UNSW, School of Computer Science and Engineering

COMP3431: Assignment 1

Due: September 11, 2015

1 Overview

The purpose of this assignment is to provide students with familiarity with robots. In particular, this assignment will give students experience with processing sensor input, localisation, SLAM, and navigation.
Students are to perform this assignment in groups of four. You are requested to work closely together on the project, developing the code together. If you use the "divide and conquer" technique of splitting the project into parts, developing the parts separately, and then recombining please make sure you each have a broad understanding of how the other parts work.

1.1 Deliverables

This assignment requires you to develop behaviours for a turtlebot robot.
During the lab each team will demonstrate their code running on a robot. The solutions will be compared and graded in the lab. You will also submit both your source code, and a report in an easily readable format (e.g. plain text, html, PDF) describing your approach. These documents should be emailed to mmcgill@cse.unsw.edu.au.

1.2 Due Date

Each group will have 15 minutes during the labs on Tuesday, September 8 2015 or Thursday, September 10 2015, in which to demonstrate their behaviours. The groups will also email their ROS packages including source code and launch files as well as a written report to Matthew McGill by 11:59pm on Friday, September 11 2015.

2 ROS Modules/Nodes

This assignment will call for at least 3/4 ROS modules. They are probably best developed as separate nodes/nodelets but can be combined if you choose.

1. A exploration module for initially exploring the maze.
2. A vision node to detect, identify and locate the coloured beacons.
3. A waypoint navigator for traversing the waypoints in the prescribed order.
4. A planner to keep track of the beacons as they're located and decide between visiting a specific node and continuing the exploration.

The marks for these modules will be based on how the perform in the demo and how well they are documented in the report.

2.1 Exploration (10 Marks)

At the start of a run the robot will be placed in an maze which not have been seen before. You will also be given a list of waypoints/beacons that will need to be traversed in order. The first task will be to explore the maze searching for the required waypoints.
Any exploration strategy can be used for mapping the maze and locating the waypoints. The maze will be designed such that a simple wall following algorithm can successfully traverse the emtire maze.

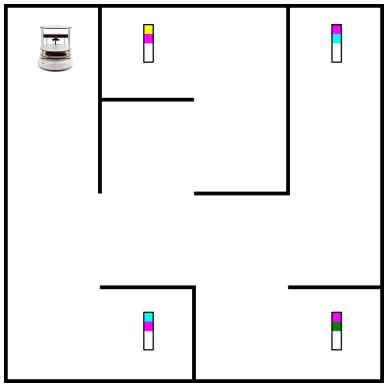

Figure 2.1: Example maze showing beacons and turtlebot

2.2 Beacon Recognition and localisation (5 Marks)

You will need to process the data from the robot's colour camera to identify beacons as the robot encounters them. The beacons will be placed on blocks so that they will be guaranteed to be visible to the lasers located on the top of the robots. This allows the

relative location of a beacon to be inferred when it is detected. Alternatively you can use the depth image or pointcloud data to find the relative position of a beacon.
Be warned, given the layout of the maze and placement of the beacons it is possible for more than one beacon to be visible in the camera image simultaneously.

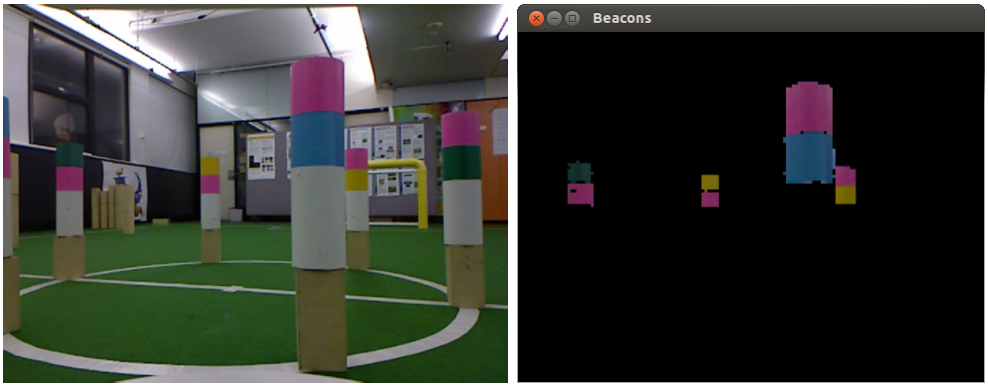

Figure 2.2: Beacons recognised by colour

2.3 Waypoint Traversal (10 Marks)

Given the design of the maze it may be possible that a wall following algorithm will eventually visit all of the beacons in the required order. This will however take much more time than is required and could well take longer than the 15 minutes available. A more intelligent choice is once the position of the beacons is known to visit them in order. A simple A* search over the current map can be used to plot a path from the robot's current position to a desired waypoint.

2.4 Planner (5 Marks)

The planner is responsible for keeping track of the beacons as they're located and deciding when to switch between exploring and visiting a specific node.

3 Report (5 Marks)

Much of the work in robotics requires being able to explain and justify your decisions to others with whom you work. The purpose of the report is for you to explain the algorithms and behaviours you have used. You should explain not just what you have implemented but also why you have chosen to build it the way you have.

4 Available Packages

4.1 Banned packages

For this assignment you may NOT use either the `move_base` or `crosbot_explore` packages.

4.2 OpenCV

All of the lab machines have OpenCV installed. It is also available in the standard ROS repositories and you are free to use it or not as you choose.

4.3 Comp3431

To aid you with this assignment some nodes and source code will be available in a git repository. To access this code run this command:

```
git clone git://robolab.ai.cse.unsw.edu.au/ros/2015/comp3431
```

Inside the Comp3431 repository will be the assign1 package.

4.4 Mapping

You are free to use which ever mapping solution you choose. You will have access to the standard ROS mapping agorithms GMapping and Hector SLAM. Our inhouse implementations of Fast SLAM and Graph SLAM will also be available.

5 ROS topics

There are a very large number of topics streaming while the robot is in operation. Some that you should pay attention to include:

`/camera/rgb/image_color` (`sensor_msgs/Image`) The colour image from the camera.

`/scan` (`sensor_msgs/LaserScan`) The scan data from the laser. If the robot is without a laser this data is generated from the Kinect's depth image.

`/tf` (`t2_msgs/TFMessage`) All of the tranforms for the robot get published to this topic. For simplicity you will probably want to use a tf listener rather than subscribing to this topic yourself.

`/cmd_vel_mux/navi` (`geometry_msgs/Twist`) The topic you should publish to, in order to get the robot move. Only the linear.x and angular.z components can be used by the turtlebots' drive system.

`/map` (`nav_msgs/OccupancyGrid`) This is the latest version of the map. It will be published every few seconds as the robot moves.