Last modification: November 17, 2020

RBE3002: Unified Robotics IV
B-Term 2020/2021
Lab 4 (Final Project): SLAM

Introduction

Welcome to the final lab! It is time to combine everything that you have learned, and build on top of that to navigate and explore an unknown maze. The purpose of this lab is to explore and map an unknown area. During this lab, you will use the packages you have created during the previous lab and will build several new packages. In addition, you will use the `Gmapping` package to build the map.

Objectives

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

1. Localize the robot in the world
2. Drive around the map while avoiding obstacles
3. Find interesting areas to expand the map
4. Find optimal path to the interesting point
5. Identify when the map is complete
6. Navigate back to a point in the map

The Final Project

What Your Robot Will Have to Do

1. **PHASE 1:** Create a map of the environment using Gmapping
   (a) Calculate the C-Space of the map
   (b) Find a frontier to explore
   (c) Plan the optimal path to the frontier
   (d) Navigate to the frontier while avoiding obstacles
   (e) Repeat until map is completed
   (f) Save the map on file

2. **PHASE 2:** Using the map, navigate back from where you are to the starting position while avoiding obstacles.
3. **PHASE 3:** Navigate to a specific goal position in the maze. The goal position will be chosen by the evil professor.

How the Final Project Will Be Evaluated

The lab will be considered fully done if show your work with **noisy** robot that you used in Lab 2 for extra credit. In other words: the noisy robot was extra credit for Lab 2, but it's now **required** for full credit. You can find the code for the noisy robot at this link. You can also modify the 'world' argument to change the world your robot is spawned in. See this link to learn how to pass an argument using ROS Launch.

If your code works for the non-noisy robot (the original Turtlebot) but not the noisy one, we'll take 40% of the points off.

You *can* use the pre-made navigation stack (`amcl` + `gmapping` + `move_base`), but this will also incur in a 40% deduction on your final score.

PHASE 1 (45 points)

- The robot will start at one of the four corners of the maze and explore the entire maze to create and save the 2D occupancy grid in a map file.
- This should be achieved by autonomously detecting the frontier of an unexplored area such that entire maze is explored. The robot should stop after exploration is complete.
- A team can use a maximum of 3 runs to generate the map. Every run starts in the same initial location.
- If a team is unable to complete this task, they can use Rviz to generate the map manually with a 25% penalty. If the first 2 tries fail, it might be a good idea to use this.

PHASE 2 (10 points)

- The robots must be able to navigate back to the initial position.
- The initial position will be indicated using Rviz.

PHASE 3 (45 points)

- In the final navigation run, the robot starts from a specific corner of the maze and it must reach the evil location decided by the professor.
- We will start a timer when the robot completely moves out of the starting cell. The timer will stop when the robot reaches the goal location.
- If your robot completes the run within 5 minutes, you'll get full points.

- You'll repeat this run 3 times and we'll record the best time among those.

More About ROS

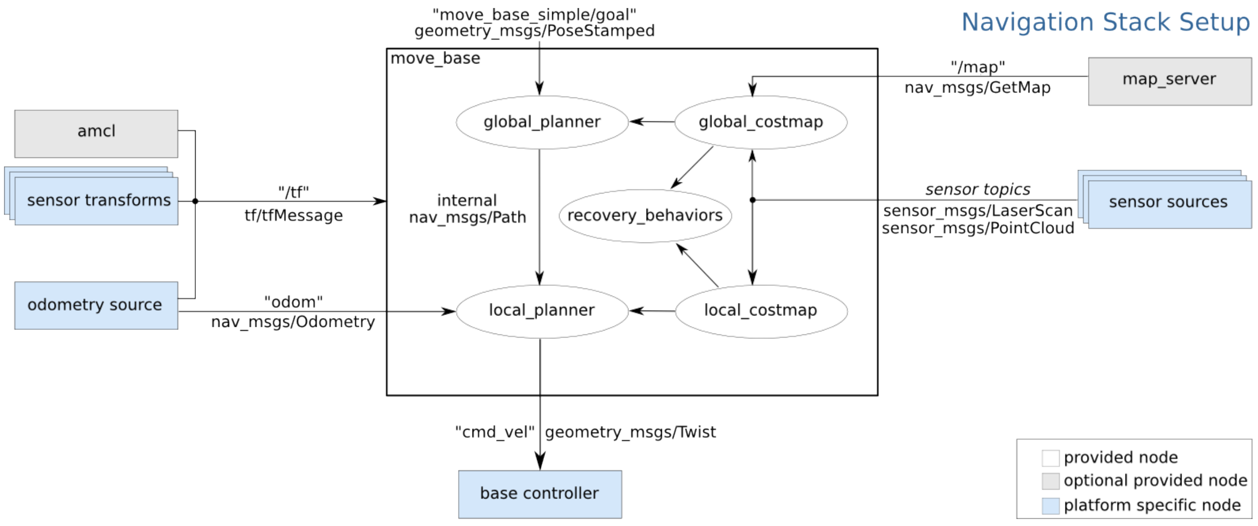
If you get stuck, keep in mind that the Turtlebot3 E-Manual exists! They have some pretty good resources on how to use the tools we will be talking about: Turtlebot 3 E-Manual

The ROS Navigation Stack

ROS provides a full **navigation stack** (remember: stack = collection of packages) that includes:

- Mapping — through the Gmapping package
- Localization — through the AMCL package
- Navigation — through the move_base

Have a look at the ROS Navigation Stack Tutorial for more information.



Gmapping

Gmapping is an algorithm for SLAM that takes laser readings and generates an occupancy grid. If you have not downloaded Gmapping, you can download it using the following command

```
$ sudo apt-get install ros-melodic-gmapping
```

To run Gmapping, you need to use the `slam_gmapping` node. To run this node, execute

```
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```

Gmapping publishes on the `/map` topic, so you won't need a map server in this lab because Gmapping will provide you with a map. Check `http://emanual.robotis.com/docs/en/platform/turtlebot3/slam/` for more information.

You must use Gmapping to complete this lab.

AMCL

AMCL is a collection of localization algorithms. If you have not downloaded AMCL, you can download it using the following command

```
$ sudo apt-get install ros-melodic-amcl
```

To run AMCL, use the command

```
$ rosrun amcl amcl scan:=scan
```

This command instructs AMCL to subscribe to the `scan` topic, which is the topic on which the TurtleBot3 publishes its laser scans. AMCL also subscribes to the topic `/map`, so you can send updated maps to the robot if needed. You will also need to send an initial map using a static map server, unless you pass an alternative parameter in the launch file. See the ROS Wiki about AMCL for more information.

If you want to run with Turtlebot configurations, use this command:

```
$ roslaunch turtlebot3_navigation amcl.launch
```

For Phase 3, teams should use a Static Map Server + AMCL. AMCL is required to help the robot localize within the map, and it needs a package feeding it a version of the map.

Move Base

Move Base is a package that contains several navigation algorithms. If you have not downloaded Move Base, you can download it using the following command

```
$ sudo apt-get install ros-melodic-move-base
```

You can use Move Base for testing, but your job really is to write your own navigation logic. You are encouraged to reuse all the work you did in Lab2 and Lab3, and restructure your code if you think it's a good idea.

About Frontiers

We discussed in class about frontiers. Refer to the class slides in how to pick a frontier and how to calculate its centroid.

To actually calculate a frontier, treat your occupancy grid as, well, a grid a apply to it an edge detection algorithm. Before applying the algorithm, you'll beed to threshold your map so you only consider the *walkable* boundaries between known and unknown.

Once you have a set of points that forms a frontier, you need to label the frontiers together. You can apply the same dilation algorithm you used to calculate the C-Space, followed by an erosion algorithm that turns the frontier back to skinny — but connected! To label the frontiers, you just pick a random point and navigate through the connected points, adding them all to the same "bin".

Keep creating new bins with unmarked points. When no points are left, use the bins to decide which frontier to visit next.

Saving the Map

Map files can be saved using the `map_saver` node from the operator computer. To run the `map_saver`:

```
$ rosrun map_server map_saver -f <map_name>
```

where `<map_name>` is a file name for your map.

Hint: It's a good idea to modify the heuristic of A* to penalize the number of turns in a path. For instance, when you detect a turn, you could add a little penalty (e.g., 0.01) for each turn.

Simulation

We have provided you with several test maps that can be loaded in Gazebo. You can use these maps to test your code before putting it on the real robot. As your robot drives around the map in Gazebo, it will emulate its real world counterpart.

The maps and associated launch files for testing can be found here:
https://github.com/WPIRoboticsEngineering/RBE3002_template. Add this repository to your workspace:

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/WPIRoboticsEngineering/RBE3002_template
$ cd ~/catkin_ws
$ catkin_make
```

To make your life easier, in your `~/.bashrc`, you want to add the following:

```
$ export TURTLEBOT3_MODEL=burger
```

This would automatically identify your model is burger.

To test stuff out, use these commands.

- If you would like to control the robot with your keyboard, launch the teleop node to control the robot using the following command:

  ```
  $ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
  ```

- Launch the simulation using the following command:

  ```
  $ roslaunch RBE3002_template maze_sim.launch
  ```

- Launch `Gmapping` + `move_base` using the following command:

  ```
  $ roslaunch RBE3002_template practice_exploration.launch
  ```

Go through the launch file and copy the files of the template into your Lab4 package.

Grading Rubric

Potential Penalties

If you cannot generate a map automatically, you may manually use a map for a 25% penalty.

If your code works for the non-noisy robot (the original Turtlebot) but no the noisy robot, you will receive a 40% penalty.

If you use Move Base for the navigation stack, you will receive a 40% penalty.

Sign-Offs

Sign off	Points
1) Visualize map in Rviz	15
2) Build full map	30
3) Go back to starting point	15
4) Go to difficult point in map chosen by us	25
5) Go back home once again	15
Extra credit: the fastest robot to complete 2)	30
Extra credit: the second fastest robot to complete 2)	10
Extra credit: the fastest robot to complete 4) and 5)	30
Extra credit: the second fastest robot to complete 4) and 5)	10

Final Report

Description	Points
Introduction effectively presents the objectives and purpose of the lab	10
Methodology gives enough details to allow for replication of procedure	30
Results open with an effective statement of overall findings, presents visuals clearly and accurately, presents findings clearly and with sufficient support	20
Discussion opens with an effective statement on the goals of the lab, backs up the statement with reference to appropriate findings, provides the sufficient and logical explanation for the statement, and addresses other issues pertinent to lab	20
Conclusion convincingly describes what has been learned in the lab	10
References are included and correctly referenced	5
Grammar and spelling are correct	5

Point Breakdown

$50\% \rightarrow$ Final Report

$50\% \rightarrow$ Signoffs