Autonomous Navigation of Turtlebot in an Indoor Environment

Important Note: This project has two components. The first part of the project will be due on 13/02/2017 and will consist of a 3-page report along with your source code, to be submitted on IVLE. Part 2 will include a final report, demonstration and presentation. The codes and Final Report are to be submitted by 27/02/2017, and the presentations will be held in lab the week following recess week.

This is a group project, and you can have a maximum of two persons per team. You are free to choose any of your classmates as your teammate. Nevertheless, we strongly encourage that all the members of the group contribute equally towards the project.

Grading of Course Project

- Part 1 (3 page report and code) : 10% of the total grade for the course
- Part 2 Demonstration and Presentation : 20% of the total grade
- Final Project Report : 10% of total grade

1. Background

Robots are becoming more and more commonly used in daily life, especially in situations that pose a threat to humans, such as mining accidents, explosions, urban disasters etc. The use of a robot could greatly reduce the risk of harm to emergency personnel including firefighters, police officers etc.

For your project, we are going to model the autonomous navigation of the Turtlebot robot while it is performing a search and rescue operation. For the first part of the project, we can assume that the robot knows the blueprint of the indoor environment and it merely needs to navigate to a given position within the room. This can be achieved by a simple path-planning algorithm (such as a wall follower), or even by hardcoding the distances that the robot needs to travel.

In Part 2, the robot is unaware of the room layout, and is only given the location of where it needs to go within the room. For accomplishing this task, a more robust path-planning method will need to be implemented.

2. Required Software

To complete this project, you will need to be familiar with the Robot Operating System (ROS) software. Installation instructions and other helpful tutorials can be found on *wiki.ros.org*. Additionally, you should be writing your programs in C++.

The version of ROS used for this class is ROS Indigo and it should be installed on Ubuntu 14.04 Trusty Tahr. You can also access ROS by using the computers in E4A – Mechatronics and Automation lab.

3. Project Part 1

As specified above, for the first part of the project, you will be given a .world file consisting of the following indoor environment

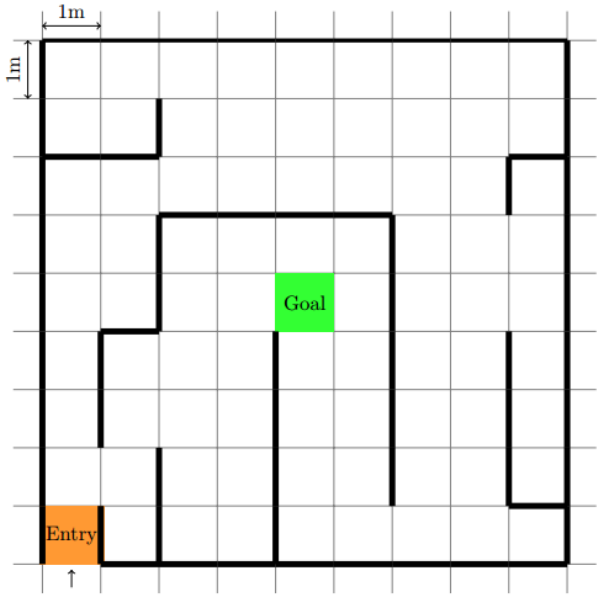

Figure 1: Indoor environment for part 1

You will start out at the position given in orange, and will need to navigate to the goal position given by green. Once the final position is reached, the robot can merely remain there, and does not need to return to its original position. Any path planning method of your choice can be used for solving this problem. It is suggested that you also implement a feedback controller so as to counter the drift of the robot.

You will need to submit a small report (Maximum 3 pages), describing how you solved the problem given in Part 1, especially if you used any specific path planning algorithms. You need not write introduction, conclusion etc. for this report. Only a short analysis of what you have done will suffice. Your codes, report and read-me file (describing exactly how to run your program) is to be uploaded to IVLE by 13/02/2017 in a .zip file of the form *"MatrNoofMember1_MatrNoofMember2_Part1.zip"*. Please make sure to include any other files that may be needed to run your program.

Supporting Files (Uploaded on IVLE)

- Codes (in the folder *CPP_files.zip*):
  1. turtlebot_move.cpp – To move the Turtlebot a set distance in 'x'
  2. turtlebot_turn_cw.cpp – To turn the Turtlebot in clockwise direction
  3. turtlebot_turn_ccw.cpp – To turn the Turtlebot in counter-clockwise direction

4. turtlebot_ee4308_obs_avoidance.cpp – This will give you the depth at the center of the image, as well as the position of the left most edge and right most edge. You can directly use the depth information at the center to complete your task, but including the edge data in your obstacle detection algorithm will be the more optimal approach.
5. CmakeLists.txt – Accompanying the "turtlebot_ee4308_obs_avoidance.cpp" file

Note: These codes are only intended as a reference. You can write your own codes for accomplishing the task if needed.

- Launch Files (in the folder launch_files.zip)
  1. ee4308_turtle_part1.launch – Copy to */opt/ros/indigo/share/turtlebot_gazebo/launch/*
  2. kobuki.launch.xml – Replace the original file in */opt/ros/indigo/share/turtlebot_gazebo/launch/includes/*
  3. ee4308_proj_part1.world – Copy to */opt/ros/indigo/share/turtlebot_gazebo/worlds*

Now you should be able to run your launch file to start the simulation in gazebo using:

roslaunch turtlebot_gazebo ee4308_turtle_part1.launch

Note: Sometimes the simulator can fail to launch initially. So, just try to run it once more. If not, check if all the files are in the correct location.

4. Project Part 2

For this part of the project, you will be working with an unknown environment, and so will need to modify your algorithms accordingly. The starting and ending positions, as well as the size of the grid, will be the **same** as given in figure 1, but how the walls are arranged may vary. **Another** .*world* and *launch* file will be released on IVLE before 13/02/2017 so that you can test your different path-planning methods. You may use this environment to explain your path-planning/obstacle avoidance algorithms during your presentation and demonstration.

Nevertheless, keep in mind that this may not be the same environment that your code will be finally tested on by us. You will be given points based on the speed at which you complete the task, as well as the robustness of your code for different scenarios. Note that there is no singular path-planning method that performs well for all cases. So try out different ones during your project and choose whichever one fits best.

As a part of your project, you will need to do a Presentation on Week 7 (Week after recess week). During the presentation, the members should clearly explain the path planning method that they chose and why it is optimal for the problem given.

Your code (clearly commented, along with a readme file describing how to run it) and the final report should be submitted to IVLE in a zip file of the form *"MatrNoofMember1_MatrNoofMember2_Part2.zip"* by 27/02/2017.

Format of the Final Report

The Report should be organized in the following sequence

- A Cover paper which specifies the project title, names/matriculation numbers of all the group members, and date
- An Abstract of 50-100 words on a separate page
- A Contents table on a separate page
- Introduction
- Body of the Report – Should include details on the path planning methods that you tried and which one you chose. Any figures, equations or tables you include should be clearly labeled.
- Conclusion
- References and Appendices

Do not put unnecessary information in your report and try to restrict to 10 pages (excluding cover page and appendices) with a 12 pt. font.