

AA 274A: Principles of Robot Autonomy I

Section 4: Visualizing Information from Robots

Our goals for this section:

1. Potentially become familiar with catkin package installation
2. Become familiar with tools for visualizing information from your robots.
3. Write a marker using rviz.

1 Package Installation

Before we dive into information visualization, we may have to install some packages to endow our robots with the necessary capabilities we're looking for. Chiefly, some of your turtlebots do not have the **velodyne** catkin package installed. This poses a problem since, without it, we cannot use the scans coming out of the velodyne.

First, we need to check to see if the package is already in our catkin workspace. We can do this by checking the contents of the `catkin_ws/src` directory

```
1 | cd catkin_ws/src
2 | ls
```

If you see a folder named **velodyne** then you already have the package.

Already have the package: Great! Let's update it to make sure it's up to date.

```
1 | cd velodyne
2 | git pull
```

Don't have the package: Great! Let's get it now. To add a new package to the our catkin workspace, simply clone the package from where it resides to the `catkin_ws/src` directory. Fortunately for us, the **velodyne** package is easily-accessible through GitHub. To obtain it, execute the following from within the `catkin_ws/src` directory (you can check this by executing `pwd`):

```
1 | git clone https://github.com/ros-drivers/velodyne.git
```

Build the package: Once the package has finished cloning/pulling, we need to rebuild our catkin workspace. Recall, we can do this as follows

```
1 | cd .. # We want to be within the catkin_ws directory
2 | catkin_make
```

This will take some time, so leave it alone until it's fully finished. Once it completes successfully, rejoice! You now have the necessary packages to move on.

2 Information Visualization with rviz

rviz is a 3D visualization tool for ROS, visualizing information such as maps and point clouds in a much more intuitive way compared to `rostopic echo`.

2.1 Full Turtlebot bring up

Start roscore:

```
1 | roscore -p $ROS_PORT
```

In a new terminal tab or window, run the Gazebo environment:

```
1 | roslaunch asl_turtlebot section4_demo.launch
```

Problem 1: Once this is all running, which rostopics are available? Paste this list in your submission.

2.2 Starting up rviz

Open another terminal and execute `rviz`.

2.3 Populating rviz

Now that we have `rviz` running, let's quickly check that we can actually see the entire `rviz` GUI (this is a potential problem with TurboVNC and screen sizes). If you cannot scroll down and see the display's Reset button, then you cannot see the full `rviz` screen. To fix this, please click **TurboVNC Viewer -> Preferences -> Connection** and change the **Remote desktop size** to a larger value. This should bring up scroll bars on the bottom and right side of the TurboVNC screen with which you can scroll to see the full display.

Now let's visualize some data!!

2.3.1 Point Cloud

In `rviz`, to add a topic to the display, you need to click on the "Add" button on the bottom left. Once you've clicked this, a popup will come up. In the popup, click the "By topic" tab and scroll down until you see "LaserScan" under `/scan`. Double-click this and watch the scanned points appear.

2.3.2 Camera Image

Next, follow the same procedure, but this time expand `/camera /image_raw /compressed` and double-click on Image. This is a rendered view of the camera attached to the Turtlebot in Gazebo.

2.3.3 gmapping's Environment Map

Next, add "Map" under `/map`. This is what the mapping package we use provides us from LIDAR data.

2.3.4 gmapping's Odometry

Next, add "Odometry" under `/odom`. This is what the mapping package we use provides us as the robot's odometry from LIDAR data.

2.3.5 Coordinate Axes

Next, add “Axes” from under the “By display type” tab. This shows us where the world origin point is and the orientation of the axes.

2.3.6 tf’s Transform Tree

Next, add “TF” from under the “By display type” tab. This shows us the axes origins and orientations of any other frames we’ve defined. You can see we’ve defined quite a few for this robot.

Problem 2: Take a screenshot of your rviz display after all of the above are running. Teleop the robot (using `roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch`) to move it to a different place and see your rviz display change.

2.4 Saving rviz’s Current Configuration

That was quite a lot of topics to open, imagine doing that every time you wanted to debug your ROS stack! Thankfully, there’s a way we can save this current configuration to a file and open it with `rviz`, using **File -> Save Config As**.

Problem 3: Paste the contents of your created .rviz configuration file into your submission.

Close and reopen `rviz`, load the config file you just created so you can see that it reloads all of your previously-opened topics.

3 rviz Markers

Markers are a very helpful tool for visualizing intermediate rewards, computed trajectories, etc. all in `rviz`! The way markers work is by taking advantage of `rviz`’s built-in handling of ROS topics and messages. To create a Marker, we’ll publish a Marker message to a topic and then view the topic in `rviz`.

Code to create a Marker object can be found in the `code/` directory of today’s section. Running it creates a green oblong sphere at world coordinates (1, 1, 1), each in meters.

Problem 4: Change the Marker to look like a normal red sphere and place it 1m in front of the robot (think about which axis this corresponds to), include a screenshot of your marker and its placement in your submission.

4 Cleanup

This time, you do not have to remove any files (you’ll want to leave the `velodyne` catkin package in your catkin workspace as well as keep your `.rviz` configuration file around). Further, we have noticed that some of you are fully removing your `catkin_ws` between each section. While this was more or less inconsequential for the previous sections, from now on please **DO NOT** do this.