RBE - 550
WildFire

In this assignment I have implemented a planner that utilizes two forms of motion planning algorithms to navigate a fire truck through a maze-like environment which I created in assignment 2. I have tested two types of motion planning algorithms

1 - A* search algorithm

2 - Probabilistic RoadMap Planner.

Environment: I have created the environment and simulated it using pygame. The environment consists of randomly distributed obstacles.

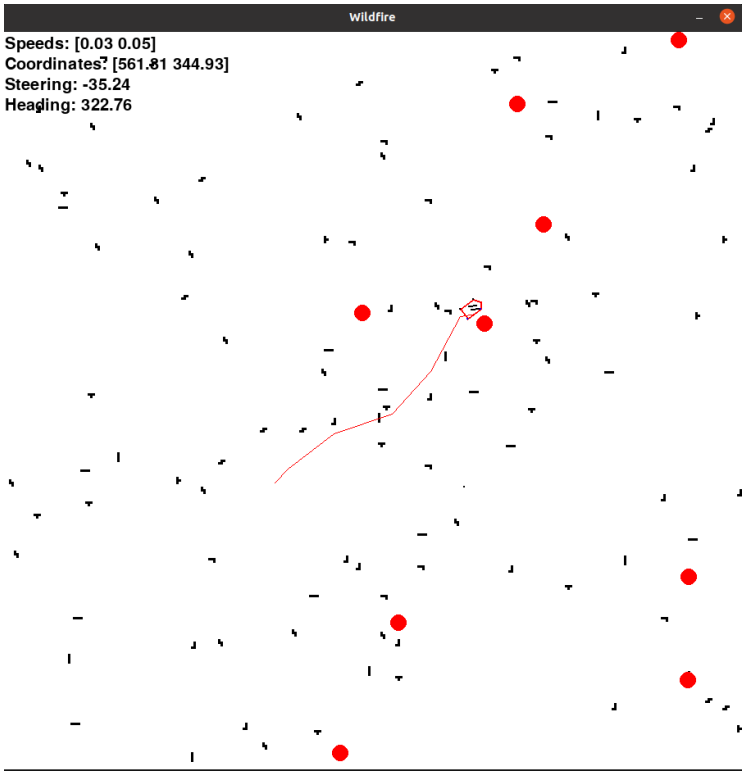

Fig 1: Robot randomly respawned and red spot are the places where the fire is starting to break and everything is in continuous space.

Robot: To make the agent/car more robust. I have created several classes that would take into consideration various parameters of the agent. Like, Controller class, Planner class and Robot class.

Algorithm:

The A* Algorithm[1]: The robot would move to any of the 8 - connected neighbors. The collision checking is performed by observing if any of the robot/agent lines are intersecting/overlapping with the lines of the obstacle. (The pygame.draw.line function made this easily).[2]

To make the code more organized and neat for the user, separate classes were created. A controller class is stepped and the position of the robot/agent is updated in the while loop. Simultaneously, the state of the world and robots is updated and displayed. For the given vehicles they used the same underlying planner i.e. A* Algorithms. A* is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost (least distance traveled, shortest time, etc.). It does this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied.

```
▶       A* (start, goal)
1.      Closed set = the empty set
2.      Open set = includes start node
3.      G[start] = 0, H[start] = H_calc[start, goal]
4.      F[start] = H[start]
5.      While Open set ≠ Ø
6.          do CurNode ← EXTRACT-MIN- F(Open set)
7.          if ( CurNode  == goal ), then return BestPath
8.          For each Neighbor Node N of CurNode
9.              If ( N is in Closed set ), then Nothing
10.             else if ( N is in Open set ),
11.                 calculate N's G, H, F
12.                 If ( G[N on the Open set] > calculated G[N] )
13.                     RELAX(N, Neighbor in Open set, w)
14.                     N's parent=CurNode & add N to Open set
15.             else, then calculate N's G, H, F
16.                     N's parent = CurNode & add N to Open
```

Fig 2: A* Search Algorithm Pseudocode

After the planner returns the trajectory, I feed these waypoints to the robot controller and this moves the robot to the specific waypoints.

Probabilistic RoadMap (PRM): The basic idea for PRM is to take random samples from the configuration space of the agent, testing them for whether they are in the free space, and use a local planner to attempt to connect these configurations to other nearby configurations. The starting and goal configurations are added in, and a graph search algorithm is applied to the resulting graph to determine a path between the starting and goal configurations.

This algorithm is very useful but it does not give the optimal solution every time. PRM inspired from A* Search algorithm.

After the planner returns the trajectory, I gave the waypoints to the agent and this moves the agent to the specific waypoints.

Result:

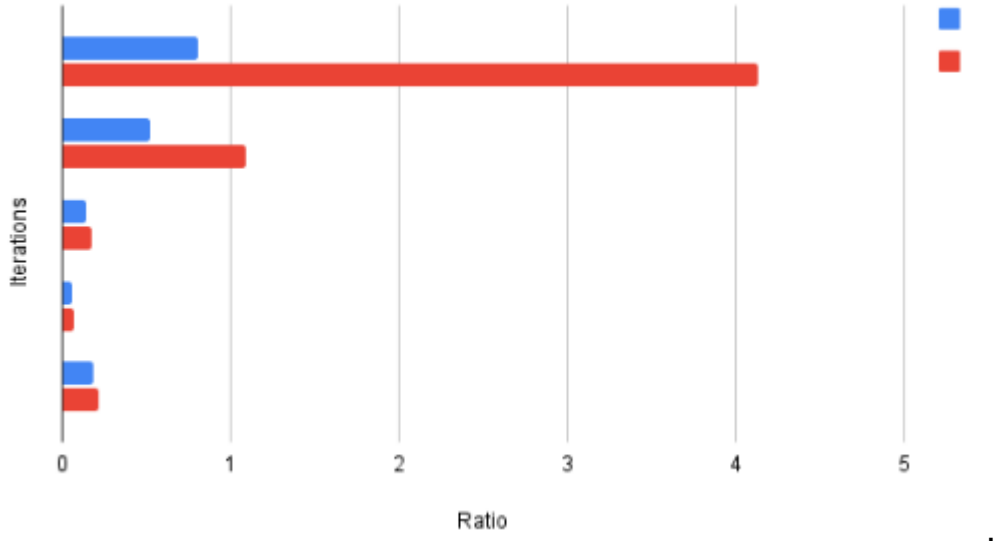

Fig 3: A* Planner: ratio of intact to total, in red: ratio of extinguished to burned

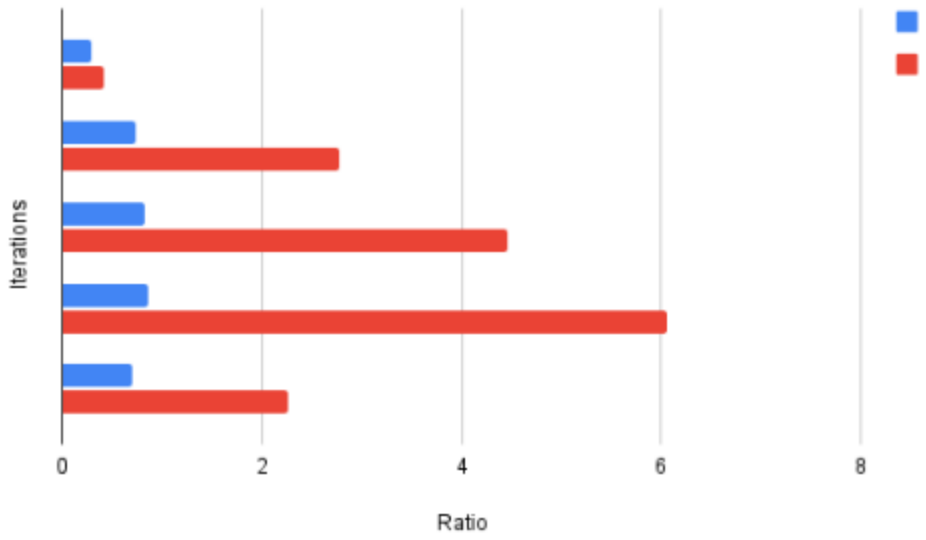

Fig 4: PRM Planner: ratio of intact to total, in red: ratio of extinguished to burned

The PRM performs better in extinguishing the fires than A* search algorithm. Although the PRM does not necessarily plan the most efficient path, it still performs pretty well in extinguishing the fires. A always give us most optimal path. In a static environment computing the roadmap prior is better as there is lesser strain on the CPU to perform real time operations. The PRM method took up more resources in time and computational power.

Reference

[1] Sakai, Atsushi, et al. ”Pythonrobotics: a python code collection of robotics algorithms.” arXiv preprint arXiv:1808.10703 (2018).
[2] How to check if two given line segments intersect?
[3] https://github.com/RitikJain12/Wildfire