CS747 - PROGRAMMING ASSIGNMENT- 3 REPORT

NAME - ARPIT DWIVEDI

Roll Number-200100032

Department of Mechanical Engineering

1 *Task 1*

For this task i have coded the controller of my own, where i have decided the control inputs for the car as per following situation:

- First I calculated the required head angle of car to reach exit from simple equation:
$$tan\theta = \frac{0 - car_y}{350 - car_x} \tag{1}$$
- Now I had given the control input of steer to car depending on the car angle at that state and required angle of rotation's sign to decide how to let the car to rotate and align with the direct line connecting car and exit
- This is very intutive controller for the situation of task 1 to let the car drive through the parking and reach the exit
- Now depending on the sign of angle to be rotated to align the car with the line connecting with the target we will give the throttle and steer(positive and negative) to car and move it.

2 *Task 2*

For the task 2, I had divided the prooblem statement in two parts:
1) **Path planning**: The path the car need to traverse to avoid pits and drive through the parking and reach exit
2) **Controller**: Next part is to give control inputs to car to follow the path planned above

2.1 Planning using A*

The A* algorithm has been implemented using the help of online resources whose links have been attached in the references. The A* algorithm avoids obstacle using the cost assigned to each node in the discrete space. The resolution I have chosen is 10. The buffer zone of car has been chosen to be 10 so that we can get the path sufficiently away from the obstacles and wall to avoid the collision while moving near them and also to decide the zone of node in which we can the assign the car to be.
Following are the key points covering how the path planner works:

- First we provide the planner with the start position, goal position, resolution, buffer distance of car and pits centre.
- The Algo will first accumulate all the obstacle nodes with their x and y points as per the coordinates of centre of the pits and the walls
- The next step is to define the obstace map and the possible motions while checking the nest child notes from the parent nodes

- The checking and appending the node with minimum total cost(sum of heuristic and node local cost(movement)) will go on simultaneously till we reach the goal node where the algorithm terminates
- The obtained array of x and y points are passed to controller for further implementation.

2.2 Pure Pursuit LIKE Controller

Following are the some important key things which this controller has in it in addition to the normal nature of pure pursuit:

First we are not going to implement the pure pursuit after the point when the the left path to be traversed is between all the pits that is we can directly drive the car through the straight line connecting the car to the exit.

This is ensured with function *check-clear-path* We will search for the point which is nearby the car in path points and then try to align the car yaw angle with the slop of line connection the car to that target point by giving steer. If the requires angle is less than 3 degrees then we can give the throttle to the car to move forward to chase the point.

The car need to avoid the obstacle with proper distance hence I have increased the virtual region of pits to 150*150 points from 100*100 in path planning.

Now depending on the sign of angle to be rotated to align the car with the line connecting with the target we will give the throttle and steer to car and move it.

To avoid the collision near the walls and pits the limit i had given to speed is 8 so that car can be enough slow near those regions to avoid collisions.

3 References

- https://www.cs.cmu.edu/ motionplanning/lecture/AppH-astar-dstarhowie.pdf
- https://github.com/topics/a-star-algorithm?o=descs=updated
- https://github.com/AtsushiSakai/PythonRobotics
- https://thomasfermi.github.io/Algorithms-for-Automated-Driving/Control/PurePursuit.html