ROS-I Basic Training "Mobile Manipulation"
ROS Gmapping / AMCL Tutorial

Instructor: M. Stuettgen, N. Limpert

August 9, 2017

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Contents

1 Introduction **2**

2 Terminal usage **2**

3 Gmapping **2**
3.1 Configuration 2
3.2 Usage 3

4 Map Server **4**
4.1 Map Saver 4
4.2 Map Server 4
4.3 Cosmetics 4

5 Adaptive Monte Carlo Localization **4**
5.1 Configuration 5
5.2 Usage 6

1 Introduction

During this tutorial, you will learn how to create a 2D map from incoming laserscans using the ROS package *gmapping*. This map will then be used to localize the Robot using the ROS package *AMCL*.

- Lines beginning with $ are terminal commands
- Lines beginning with # indicate the syntax of the commands

2 Terminal usage

- opening a new terminal : ctrl+alt+t
- opening a new tab inside an existing terminal : ctrl+shift+t
- killing an active process inside a terminal: ctrl+c

3 Gmapping

Gmapping is a ROS wrapper for OpenSlam's Gmapping. The gmapping package provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam_gmapping. Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

3.1 Configuration

The slam_gmapping node takes in <sensor_msgs/LaserScan> messages and builds a map (<nav_msgs/OccupancyGrid>). For configuration, the names of the following frames / transforms are required:

- base frame of the robot (default: "base_link")
- odometry frame of the robot (default: "odom")

- map frame (default: "map")

And the following topics are subscribed:

- transforms (default: "/tf")
- laser scan (default: "/scan")

Before setting up a launchfile, make sure you checked the actual names using the known tools, because they might be different from robot to robot!
Now go to your package and create a launchfile called *gmapping.launch* with the following content:

```
<launch>
  <node name="gmapping" pkg="gmapping" type="slam_gmapping"
                          output="screen">
    <param name="base_frame" value="<name-of-base-frame>" />
    <param name="map_frame" value="<name-of-map-frame>" />
    <param name="odom_frame" value="<name-of-odometry-frame>" />
    <remap from="tf" to="<name-of-tf-topic>"
    <remap from="scan" to="<name-of-laserscan-topic>"
    </node>
</launch>
```

3.2 Usage

After successful configuration, make sure that the robot is up and running, then start gmapping:

```
  $ roslaunch myfirstpackage gmapping.launch
```

Add "map" in RVIZ and watch how the map is build depending on incoming laserscans and odometry!

4 Map Server

The map_server package provides the map_server ROS Node, which offers map data as a ROS Service. It also provides the map_saver command-line utility, which allows dynamically generated maps to be saved to file.

4.1 Map Saver

The map_saver node retrieves map data and writes it out to map.pgm and map.yaml. Use the map_saver node to save your map to disk:

```
$ rosrun map_server map_saver -f mymap
```

4.2 Map Server

A saved map can be provided by running the map_server node:

```
$ rosrun map_server map_server mymap.yaml
```

4.3 Cosmetics

If you still got time you can try to improve the quality of the map by editing it with an image manipulation tool (e.g. gimp) to get rid of sensor missreadings or obstacles, you do not want to be permanent in the map. The better the map, the better the localization!

5 Adaptive Monte Carlo Localization

AMCL is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map.

5.1 Configuration

AMCL takes in a laser-based map, laser scans, and transform messages, and outputs pose estimates. On startup, amcl initializes its particle filter according to the parameters provided. The following topics are subscribed:

- transforms (default: "/tf")
- laser scan (default: "/scan")
- initialpose (default: "/initialpose")
- map (default: "/map")

Before setting up a launchfile, make sure you checked the actual names of the topics or frames using the introduced tools, because they might be different from robot to robot!

Now go to your package and create two launchfiles called *mapserver.launch* and *amcl.launch* with the following content:
mapserver.launch:

```
<launch>
  <arg name="map_file" default=
    "$(find [package where your yaml file is])/mymap.yaml"/>

  <!-- Run the map server -->
  <node name="map_server" pkg="map_server"
      type="map_server" args="$(arg map_file)" />

</launch>
```

amcl.launch:

```
<launch>
  <node pkg="amcl" type="amcl" name="amcl" output="screen">
    <param name="odom_model_type" value="omni"/>
    <param name="odom_frame_id" value="<name-of-odom-frame>/>
    <param name="base_frame_id" value="<name-of-base-frame>"/>
    <param name="global_frame_id" value="<name-of-map-frame>"/>
    <remap from="scan" to="<laserscan-topic>"/>
  </node>
</launch>
```

5.2 Usage

After successful configuration, make sure that the real robot or the simulation is up and running, then start the map server and amcl:

```
$ roslaunch myfirstpackage mapserver.launch
$ roslaunch myfirstpackage amcl.launch
```

Startup RVIZ:

```
$ rosrun rviz rviz
```

In RVIZ, add "PoseArray" to be displayed, then use the "2D Pose Estimate" button the set the inital pose for the robot inside the map. Drive around and watch AMCL doing its work!