Robotics Project

Shravan Singh

June 2022

1 Mapping

I used the gmapping slam module in ROS to create the map. Or rather to create a sketch of the map. However I found that my robot kept on bumping into obstacles as it traversed its path. This led me to the conclusion that the obstacles did not account for the robots width. I then added padding around each obstacle. My first approach in this aspect was to use an online morphology tool to add Binary Dilation to the obstacles https://planetcalc.com/9325/ Morphology Tool. This worked however it did not add enough padding. My next approach was to manually edit the map with Microsoft Paint. This worked extremely well as I was able to add the sufficient padding to the obstacles so that it accounted for the robots width. I also used jupyter notebook to convert the edited map file into a grey scale image which could be used in my algorithm. I used the PIL library. The map generated was a 4000x4000 image. I used the default parameters except for the recommended change of the minimum score argument which was set to 100000.

2 Path Planning

In this project I used a Probabilistic Road Map to plan the path. This first consisted of sampling a number of points around the map. I restricted that to be in the range of [-20:20] on both the x and y axis in the real world. I had to sample these points onto the actual map file which was of dimensions 4000x4000. After that a graph was created of these points. Adding the edges to the points in the graph was one of the major problems. This is because we cannot add an edge between 2 points if an obstacle exists between them. This was the next problem. To check if there was an obstacle between 2 points. To solve this problem, I went back to grade 9 and realised that with 2 points we can form a straight line mathematical equation. This equation was key in solving the problem of obstacle collisions. It was quite fascinating as this straight line went through indexes of a 2D matrix that represented the map. For every x, we get some predicted y value. That pair of coordinates [x,y] was then checked in the map matrix representation to see if there was an obstacle there. The only down fall to this solution was that the robot had to travel alongside straight lines which was not hard to implement. Another problem that had to be solved were steep gradients. If you would imagine the coordinates [2000,2000] and [2001,2080]. X axis moves along by one step yet the y axis moves 80 steps. To solve this problem I would simply iterate through the y indexes whenever the step size was greater than 1

3 Motion Planning

For the motion planning. I used a simple PID Controller. However with a *twist*. As listed in the section above the robot can only move in straight lines. So what I did was to always rotate the robot to the correct angle first, and then move to the new coordinate. The error metric was Pythagorean. The distance error was calculated as the square root of the x and y values squared.