Control of Mobile Robots
CDA 4621

Lab Report 3
Motion Planning

By: Brandt Ousley
(U19292075)

Introduction

This document is a report for Lab 3: Motion Planning, CDA 4621, that consists of C.2 Task 1 – Motion to Goal and C.2 Task 2 – Bug Zero Algorithm. It goes over in detail, the objective of this lab, the methodology of each task, and a conclusion.

Objective

The objective and purpose of this lab was to learn how to implement motion planning for a robot so it could reach a goal whilst avoiding any obstacles that comes in between it and the goal.

Task 1 – Motion to Goal

The figure below was provided showing the camera color recognition and the maze that it will run on.

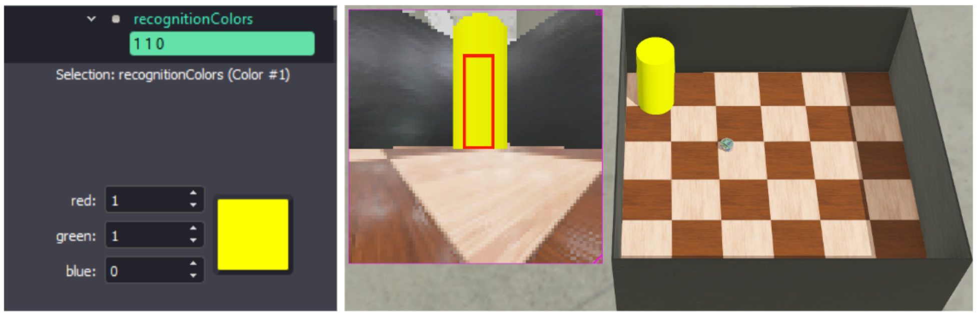

Figure 1. Camera color object recognition in Webots for motion to goal.

In this task, the robot must reach the yellow cylinder pictured above and stop within 0.5m from it. As you can in figure 1, there no obstacles between the two, the robot and the yellow cylinder (goal). It is asked that the robot should be tested from different starting locations and orientations. As the goal may not be visible at certain times when headed a different direction, the robot will have to turn until the goal is in view and then move towards it.

This is required for full and correct completion of this task:

- Robot reaches the cylinder from any starting position and orientation
- Robot finds the cylinder when the cylinder is not seen in the camera
- Robot stops 0.5 meters or less from the cylinder

Task 2 – Bug Zero Algorithm

The figure below was provided to show the different object configurations, with the robot starting at 'S' and ending the goal 'G' (which is a yellow cylinder in the recordings)

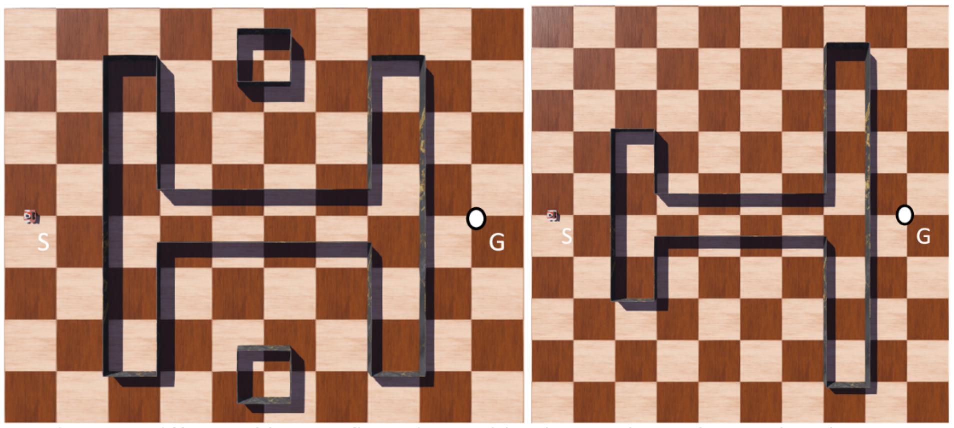

Figure 2. Different object configurations, with robot starting and 'S' and goal at 'G'.

This task asks us to implement a bug zero algorithm so the robot can traverse through the mazes in figure 1. This is different from task 1, as it includes obstacles, so the logic is a bit different, and the robot has to use wall follow and bug zero to complete the task correctly and successfully. We are allowed to use the camera to recognize the camera and its orientation and distance sensors to compute the distance. The task should be finished in less than 3 minutes. It is asked that our algorithm should perform correctly for both, left or right turns.

This is required for full and correct completion of this task:

- The robot must stop within 0.5 meters of the cylinder.
- The robot must navigate around walls without hitting them.
- The robot must switch between motion to goal and wall following.
- The robot must switch between wall following and motion to goal.
- Robot must perform clean and smooth left or right turns consistently.

Methodology

Task 1 – Motion to Goal

This task was completed using a PID controller, it uses rotationPIDBug() to calculate the absolute value of the PID error term, scaled by a constant factor 'K', for controlling the rotation of the robot towards the yellow cylinder (goal). The motionToGoal() function iterates over detected objects and adjusts the robots motion based off the position of the cylinder in its camera view. If the cylinder is off center or at any position the robot is not facing it or the middle, the robot adjusts it velocities to rotate towards it. Once it is aligned with the center of the cylinder (320 pixels – 640/2) it will move forward to it and stop within 0.5m or less of the goal. The camera was accessed like the example camera class PDF we were provided with. To help complete the task and recognize the center, obj.getPositionOnImage was used. Lastly, the front lidar was used for distance detection to know when to stop within 0.5m and the task runs for 30 seconds showing it can reach the goal from any orientation and position.

Task 2 – Bug Zero Algorithm

To complete this task, I had to implement a bug zero algorithm, to have the robot traverse both of the mazes to the goal correctly. Since there were obstacles (walls) in the way of the goal for each task, the robot switches between wall following and bug zero algorithm throughout.

The bug zero algorithm:

- Iterates through a list of objects representing potential targets.
- Calculates the position of each object on the image captured by the robot's camera.
- Depending on the position of the object, it adjusts the robot's movement to either rotate towards or away from the object.
- If the object is within a certain threshold, to left or right, it rotates the robot accordingly using PID.
- If the object is within a certain distance ahead of the robot, it moves forward towards it, adjusting the speed based on the distance.

The controller code can take a parameter, left or right, to specify which wall to follow (turn left or right). The robot will switch between the two algorithms, when the goal is detected it will move towards it but when an obstacle is in its way it will navigate around it using wall follow, then switch back to bug zero when the obstacle is in sight. If no objects or obstacles are detected it will continue with wall following, adjusting its speed to maintain a certain distance from the specified wall. The task is completed within 3 minutes for each maze and each wall follow. It uses logic we learned from the prior two labs.

Conclusion

To start off, this was a very interesting and intriguing lab, yet I found it to be very difficult. However, it was very rewarding, and I learned a lot from this. One reason why I feel that this lab was very time consuming/difficult for me was due to the fact I had to implement wall follow code from last lab into task 2, but my code was not originally great at all and performed very poorly. This led to me having to completely restart my wall follow code and do it based off the provided wall follow code PDF. To me, for both tasks, the robot runs great and meets all the standards, however, it could use a good fine tuning adjusting certain values to make it run more smoothly. One big thing I took away from this lab was how much easier and simpler it is to code in the MyRobot.py file and then call what is needed in the controller code. As I use to code directly in the controller on WeBots, the text editor was not great. This led to me reimplementing things we learned from lab and lab 2, which helped tremendously as it not only easier and smoother to code, but my calculations are more fine tuned/on point allowing for the robots behavior to be better. This should make it easier for the next labs as a lot of the basic functions should always have to be used for the robot's behavior. In the end, this lab was definitely worth all the time I put into it, as it gave me a deeper understanding of how

PID and wall follow works, how to integrate the camera to do certain things and how I can tie it all together into using bug zero at the same time to complete the required task(s).