Lab 7: Intro to Gazebo and Rviz

CSCI 3302: Introduction to Robotics

Report due 4/7/20 @ 23:59pm

Total score: 50% of previous labs

(No early turn-in bonus)

The goals of this lab are:

- Get familiar with Gazebo and Rviz tools
- Get familiar with launch files
- Learn to use a powerful mapping tool
- Use basic ROS navigation stack

You need:

- A system running ROS Melodic (see Lab 0)

Overview

In practice, we don't directly deploy the code on a real robot but test it rigorously on the simulator first since robots are very expensive and actions have consequences in the real world. **Gazebo** (http://gazebosim.org/) and **Rviz** (http://wiki.ros.org/rviz) are two popular GUI tools that help in robotics development. They provide simulated worlds, sensors, robots etc. Typically, the world is scanned using a variant of SLAM (Simultaneous Localization and Mapping, https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping) whose theory is out of scope for now but we will learn to use it in simulator so that we can map the world. ROS also has its own navigation stack which helps in planning and control and comes with a basic planner so that we don't have to build basic planners and controllers if our focus is something different.

Instructions

Each group must develop their own software implementation and turn in a **single report** that contains:

- The code (1 .py file)

- One individual lab report in PDF format. Please put the number of each question next to your answers, rather than turning in your answers as an essay
- A video of Turtlebot3 reaching the given goal showing both Rviz and Gazebo windows. If your video file is bigger than 50mb, please submit a link to google drive or some other cloud service of your choice.

If your group does not finish the implementation by the end of the class, you may continue this lab on your own time as a complete group.

Part 1: Install the Turtlebot3 packages

```
$ sudo apt-get install ros-melodic-joy ros-melodic-teleop-twist-joy
ros-melodic-teleop-twist-keyboard ros-melodic-laser-proc ros-melodic-
rgbd-launch ros-melodic-depthimage-to-laserscan ros-melodic-rosserial-
arduino ros-melodic-rosserial-python ros-melodic-rosserial-server ros-
melodic-rosserial-client ros-melodic-rosserial-msgs ros-melodic-amcl
ros-melodic-map-server ros-melodic-move-base ros-melodic-urdf ros-
melodic-xacro ros-melodic-compressed-image-transport ros-melodic-rqt-
image-view ros-melodic-gmapping ros-melodic-navigation ros-melodic-
interactive-markers
```

```
$ cd ~/catkin_ws/src/
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
$ cd ~/catkin_ws && catkin_make
```

Part 2: Mapping

1. In this part we will launch one of the example worlds along with the robot using the turtlebot3_world.launch inside the package turtlebot3_gazebo by typing
   ```
   $ roslaunch turtlebot3_gazebo turtlebot3_world.launch
   ```

   This should launch Gazebo rendering a robot and an example world with static obstacles. RGB colored axes are XYZ axes respectively.

2. Next we will launch a laser based mapping node.
   ```
   $ roslaunch turtlebot3_slam turtlebot3_slam.launch
   slam_method:=gmapping
   ```

This should launch Rviz rendering the robot and the live laser data

3. Now in order to map this world, we need to move our robot and collect the data through laser scans.
   For that we will launch a teleoperation node to control our robot using w, a, s, d, x keys.

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

   Move your robot so that it can 'see' the gazebo world. Remember to use key 's' to stop the motion. The w-s keys change the velocity and and the a-d keys change the angular velocity.

4. Now we want to save the map

   ```
   $ rosrun map_server map_saver -f ~/map
   ```

   By default the map is saved in the /home/username folder. It has 2 files. A .pgm file which has a 8-bit grayscale image of the world as seen from top. A .yamp file whose description can be read from http://wiki.ros.org/map_server

   We now have mapped our world. You can use Dijkstra's to plan using this 2d image of the top view which you have already done in the past. Most of the ROS based mobile robots are controlled using a topic **/cmd_vel** which allows only velocity and angular velocity values. You can design a controller that publishes on this topic to move the robot.
   Example - http://wiki.ros.org/Robots/TIAGo/Tutorials/motions/cmd_vel
   But for this lab we will use ROS's Navigation stack which has some in-built planners that take a goal, plan and execute the plan to reach the goal.

Part 3: ROS Navigation Stack

1. Kill all the processes. This is important.
2. Now we need to set the initial robot pose to be the same in Gazebo and the navigation stack. For that, edit
   `~/catkin_ws/src/turtlebot3/turtlebot3_navigation/launch/amcl.launch`
   to have the same initial pose as the file
   `~/catkin_ws/src/turtlebot3_simulations/turtlebot3_gazebo/launch/turtlebot3_world.launch`

3. Feed map to the navigation stack - If you open the
   `~/catkin_ws/src/turtlebot3/turtlebot3_navigation/launch/turtlebot3_navigation.launch`
   you can find the "map_file" argument which looks for the map files that you just saved at a particular default location. Copy your map files there or change the address to point it to the map location

4. We will now restart our robot in gazebo
   ```
   $ roslaunch turtlebot3_gazebo turtlebot3_world.launch
   ```
   and also start the navigation stack by
   ```
   $ roslaunch turtlebot3_navigation turtlebot3_navigation.launch
   ```

   You can uncheck the local and global map boxes for a clearer visualization.

5. Your job is to now develop a python script that takes the goal pose (x, y, θ) as arguments, and uses the navigation stack to publish the goal to **/move_base_simple/goal** topic.
   Use `rostopic info topic_name` to find out the message type and search the documentation/examples to create a message.
   (The 'frame_id' field would be 'map' because you cannot provide a goal without referencing to a frame of reference. In this case it's the 'map' that you can see in Rviz Global Options > Fixed Frame which is the one that we saved.)
   Your code should have the 3 arguments -x, -y, -theta. If an argument is not supplied handle it in the code.

6. Extra - If you want you can avoid using the navigation stack in favour of a more robust, fast and advance planning library MoveIt.
   MoveIt is a very popular library to specially deal with robotic arms which have more degrees of freedom compared to a mobile robot

Part 4: Visualizing sensor data

1. You can access any data through subscribing to a topic but Rviz gives a neat way to visualize it. Click the **Add** button, go to **By Topic** tab and select **/camera/rgb/image_raw/camera** and click **OK**. You should see the camera stream in Rviz. You can turn off the LaserScan if it obstructs your view.

Part 5: Lab Report

1. Run your python code for **x = 0.5** and **theta = 1.0** (note that the rotation is provided in a rather weird way. It's represented as a Quaternion which is a popular way to represent angles).
   For example this could look like - `yourCode.py -x 0.5 -theta 1.0`
   Make the video of the robot starting and reaching the goal.

2. Part 2.3 - Provide a screenshot of the mapping midway through the process where you haven't mapped the complete world.

3. Part 2.4 - What is the map's resolution in terms of meters per pixel? Also what is the pose of the map's lower-left corner with respect to the world frame?

4. What is the default initial pose set in Part 3.2?

5. What are the launch files that are launched hierarchically when you launch `turtlebot3_world.launch` and `turtlebot3_navigation.launch`.

   Provide your answers for both the files in an hierarchical format.
   For example -

```
X.launch ---- Y.launch
              |-- Z.launch
              |-- A.launch
                  |--B.launch
```

6. Write a one-line summary describing the main function of each of the launch files that you mention above.

7. Provide a screenshot of your camera running in Rviz from Part 4.

8. What are the new topics that spawned after launching `turtlebot3_navigation.launch`?

9. How much time did you spend doing this lab?

To be (optionally) submitted separately via e-mail to Prof. Roncone: aroncone@colorado.edu

(Optional, confidential, not for credit) A brief description of any problems (either technical or collaborative) encountered while performing this lab (e.g., issues with the clarity of instructions, clarity of documentation, lab colleague’s behavior, etc.)