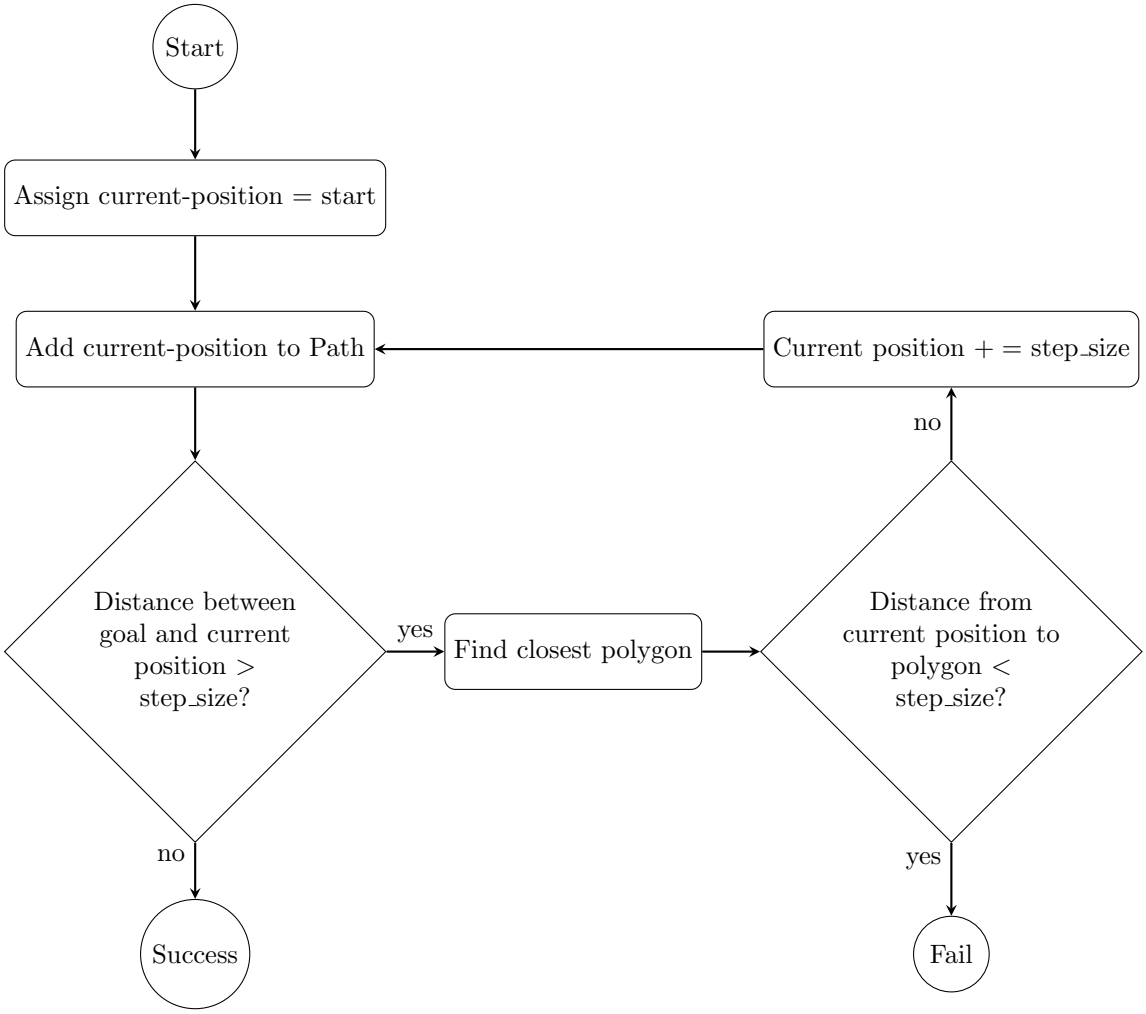
Bug 1 algorithm - Project #1

ME 179P - Robotics: planning

Kenny Hoang Nguyen
X299187

February 2020

1 Flowchart of BUGBASE



2 Improvement to implement bug1

To implement the bug 1 algorithm using bugbase as a base, I would first modify bugbase so that it takes in a path it can just append and expand.I want the code to run in two modes; "normal" where it has not hit an obstacle, which will be run with the bugbase code, and "collision" where it will run a special code when the robot hits an obstacle for circumnavigating around obstacle. The circumnavigating part

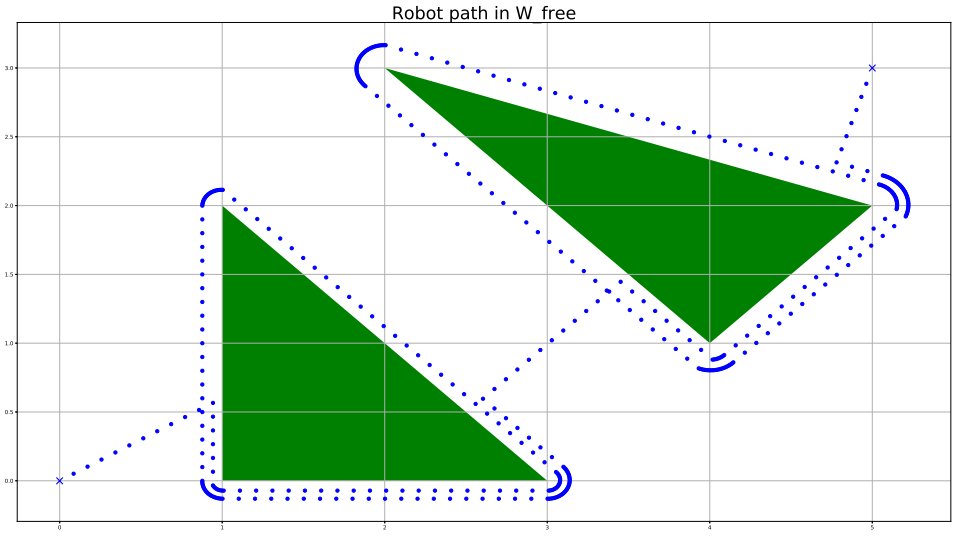


Figure 1: The path robot has taken through the given workspace

of the code will just make the robot follow the edge of the obstacle it has collided. Here we use the code from homework 2 to find a tangent. Then as it is circumnavigating the obstacle we want to find the point it is closest to the goal. When the code has detected it has run the obstacle once, it will go to the point it is closest to the goal and leave from there. To calculate this distance we use the codes we made in homework 1.

3 Bugs in my code

I believe that the project description was somehow vague, so I had to make some assumptions. I made the assumption that each obstacle has a distance between each other that is atleast twice the step size. In addition I had to make the assumption that the polygons/obstacles are convex. With these assumptions the only time the robot won't have a path to the goal is if the goal point is in a obstacle, meaning it is unreachable. The code may work for non-convex objects as well but with this implementation the robot may end up failing, even if there exists a path. So the code

4 Result from code execution

The step size is fairly large so we will never go all the way to the edge of the obstacle, however we see that the code works for the free workspace in 1.

In figure 2 we see that we have the monoticity property meaning the robot will indeed get to the goal if there exist such a path. I have assummed that it takes $0.1s$ for the robot to take a step, so when the robot goes around the corner of obstacles it takes shorter steps at the same amount of time.

In this environment the code takes approximately $0.07s$ to execute and the total path length the robot has to take are 25.72 units.

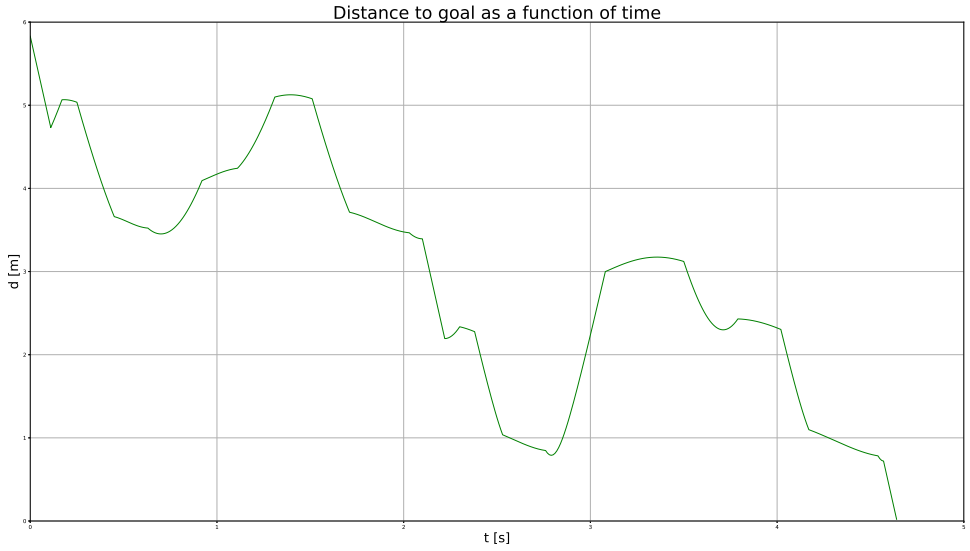


Figure 2: The distance to goal as a function of time