EE4308 Part 1 - Project Progress Report Franky Laurentis (A0144164L)
Turtlebot Autonomous Navigation Tan RongDe Darius (A0146654Y)

1. Introduction

Our basic idea is to divide the map into a 9 x 9 grid coordinate system and to use the A* algorithm for path planning towards the goal coordinate. The obstacle avoidance is done by checking the presence of walls using the depth data from the RGB-D kinect sensor and updating the A* algorithm grid with the presence of walls to modify the path planning. The motion of the Turtlebot is executed using odometry to check if the bot has reached its intended coordinate in the grid.

This basic system can then be refined by smoothening the motion to allow for smooth turns while moving or by adding edge detection in the depth sensor to allow for preemptive wall detection and to keep the Turtlebot between the 2 sides of the wall in the middle of the path.

We are also planning to add a directional semaphore function to simplify the control system for the robot to track its previous position and orientation in a more organised manner.

2. Completed work

Using C++, we have completed a depth information node to publish camera depth information, a position information node to subscribe to odometry data and publish the X, Y pose and yaw orientation and a bot controller node to subscribe to both position and depth information, and to publish geometry messages to the `mobile_base/commands/velocity` topic to move the Turtlebot.

The bot controller C++ file also includes an algorithm header file to implement the path planning algorithm, in the current case the A* algorithm, and return the next coordinate in the planned path towards the goal to the bot controller node. This header file also contains the code for the grid system, which is updated by the depth information node when the Turtlebot faces a wall.

3. Current status

We have completed basic testing of the motion in the grid coordinate system from coordinate to coordinate and have implemented the A* search algorithm. We have tested the code for world 1 and the Turtlebot successfully managed to navigate to the goal autonomously using our code by moving from coordinate to coordinate while checking for the presence of a wall in front of the bot and updating the A* algorithm. In our tests, the Turtlebot takes approximately 72s to reach the goal with our current configuration. However, there have also been exceptions when running the code that the Turtlebot gets lost and is unable to determine its current coordinate which could be due to our tolerances or due to the Turtlebot being unable to determine its current coordinate accurately after detecting a wall.

Upon testing in world 2, we also found that the Turtlebot seems to get lost at certain moments which could possibly be due to the inaccuracies in the motion in terms of deviations from the expected path

as the path to goal for world 2 is much longer than that in world 1, which causes a buildup of errors over more distance moved. Upon testing with higher tolerances, the motion allows for higher deviations but this consequently affects the accuracy of the movement, which causes the Turtlebot to move more / less than expected to get from a coordinate to an adjacent coordinate which causes the Turtlebot to hit the walls at certain points.

We have also been actively working on a private GitHub repository, which will be made public upon project completion, as our version control and collaborative code sharing platform.

4. Roadblocks

Our current roadblocks include increasing the accuracy of the motion of the Turtlebot to prevent large deviations from the intended motion while having appropriate tolerances to accommodate for deviations in the motion.

If we decide on improving the motion to allow for smooth turns or to allow for preemptive wall detection, the code allowing for those functionalities will need to be developed as well.

5. Approximate time schedule

Academic Week	Day / Date	Action
6	Thurs, 21 Feb	Write progress report
	Fri, 22 Feb	Submit progress report
Recess	Mon, 25 Feb	Fine tune motion control
	Wed, 27 Feb	Add turn smoothing functionality
	Fri, 1 Mar	Touch up and complete all code
7	Mon, 4 Mar	Write final report, prepare for project demo
	Fri, 8 Mar	Project demo (tentative)
		Submit final report and code