Assignment 1: ROS, Gazebo, and Fetch

September 4, 2018

This is our first assignment which helps you get familiar with ROS and Gazebo, while running the Fetch robot in simulation. Follow the instructions below to install and run several scripts. At several points you will be asked to take screenshots of your result and upload them through Canvas. If your computer is already equipped with ROS indigo and Ubuntu 14.04, feel free to use your own machine. We have not been able to do provide a Docker instance, but if you made one with ROS and Gazebo, feel free to share it with the rest of the class. The detailed tutorial by Fetch Robotics on which this assignment is based can be found at `http://docs.fetchrobotics.com/gazebo.html`.

Part I
Installation

The ROS indigo and Ubuntu 14.04 have been installed for you. The next step is to install the Fetch simulator. Open a command-line shell and type the following commands

```
$ sudo apt-get update
$ sudo apt-get install ros-indigo-fetch-gazebo-demo
```

This installs the package that contains some interesting demos.

Part II
Simulation

1 Start simulation in Gazebo

Start the simulator and generate a Fetch robot in simulation environment, by

```
$ roslaunch fetch_gazebo simulation.launch
```

The first time you launch this, it will take some time to obtain the model from website. Hence, if the process fails or takes a long time, press Ctrl+C to quit and relaunch again.

Watch the results and observe the Fetch robot. Probably you cannot see the Lidar scan (blue area shown in tutorial) in your simulation at the beginning. To fix this, you need to set visibility of Lidar scan in its .xacro file as true (the default value is false). To do so, we will navigate to the associated directory with “roscd”, and then edit the file:

```
$ roscd fetch_gazebo
$ sudo gedit robots/fetch.gazebo.xacro
```

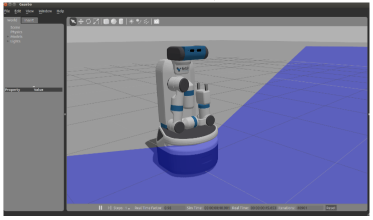
The sudo above is because the files is owned by root. Locate the <laser_link> block in the file and modify the following line:

```
<visualize>false</visualize>
```

to say instead

```
<visualize>true</visualize>
```

Save the file, relaunch, and you should see the blue lidar scan in your simulation now. You can also play around with the parameters on the left panel, click different items to see the corresponding parts of Fetch. Your simulation should be similar to this one,



- Exercise 1: Take a screen shot of the Fetch robot with the lidar scan visible.

2 Visualize in Rviz

Open a new shell (don’t close the previous one) to visualize the Fetch robot via the ROS visualization application “rviz”. You can manually set up your rviz visualization or use a predefined configuration file.

2.1 Manually set up

Type the following command line to start rviz,

```
$ rosrun rviz rviz
```

Probably you can't see anything in rviz at the beginning.

1. Click 'Add' at bottom left corner -> Under 'By display type' tab choose 'RobotModel', add it -> on left panel of rviz, change the 'Fixed Frame' from 'map' to 'base_link', you should see the Fetch robot in rviz now.

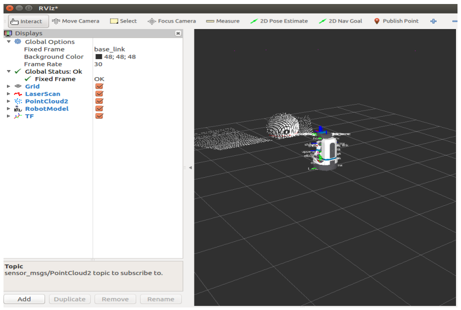
2. Add the 'TF' to visualize transformation frame, 'LaserScan' for visualizing laser , and 'PointCloud2' to generate point cloud of environment.

3. Click the left triangular button near 'LaserScan', then click the area right to the 'Topic' item, choose '/base_scan' topic to visualize the laser scan.

4. Similarly, change the topic of 'PointCloud2' as '/head_camera/depth_downsample/points'. You should see a grey area generated in front of Fetch

5. Try to add some objects into your simulation world by clicking different icons on top panel in Gazebo. Left click the icon, for example a sphere, and left click the area in front of Fetch to add that 3D sphere into your simulation world.

6. The result may be similar like this(with a sphere in front of Fetch). Take a screenshot of your simulation.



Global Options
Fixed Frame base_link
Background Color 48; 48; 48
Frame Rate 30
✓ Global Status: Ok
✓ Fixed Frame OK
Grid ☑
LaserScan ☑
PointCloud2 ☑
RobotModel ☑
TF ☑

Topic
sensor_msgs/PointCloud2 topic to subscribe to.

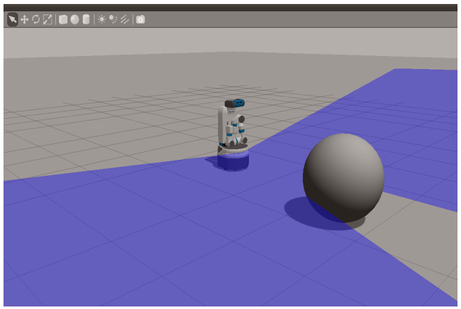
- Exercise 2: Take a screen shot of Fetch robot with TF, laser scan(red) and point clouds(white).

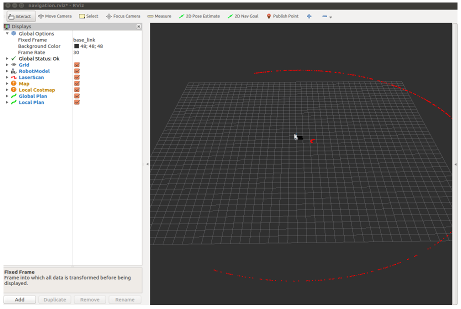
2.2 Use predefined configuration file

Instead of doing this manually, you can load default .rviz configuration file for Fetch. Use roscd again to navigate to the fetch_navigation/config directory and load the rviz file.

```
$ roscd fetch_navigation/config
$ rviz -d navigation.rviz
```

On the left panel, change the 'Fixed Frame' from 'map' to 'base_link', follow the instructions in 2.1 to change topics for visualization, and take a screen shot of your simulation. This time you will have 2 extra items added to the left panel, 'Map' and 'Local Costmap', we will skip this part and discuss it in navigation part.





Global Options
Fixed Frame base_link
Background Color ■ 48; 48; 48
Frame Rate 30
Global Status: Ok
Grid ☑
RobotModel ☑
LaserScan ☑
Map ☑
Local Costmap ☑
Global Plan ☑
Local Plan ☑

Fixed Frame
Frame into which all data is transformed before being displayed.

- Exercise 3: Take a screen shot of Fetch robot with your simulation world and corresponding rviz window[default configuration file].

3 Running mobile manipulation demo

This is a demo showing how navigation, perception and Moveit! work together to achieve some complex task. Use the following lines to run this demo.

- Open a new shell (make sure you already quit the previous ones),

  ```
  $ roslaunch fetch_gazebo playground.launch
  ```

- To start the simulation in gazebo. Open a new shell and use the following line to run the demo. You could see the Fetch robot search for object and navigate itself towards the table, it will stop in front of the blue cube and try to pick it up.

  ```
  $ roslaunch fetch_gazebo_demo demo.launch
  ```

- Open a new shell, you can use rviz to visualize this process. Use the same command to open the window and change the corresponding topics as previous steps.

  ```
  $ rosrun rviz rviz
  ```

- Exercise 4: Take a screen shot of Fetch robot in gazebo, your simulation might look like the following picture.