

# ROS-I Basic Training “Mobility”

## ROS Gmapping Tutorial

Instructor: Marcel Stuetzgen

August 1, 2017



---

---

## Contents

|          |                         |          |
|----------|-------------------------|----------|
| <b>1</b> | <b>Introduction</b>     | <b>2</b> |
| <b>2</b> | <b>Terminal usage</b>   | <b>2</b> |
| <b>3</b> | <b>Gmapping</b>         | <b>2</b> |
| 3.1      | Configuration . . . . . | 2        |
| 3.2      | Usage . . . . .         | 3        |
| <b>4</b> | <b>Map Server</b>       | <b>4</b> |
| 4.1      | Map Saver . . . . .     | 4        |
| 4.2      | Map Server . . . . .    | 4        |
| <b>5</b> | <b>Cosmetics</b>        | <b>4</b> |

## 1 Introduction

During this tutorial, you will learn how to create a 2D map from incoming laserscans using the ROS package *gmapping*

- 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
```

## 5 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 (tomorrow)!