RBE 3002 Unified Robotics IV: Navigation

B-Term 2017

Lab Assignment #3: Mobile Robot Path Planning

Introduction

In this lab, you will implement the A* path planning algorithm and demonstrate its use using RViz. This lab is to be done in groups of 3.

Objectives

Upon successful completion of this lab, you will be able to:

1. Load a map of the environment.
2. Plan a path for a mobile robot to navigate through this environment.
3. Generate a set of waypoints based on the path.
4. Use RViz to display the map and the planned path.

Pre-Lab

1. **Review class notes** on the A* algorithm and develop a plan to implement it.
2. **Identify key topics and subscribe.** Begin by writing a program which subscribes to the relevant mapping topics published by the TurtleBot GMapping example you ran during Lab 1. Create callbacks for each topic and print out some of the information contained in the messages. Become familiar with the structure of the data returned. Detailed explanations of each message type can be found here: http://wiki.ros.org/common_msgs

   To determine which what type of data is being published on a topic, try out the following commands:

```
rostopic list
rostopic echo /topic
rostopic info /topic
rostopic type /topic
```

Lab Work

1. **Develop a visualization.** RViz is a wonderful utility to visualize what the robot is seeing/thinking. This will greatly simplify testing your code, and looks much cooler than just text in a terminal (and much easier to understand too!). An example setup can be seen by running the following:

   To run the TurtleBot example RViz configuration:

```
roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Flip through the list of "Displays" which the TurtleBot Navigation RViz windows shows. Identify the types of these displays, and make a list of topics they use, which you will plan to reuse. You can see a list of all available display types by clicking the Add button at the bottom of the Displays panel. *Note: many of the displays in this list have customized names. "Map" and Costmap are actually the same display type, with different names and "color schemes."* Try experimenting with these displays by changing some of their settings, such as the color scheme of the different maps.

For this step of the lab, you will **create your own RViz layout**. Start by running the following code to open a blank RViz session. Add a number of Displays to RViz which will allow you to visualize your path planning algorithm. Descriptions of each Display type are available here: http://wiki.ros.org/rviz/DisplayTypes

To open a new, blank RViz window:

```
rosrun rviz rviz
```

You will need the following types of data in your new RViz session:

- Model of the Turtlebot (Robot Model)
- A map of the environment (Map)
- A way to visualize your path planning algorithm
    - Consider showing expanded nodes and frontier for debugging
- A start position and goal position

On the top bar in RViz, you will notice two green arrow buttons, "2D Pose Estimate and 2D Nav Goal. These can be very useful as inputs to your programs, as they publish to the topics "/initialpose" and " /move_base_simple/goal" respectively. Subscribe to them in your program, and then you can interactively send start and end points into your program, rather than hard-coding them.

Once you have configured RViz to your liking, you can save the file using the "File > save Config" menu option. Use "Save as" to save it to another folder for future reference.

2. **Load the map.** Write a package that reads in the provided map and visualizes it in rviz.

We recommend storing all your map files in a single directory, such as `\home\<yourusername>\maps`. Download lab3map.yaml and lab3map.png from the Resources page and save them in your designated map directory. Then edit the first line of the yaml file to reference this directory in the path to the image file.

This map describes an occupancy grid of the front room of the RBE Lab. To publish a static map, download the sample maps from class site and run the following command (be sure to change the path as appropriate):

```
If you would like to run in simulation with a different
map:
export TURTLEBOT_STAGE_WORLD_FILE=~/location of map.world
export TURTLEBOT_STAGE_MAP_FILE=~/location of map.yaml
before running the turtlebot stage launch file.

Or if you wish to run the full navigation stack on the
robot with a premade map:
export TURTLEBOT_MAP_FILE=/tmp/my_map.yaml
and then run
roslaunch turtlebot_navigation amcl_demo.launch
in place of gmappingdemo.launch

You can also use the map server to simply publish a map
without a robot or simulator.
rosrun map_server map_server /path/to/map.yaml
```

(note: you will have to change the topic name for "2d nav goal" or else the built in navigation stack will respond. To do this simply right click on the "2d nav goal" button and select properties)

Implement a function to overlay some random tiles over some occupancy grid cells within the map. Do this by publishing a GridCells message on a new topic. Load this topic into a new GridCells display in RViz. Documentation for the GridCells message type can be found here:

http://wiki.ros.org/rviz/DisplayTypes/GridCells

Be sure that each tile in the GridCells message lines up exactly with the expected tiles of the map. This functionality will be critical for the following step of the lab assignment.

3. **Create your A* implementation.** Implement an A* search to generate paths between two points in your occupancy grid. Use rviz to set the start and goal points and to display the progress of the A* algorithm. While searching, the map should visually distinguish expanded, frontier and unexplored nodes (i.e. make them all different colors). Once the path is found, the map should display the final path only. Create unique topics for each color of grid cell you will be using, and name them appropriately. Be sure to load up matching GridCells displays in RViz, and save any changes you make to the RViz configuration

4. **Calculate waypoints.** Once the final path has been calculated, extract waypoints representing the intermediate states the robot would need to achieve to follow the path. Express the waypoints as a nav_msgs/Path.msg for implementation in future labs in conjunction with Lab 2

Extra Credit (+10 pts)

On the real robot, you will need to constantly rerun A* as the robot learns more about the world. Design your code to continuously run as a dedicated path-planning node. It should provide a service which takes in a start and end position, and returns a path (or an error if no path exists).

Extra Credit (+5 pts)

Have the turtlebot navigate, in the real world or in simulation based on the fake map given to it.

Deliverables and Deadlines

Please submit:

1. Copy of the sign-off sheet
2. Compressed folder containing your package (e.g. jdoe_lab2.zip).
3. A brief report describing
    a. Your implementation of A*, include a description of how you calculate the cost of the path and the heuristic.
    b. How you extract waypoints from the planned path.
    c. A screenshot of your rviz interface during planning.

    If you were unable to complete the entire lab, you should describe what parts work, which do not and what problems you encountered. The complete document should not exceed 2 typed pages.
4. If you used external sample code other than the packages listed in this lab document, please also submit a document called References.txt that lists all such sources, as well as a very brief description of what you used from each one.

RBE3002 Lab3: Mobile Robot Path Planning

Name: ____

Name: ____

Name: ____

Lab Sign-offs

TASK	**SIGNED BY**	**DATE/TIME**
Prelab		
RViz configuration		
Loading the map		
Coloring the map		
A*		
Waypoints		
Extra Credit		

Grading Rubric

[100 points] All pre-lab and in-lab procedures are completed and demonstrated prior to the deadline. All deliverables and sign-off sheet are submitted on time.   The code is well commented and structured.