Robotics: Planning and Navigation

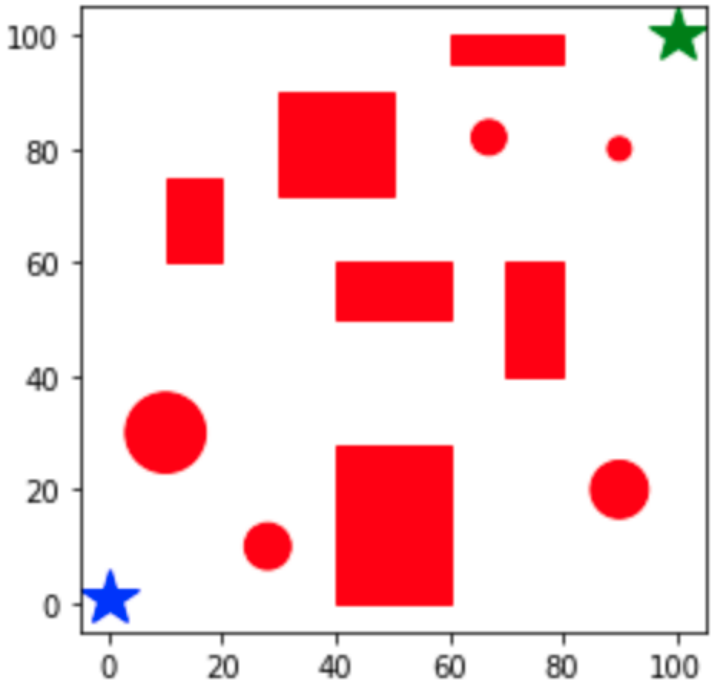
Assignment 1 Report

RRT for Holonomic Drone and Non-holonomic car

Link for Simulation Videos: [Link]

Question-1:

Custom Environment Used:



Summary of RRT:

The robot used here is a differential drive non-holonomic robot which means that the left and right wheel velocities are not coupled. The RRT algorithm is used here to do collision avoidance and reach the goal position. A brief explanation of the algorithm is as follows:

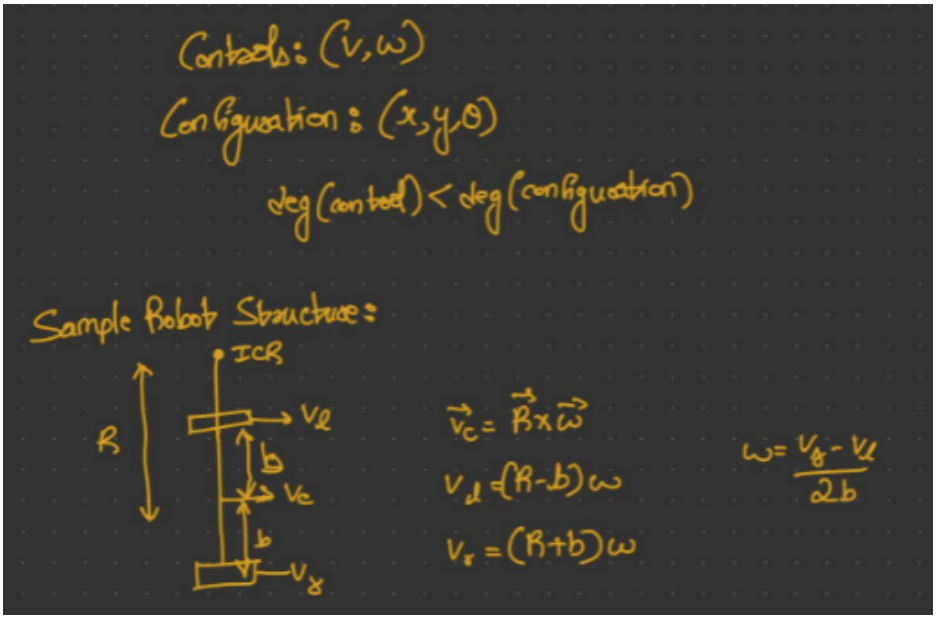
- First we get a random configuration for the robot using the random function.

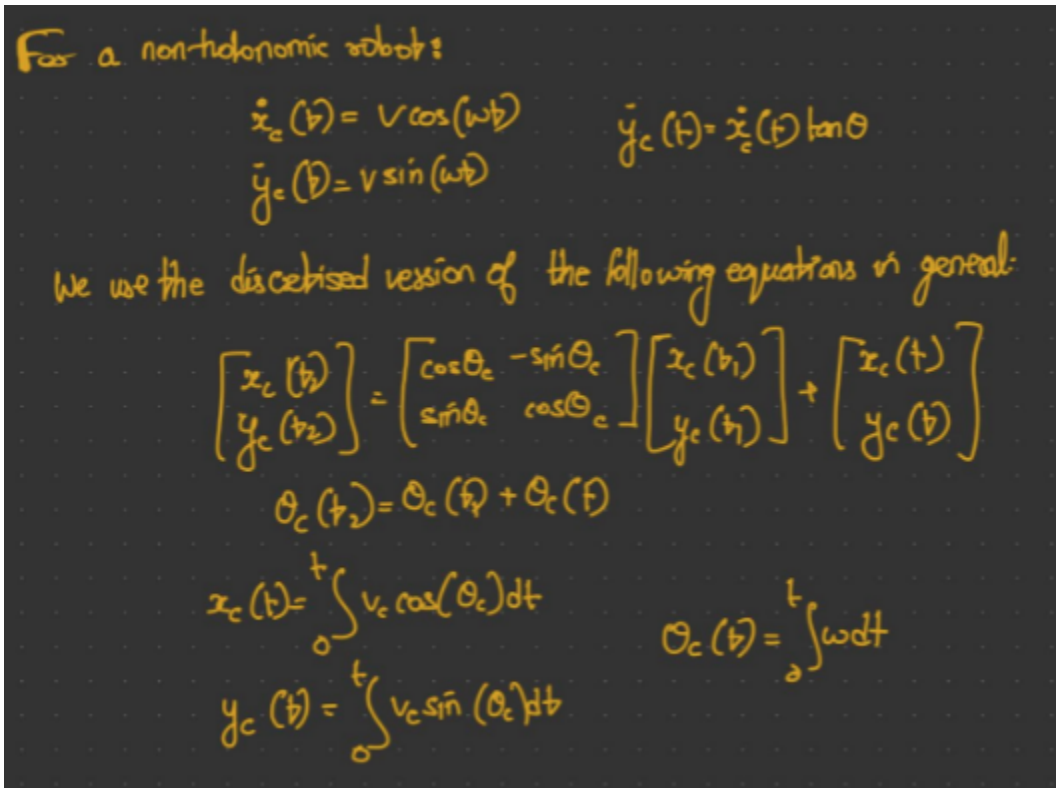
- Then from the already generated part of the tree, connect the tree towards the random configuration which is nearest to it and store each connection or we extend the already existing tree if they are in the same direction of the closest node.
- We ignore the randomly generated configuration if the edge generated or the node generated lies in a obstacle
- Set a threshold for the finish position within which if a node reaches, then it is said to have reached the goal.

Summary of generating the path:

- As we store each node's parent link, we start to traverse back from the last generated node till we reach the initial position in an iterative manner.

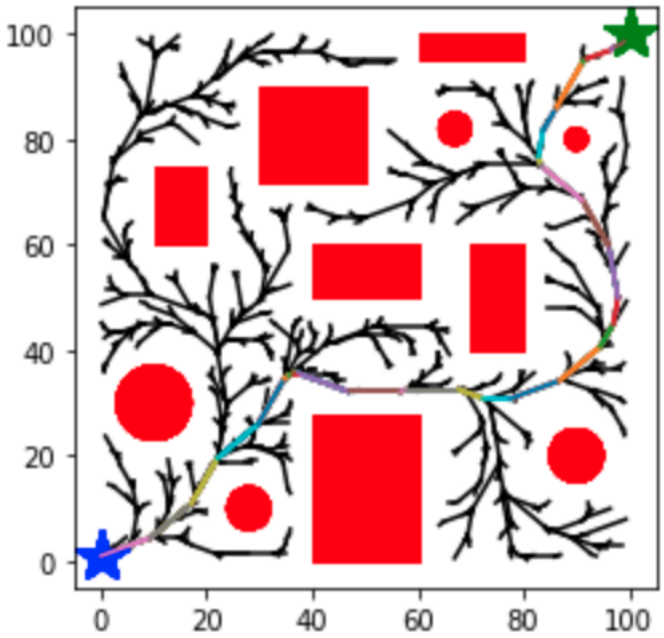
Relations:



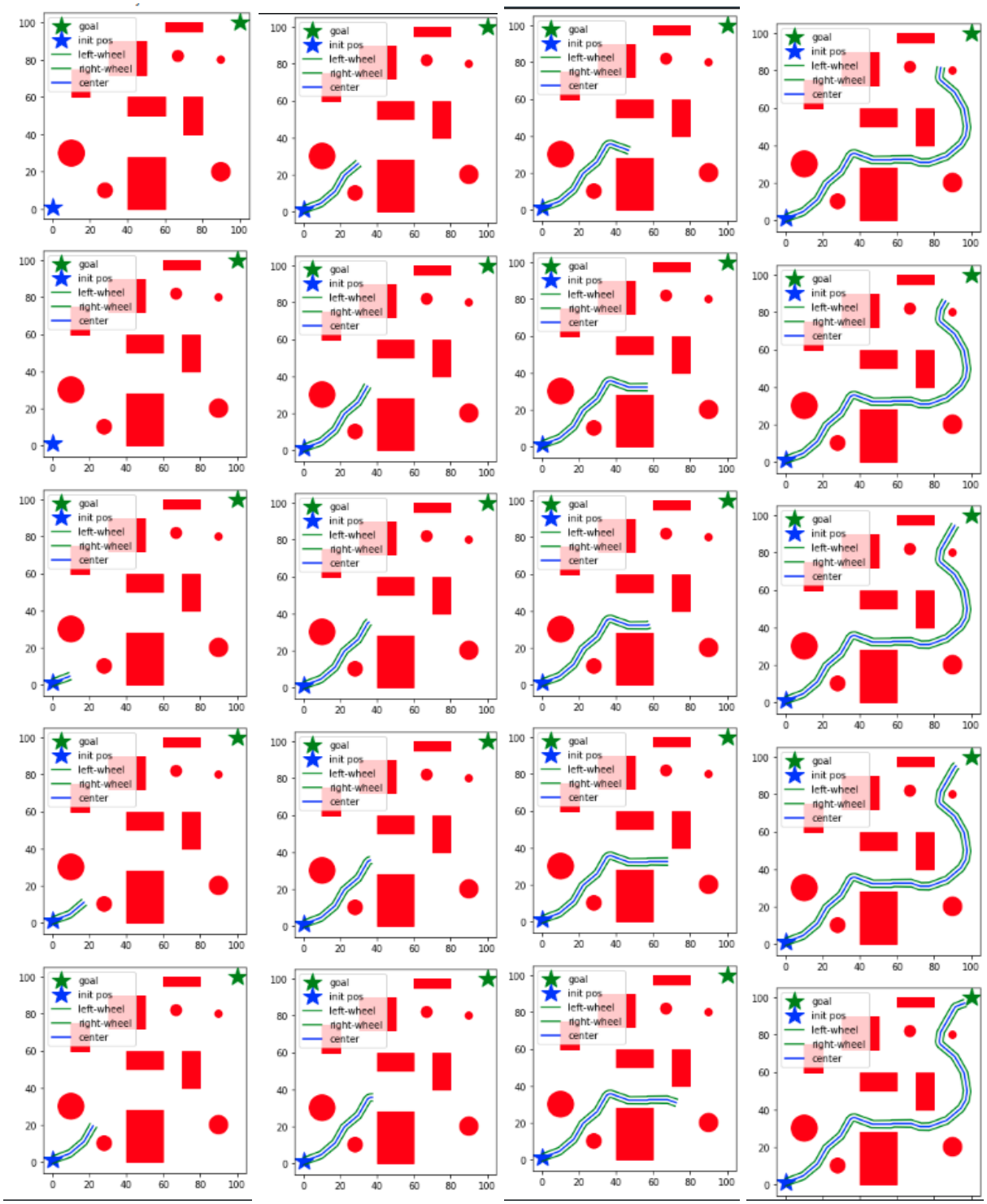


Results:

Output of trajectory with the trees generated:

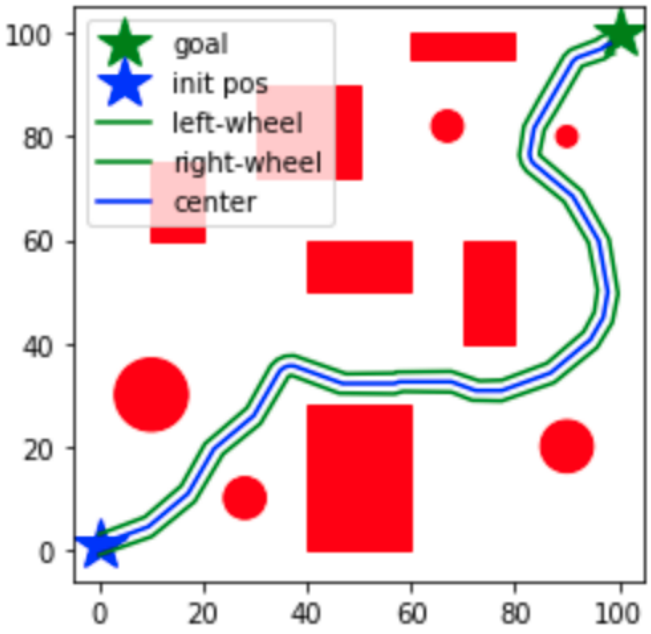


Snippets of path generation step by step:



NOTE: Above shown image is not the entire set of iterations but rather a sample

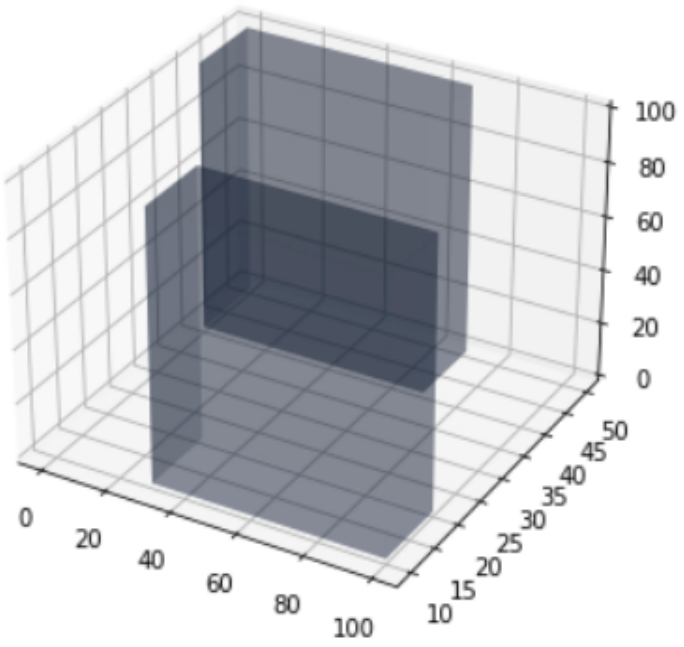
Output trajectory of the wheels following the path generated:



We see that the obstacles have been avoided while traveling from the start to end position even when the wheel distances are considered thereby validating the algorithm's working..

Question-2:

Custom Environment Used:



The fundamental idea of the algorithm remains the same. The difference is in the implementation of drone dynamics with its holonomic constraints.

- Get a random goal in greedy fashion, then find the closest node to goal and then get velocity at that node.
- Check the range of input values and coordinates that each possible input takes the drone to and then iterate through the phi and theta values to see where they end up.
- We select the next node also in a greedy fashion with min distance to goal.
- Then we update the acceleration with curren chosen input
- Then with the acceleration, we update the current velocity, position and add the node to the tree.
- After expanding the tree we now move on to getting to the actual goal.

Relations:

1) **Acceleration:**

a) Ax = -cos(phi)*sin(theta)*T/m

b) Ay = sin(phi)*T/m

c) Az = -g + cos(phi)*cos(theta)*T/m

2) **Velocity Update:**

a) Vx(t) = Vx(t-1) + Ax*dt

b) Vy(t) = Vy(t-1) + Ay*dt

c) Vz(t) = Vx(t-1) + Az*dt

3) **Acceleration Control:**

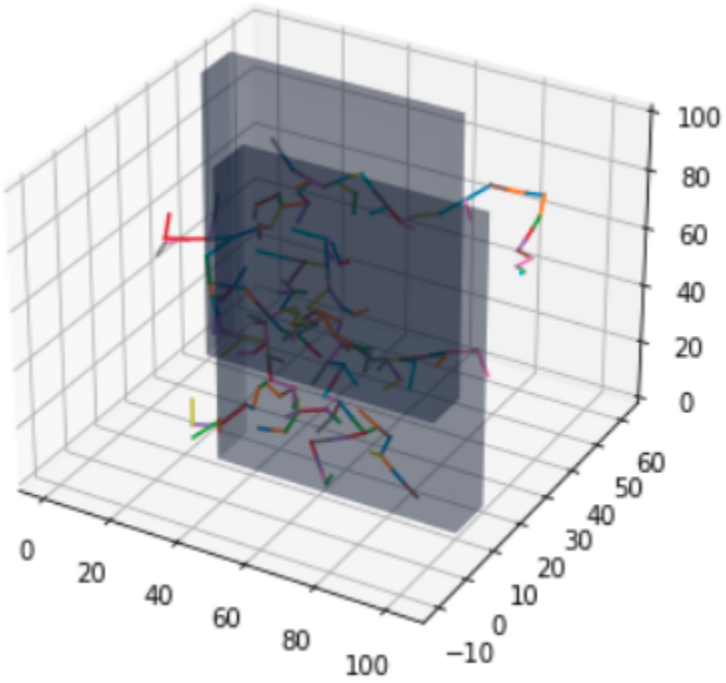
a) X(t) = X(t-1) + Vx*dt

b) Y(t) = Y(t-1) + Vy*dt

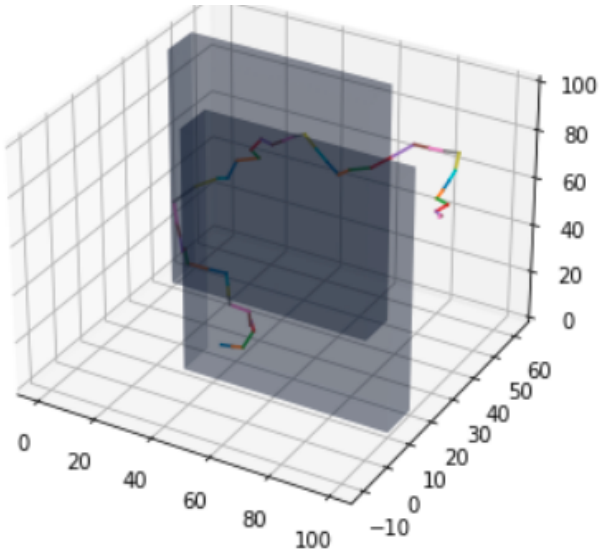
c) Z(t) = Z(t-1) + Vz*dt

Results:

Output of generated tree:



Output of trajectory generated:



We see that the obstacles have been avoided while traveling from the start to end position.

Graphs of theta, phi and Thrust vs the iterations: