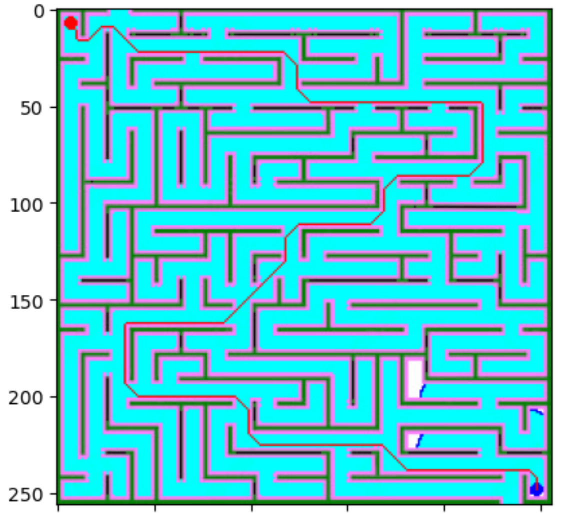

Figure 4: Heuristic Weight = 1.0 in the labyrinth

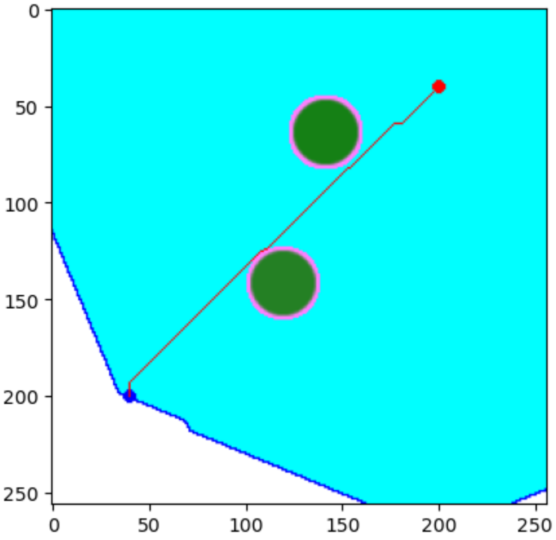

Figure 5: Heuristic Weight = 1.0 in the free space

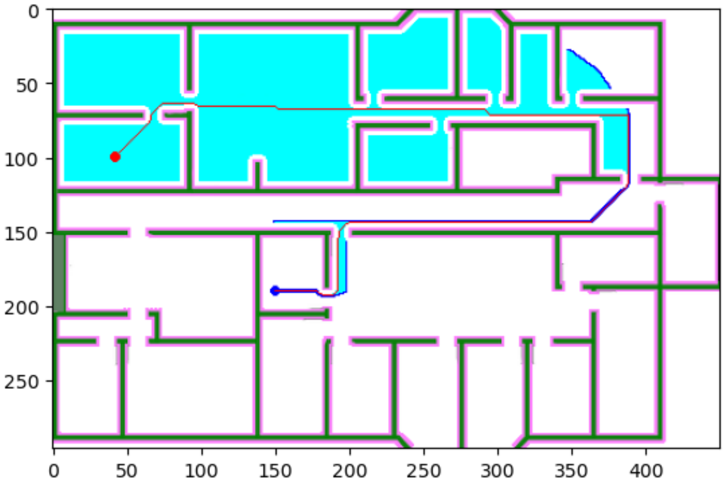

Figure 6: shortest path with a penalty for proximity of obstacles