Workshop "ROS for Engineers" - part 2

Original author (Dutch): Eric Dortmans (e.dortmans@fontys.nl)

Translation / Update (English): Kris Piters (k.piters@fontys.nl)

Preparation

Update the ros_examples package you installed in the previous exercise:

```
cd ~/catkin_ws/src/ros_examples
git pull
cd ~/catkin_ws
catkin_make
```

Install the packages needed for simulating the Turtlebot:

```
sudo apt-get install ros-kinetic-turtlebot
sudo apt-get install ros-kinetic-turtlebot-apps
sudo apt-get install ros-kinetic-turtlebot-rviz-launchers
sudo apt-get install ros-kinetic-turtlebot-simulator
```

Install extra packages for use with Stage:

```
sudo apt-get install ros-kinetic-mouse-teleop
sudo apt-get install ros-kinetic-teleop-twist-keyboard
sudo apt-get install ros-kinetic-stage
sudo apt-get install ros-kinetic-stage-ros
```

Controlling a simulated robot in Stage

In this chapter we'll use Stage, a 2D robot simulator.

Start a simulated robot in a simulated world:

```
roslaunch stage_worlds kinect_world.launch
```

Use another terminal window to start the *teleop_twist_keyboard* node for controlling the robot:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Use another window to analyze the odometry information published by the robot:

```
rostopic echo /odom
```

Use the following command to see the structure of a *Odometry* message:

```
rostopic type /odom | rosmsg show
```

Watch how the odometry information will change by driving around.

Start *rviz* for extra visualization of data produced by the robot:

```
roslaunch stage_worlds rviz_stage.launch
```

TF

Take a look at the *tf tree* while the simulated world is active:

```
rosrun rqt_tf_tree rqt_tf_tree
```

Alternatively use rqt and its tf tree plugin:

```
rqt

Plugins > Visualization > TF Tree
```

Take a look at what's being published to the tf tree with tf_monitor:

```
rosrun tf tf_monitor
```

Request specific, single transformations with tf_echo:

```
rosrun tf tf_echo /base_link /base_laser_link
rosrun tf tf_echo /base_footprint /base_link
rosrun tf tf_echo /odom /base_footprint
```

You can also request complex, multiple transformations, e.g.:

```
rosrun tf tf_echo /odom /base_laser_link
```

Navigation

Stop your running nodes (Stage, Rviz, etc). To be sure you could close all terminal windows.

We are now going to experiment with the ROS Navigation stack.

Start *gmapping* to make a map of the (simulated) robot world:

```
roslaunch stage_navigation kinect_gmapping.launch
```

Start the *teleop_twist_keyboard* **OR** *mouse_teleop* node to control the robot:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py

roslaunch mouse_teleop mouse_teleop.launch
   mouse_vel:=cmd_vel
```

You can also "move" the robot by selecting it with your mouse and dragging it around in Stage.

Drive the robot around in the simulated world, regulary returning to where you've been before. In the RViz window you'll be able see how *gmapping* tries to make a map of the world.

You'll notice it's not easy to make a good map!

We will try another robot. Stop the running simulation and start the following:

```
roslaunch stage_navigation laser_gmapping.launch
```

- Can you see and notice a difference?
- What can you conclude from your observations, having compared both results?

If you are satisfied with your map, you can save it using map_saver:

```
rosrun map_server map_saver -f /tmp/world_map
```

Stop all your nodes (if you want to be save, close all terminal windows).

Now that we have created and saved a map of the environment, the robot will be able to find it's own way based on this map. We'll use *amcl* for localization and *move_base* for navigation:

```
roslaunch stage_navigation kinect_amcl.launch
   map_file:=/tmp/world_map.yaml
```

Take a look at the Computation Graph:

```
rqt_graph
```

Use the *2D Nav Goal* button in Rviz to give the robot a navigation goal. First click the button, then click somewhere on the map (dragging before letting go of your click will allow you to set a goal orientation). As an example you could send your robot from end of the map to the other.

Stop your running nodes before continuing.

Controlling a simulated Turtlebot in Stage

Next, we'll simulate and control a robot that exists for real: the Turtlebot.

Start the Turtlebot software:

```
roslaunch turtlebot_stage turtlebot_in_stage.launch
```

Take a look at the Computation Graph:

```
rqt_graph
```

What could be the function of the *cmd_vel_mux* node? Take a look at it's parameter (YAML) file:

```
cat `rospack find turtlebot_bringup`/param/mux.yaml
```

Use your keyboard to control the robot:

```
roslaunch turtlebot_teleop keyboard_teleop.launch
```

Just like the robot before, the Turtlebot is also able to navigate autonomously. Try sending it a navigation goal by using the *2D Nav Goal* button in RViz. As you might have noticed the Turtlebot is not moving. Combine the Computation Graph, your newly acquired knowledge of the *cmd_vel_mux* parameters, and *rostopic echo* to make the Turtlebot move autonomously (Hint: priorities).

The Turtlebot itself has been visualised nicely in Rviz. In part 3 of the workshop we'll learn how to do this.

EXTRA: Controlling a simulated Turtlebot in Gazebo

Gazebo is a 3D Simulator. Running this simulator requires more computing power (both graphics and CPU), partly due to its physics engine.

Start the Gazebo simulator:

```
roslaunch turtlebot_gazebo turtlebot_world.launch
```

Run the *gmapping* node:

```
roslaunch turtlebot_gazebo gmapping_demo.launch
```

Visualise the robot during navigation:

```
roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Start the *teleop_twist_keyboard* **OR** *mouse_teleop* node to control the robot:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py

roslaunch mouse_teleop mouse_teleop.launch
   mouse_vel:=cmd_vel
```

The simulation also includes an 'actual' 3D sensor. Try adding a camera view in RViz for the topic */camera/rgb/image_raw*.

EXTRA: Navigation with a real Turtlebot

For this last assignment you need to set your *ROS_IP* and *ROS_MASTER_URI*.

The **Turtlebot tutorials** for navigation on the ROS wiki will provide you with information on how to setup and use navigation with an actual Turtlebot.

References

- ROS Tutorials
- TF
- Introduction to TF
- rqt
- navigation
- TurtleBot
- turtlebot_simulator
- turtlebot_navigation