

Lab Session 4 – Simulated Turtlebot

The TurtleBot comes with ROS software that enables the robot to make a path plan in the environment, to follow the planned path, and to localize the robot during motion. Of course, to make a plan, a map of the environment is needed. The TurtleBot also has software to create the map itself. The procedure is like this:

1. **Mapping:** You tele-operate the robot (using the keyboard or a joystick) in the environment. While you do that, the robot uses its 3D sensor to create a map. You save this map to a file. After this point, as long as the environment does not change (i.e. the obstacles do not move significantly) the robot can use this map for navigating autonomously in the environment. If the environment changes significantly, you will need to re-map.
2. **Navigation:** The robot navigates autonomously using the existing map. You can give a goal pose to the robot, and it can plan a collision-free path and then follow the path. To know where it is in the environment, the robot uses its 3D sensor for localisation.

The purpose of this lab session is to introduce you with activating the mapping and navigation software on the TurtleBot.

Many of the functionality you will use in this lab comes from the `turtlebot_navigation` package. Here is the wiki page for this package: http://wiki.ros.org/turtlebot_navigation Please bookmark this page, as it will be useful for you when you work on your project. This lab session loosely follows the two tutorials given at the bottom of that page. First, we will perform mapping and navigation in the simulated Turtlebot environment. Later we will perform similar steps to map a real environment and navigate it autonomously using a Turtlebot.

Set up the singularity environment

Using steps described in lab 1, create a singularity container and then a few Ubuntu terminals to use for later.

Making sure your git clone is up-to-date

Before you do anything else, make sure your git clone of lab4 is up-to-date:

```
cd $HOME/catkin_ws/src/lab4
```

```
git pull
```

New simulated World for mapping

Start the Turtlebot simulation using the `lab4.world` file. To do this, execute in an Ubuntu terminal:

```
export TURTLEBOT_GAZEBO_WORLD_FILE=$HOME/catkin_ws/src/lab4/launch/lab4.world
```

```
roslaunch turtlebot_gazebo turtlebot_world.launch
```

(The first time you try to use this new world file, the simulator may take a few minutes to load. Please be patient.)

Tele-operating the TurtleBot

To map the environment you need to move the robot around. The `turtlebot_teleop` package is used to tele-operate the TurtleBot. We will use the keyboard for teleoperation. In an Ubuntu terminal, execute:

```
roslaunch turtlebot_teleop keyboard_teleop.launch
```

Follow the instructions to move the robot around.

(You can use the exact same program to tele-operate the real TurtleBot. You are able to do this, because the simulated TurtleBot and the real Turtlebot are listening to the same message topics coming from the tele-operation software. The tele-operation software does not know whether it is the real or the simulated robot listening; it does not need to know!)

To build a map of the environment, you will need to move the robot around using the method described above. But first, we need to start the mapping software.

Mapping

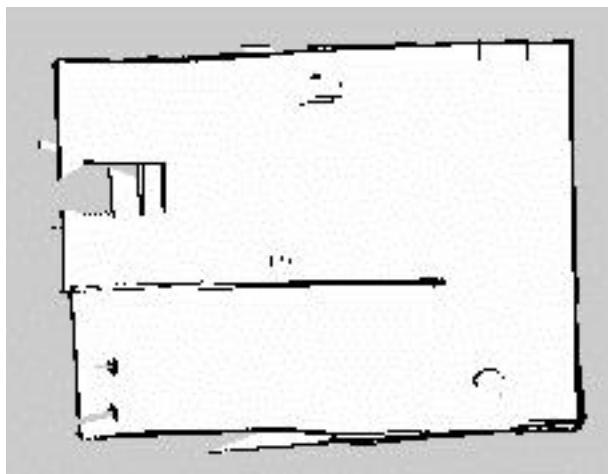
To start the mapping functionality in the simulator, roslaunch the `simulated_mapping.launch` file:

```
roslaunch $HOME/catkin_ws/src/lab4/launch/simulated_mapping.launch
```

Once the gmapping has been launched properly it is time to start the Rviz visualisation so that you can see the map being created by the TurtleBot. You must roslaunch the `view_navigation.launch` file found in the `turtlebot_rviz_launchers` package:

```
roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Now you should drive (using the keyboard) the TurtleBot in its environment by making sure its 3D sensor sees obstacles and adds them to the map. You can track this process in Rviz. This is a process that may take you 5+ minutes. You need to make sure your robot sees all edges, corners, and obstacles in the environment. You should make a few rounds in the environment so that the robot collects as much information as possible. As you drive, do not bump into any obstacles or walls, as this may change the shape of the environment and invalidate your map. At the end, your map should look something like this:



Once you are satisfied with the map you see in Rviz, you must save this map to a file. To do this, execute the following in a separate Ubuntu terminal window:

```
roslaunch map_server map_saver -f <full file path and name>
```

Where the file path and name could be something like:

```
$HOME/catkin_ws/src/lab4/maps/mymap
```

This should save a yaml and a pgm file. The pgm is your map. You can view it as an image file. The yaml file is a text file that has meta information about your map. (If you want to move your map files somewhere else, then make sure to move both the pgm and yaml files. Also open the yaml file and notice that it holds the path to the pgm file. If this path changes, update your yaml file accordingly.)

Congratulations! You have created a map of the TurtleBot's environment. It can now use this map for autonomous navigation. You can now shutdown (ctrl-c) the mapping, rviz, and the teleoperation processes. (If you have failed to create a good looking map, you can later use the map we have provided: `examplemap.yaml` and `examplemap.pgm` . But you should practice and learn how to create a map as you will need to do this on the real robot and for your project as well.)

Autonomous navigation

Now that you have built a map, it is time to get the robot moving around the map autonomously within its environment.

Note that, whenever you want the robot to navigate autonomously, you should stop the teleoperation program (the one you started above with "`roslaunch turtlebot_teleop keyboard_teleop.launch`") since it interferes with autonomous navigation of the robot.

Change your directory to go to the lab4/launch:

```
cd $HOME/catkin_ws/src/lab4/launch
```

and run this command replacing the map file with the correct filepath and name for your map file:

```
roslaunch simulated_localisation.launch map_file:= <path-to-my-map-yaml-file>
```

where <path-to-my-map-yaml-file> may be:

```
$HOME/catkin_ws/src/lab4/maps/mymap.yaml
```

Wait until you see the text "odom received!" on this terminal. (If you do not see it, scroll up and if you see a red error message saying "Map_server could not open" your file, then you were not able to specify your file path correctly. For this command to work, you have to supply the **full path** to the map yaml file. Another possible error is if you moved your map files after saving them and did not update the yaml file to reflect the new location of the pgm file.)

Then, start a new Rviz:

```
roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Once you have opened Rviz you should see the map loaded into it. Now you will need to localise the robot within the map, so that it knows roughly where it is starting. Look at the simulator window

(Gazebo) and note the robot location in the map, including its heading direction. Then, in Rviz click the '2D pose estimate' button, and then click on that location on the map and point the robot in the correct direction, so the robot knows roughly where it is and what direction it is heading.

You can now send goals to the TurtleBot using the '2D nav goal' button, click it then choose a point on the map and a direction to face upon reaching it. The robot will navigate towards it, avoiding obstacles on the way.

After the robot reaches its goal, you can send other new goals.

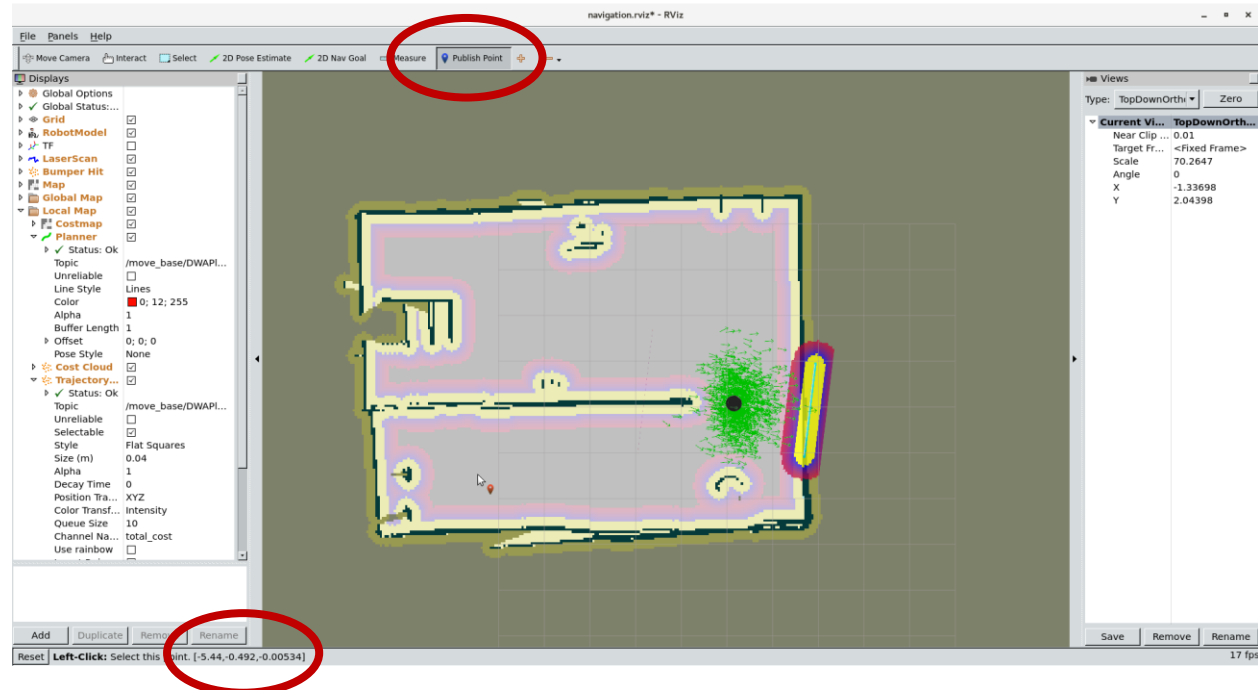
Congratulations! Your robot is autonomously navigating now.

[Sending navigation goals through Python](#)

Above, we sent navigation goals using the RViz window. You can also send goals to your Turtlebot programmatically through Python. Under lab4/src, you will find the `go_to_specific_point_on_map.py` script. Please open this script.

Between lines 84-86, you can specify x, y and theta (rotation) coordinates to your robot.

To identify the x,y coordinates of a point you desire to move to, in RViz, click the “Publish Point” option at the top menu, and hover your mouse over the map in RViz. As you hover your mouse, RViz will be displaying the (x,y,z) values of the point your mouse is on in the lower-left corner of the RViz window. See below for an example.



Using this method, you can determine the x,y coordinate you want to specify in the script. First try setting theta to zero (0) to determine which direction in your map corresponds to zero rotation. Then you can change theta as you wish. Execute the script and your robot should plan and move to the goal you specify.

Lab Session 4 – Real Turtlebot

Initializing your group's git repo

If you do not remember how to do this, please read the section “Initializing your group's git repo” from the lab1 worksheet, and initialize your group's git repo for lab4.

Setup Turtlebot laptop for your group

If you do not remember how to do this, please read the section “Setup Turtlebot laptop for your group” from the lab1 worksheet.

Tele-operating the TurtleBot

Turn on the TurtleBot and run the minimal launch file.

The `turtlebot_teleop` package is used to tele-operate the TurtleBot. We will move the robot around using the keyboard. (If you have a second remote laptop using which you can ssh into the TurtleBot laptop, it would be best to use the remote laptop for keyboard control. See the Appendix at the end of this document to learn how to do this.)

```
roslaunch turtlebot_teleop keyboard_teleop.launch
```

Mapping

All the software for TurtleBot navigation, including the mapping, localisation and planning, are in the `turtlebot_navigation` package. To start the mapping functionality, run:

```
roslaunch turtlebot_navigation gmapping_demo.launch
```

The terminal should display info informing you that it has detected the sensor and is receiving input. (Note that this `gmapping_demo.launch` file starts the 3D sensor, so there is no need to launch the 3D sensor as we did in previous lab sessions.)

Once the `gmapping` has been launched properly it is time to start the visualisation so that you can see the map being created by the TurtleBot. You must `roslaunch` the `view_navigation.launch` file found in the `turtlebot_rviz_launchers` package:

```
roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Now you should drive (using the keyboard) the TurtleBot in its environment by making sure its 3D sensor sees obstacles and adds them to the map. You can track this process in Rviz. Once you are satisfied with the map you see in Rviz, you must save this map to a file:

```
roslaunch map_server map_saver -f <full file path and name>
```

Where the file path and name could be something like:

```
$HOME/catkin_ws/src/lab4/maps/mymap
```

Congratulations! You have created a map of the TurtleBot's environment. It can now use this map for autonomous operation. You can now shutdown (ctrl-c) the mapping, rviz, and the teleoperation processes.

Autonomous navigation

Now that you have built a map, it's time to get the robot moving around it autonomously within its environment. (Please stop the teleoperation program at this point, since it interferes with autonomous navigation of the robot.) Run this command replacing the map file with the correct filepath and name for your map:

```
roslaunch turtlebot_navigation amcl_demo.launch map_file:=<path-to-my-map-yaml-file>
```

where <path-to-my-map-yaml-file> may be:

```
$HOME/catkin_ws/src/lab4/maps/mymap.yaml
```

Wait until you see the text "odom received!" on this terminal. (If you do not see it, scroll up and if you see a red error message saying "Map_server could not open" your file, then you were not able to specify your file path correctly. For this command to work, you have to supply the **full path** to the map yaml file. Another possible error is if you moved your map files after saving them and did not update the yaml file to reflect the new location of the pgm file.)

Start a new Rviz:

```
roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Once you have opened Rviz you will need to localise the robot within the map, so that it knows roughly where it is starting. In Rviz click the '2D pose estimate' button, then click the location on the map and point the robot in the correct direction, so the robot knows roughly where it is and what direction it is heading.

You can now send goals to the TurtleBot using the '2D nav goal' button, click it then choose a point on the map and a direction to face upon reaching it. The robot will navigate towards it, avoiding obstacles on the way.

After the robot reaches its goal, you can send other new goals.

Sending navigation goals through Python

You can use the same script you used in the Simulation Lab to send navigation goals through Python.

Do not forget to push any change in your files to git! Otherwise you will lose them when you logout of the Turtlebot laptop!

Appendix.

Using a remote laptop to connect to the Turtlebot laptop

Here will use the term “remote laptop” to refer to a separate laptop you may want to use in addition to the “Turtlebot laptop” that you have been distributed. It may be convenient to use a remote laptop, for example when you are remotely tele-operating the Turtlebot.

Make sure both laptops are on eduroam.

On the Turtlebot laptop, in a terminal, execute

```
ifconfig
```

Under wlan0, make a note of the IP address (inet addr).

On the remote laptop, open a terminal Then execute:

```
ssh turtlebot@<IP_ADDRESS_OF_TURTLEBOT_LAPTOP>
```

where you replace <IP_ADDRESS_OF_TURTLEBOT_LAPTOP> with the IP address you have noted.

(If the remote laptop is running Windows, you can use a SSH client, e.g. PuTTY.)

After you successfully ssh’ed, you can run ROS commands like you were on the Turtlebot laptop. For example you can run:

```
roslaunch turtlebot_teleop keyboard_teleop.launch
```