AI for Robotics-II

Assignment-II | Task Motion Planning

Carmine Miceli – Ecem Isildar – Luca Petruzzello – Baris Aker

Introduction

The purpose of this project is to create the most efficient path for the robot. There is a given map which has a size as 6x6. Robot is located on to the [0,0] point, on each corner points of the map, there are students desks located, such as: 1st desk:[-3,3], 2nd desk:[3,3], 3rd desk:[-3,-3] and the 4th desk:[3,-3] also the teacher's desk is located on [3,0] and these 6 waypoints had been initialized as a regions in *visit_domain/region_poses.txt*. Aim is to create an algorithm which finds the shortest past to the two students' desks to take the homework's and deliver them to the teacher's desk.

Functionality of the semantic attachments

Inside of the *visits_domain/waypoint.txt* file, we manually selected 24 waypoints to be connected to the initial 6 waypoints which are robot's initial position, students' desks and teacher's desk can be seen with their coordinates. Inside of the *visits_domain/edges.txt* file, all the possible connections with the distance between waypoints have been established, with a maximum number of links *k = 5*.
These two files had been called inside of the *visit_module/src/Visit.Solver.cpp* with the functions:

- parseWaypoint(waypoint_file)
- createEdges(edges_file)

The first is responsible to parse the waypoints and store them in a proper data structure, which will be used by the second function in order to compute all the possible links, with their relative distance, between waypoints. In the second case, a graph is created and updated with all the edges in order to allow the planning engine to search for the shortest path.
Then, the planning engine is responsible for calling *VisitSolver::callExternalSolver* where the cost for a specific path is estimated through the function VisitSolver::distance_euc, this function performs the Dijkstra's algorithm on the previously created graph and returns the overall distance of the path.

Results

Two actions were created in the PDDL domain, in order to allow the robot achieve the task:

- *pick_up*
- *drop*

```
; Cost: 26.556
; External Solver: 0.000
; Time 0.02
0.000: (goto_region r2d2 r0 r5)  [100.000]
100.001: (goto_region r2d2 r5 r4)  [100.000]
200.002: (pick_up r2d2 a4 r4)  [0.001]
200.003: (goto_region r2d2 r4 r5)  [100.000]
300.004: (drop r2d2 a4 r5)  [0.001]
300.005: (goto_region r2d2 r5 r1)  [100.000]
400.006: (pick_up r2d2 a1 r1)  [0.001]
400.007: (goto_region r2d2 r1 r5)  [100.000]
500.008: (drop r2d2 a1 r5)  [0.001]
```

Fig.1 Output of the planning engine

In Fig.1, we can observe the cost, i.e. the overall travelled distance, for the robot to collect two assignments and the actions performed.