Tutorial 5: Sensor Configuration and Rosbags

The F1/10 Team

Introduction

This tutorial will cover the installation of various packages and settings to be configured to get the LiDAR and Inertial Measurement Unit working on Jetson. At the end we will go through few commands that can be used to record the data to run them later on for simulations.

Transcript to the Video Tutorial

In this video we will install and configure the Hokuyu LiDAR and Razor IMU. Luckily both have of these have ROS packages readily available and hence the process is quite straight forward. For any other sensor you might want to use in future, you can look up to its package availability in ROS and follow similar steps.

Connect your jetson board to display, keyboard, mouse and an ethernet cable.

On the terminal use the following command to install the Hokuyu node package:

```
sudo apt-get install ros-indigo-hokuyu-node
```

When you connect your lidar to jetson, it generally connects to ACM ports which can be checked using the command :

```
ls /dev/ttyACM*
```

This will list all the ACM ports currently being used. If you are doing this first time, you need to provide specific permissions to the port by using the command :

```
sudo chmod a+rw /dev/ttyACM_
```

This will provide read write permissions for ROS to communicate with the sensor. Now, to check the data coming from LiDAR, as explained in the initial ROS tutorials, run the roscore. On the other terminal , run - rostopic list. This will show all the topics currently being published.

The scan topic is the one we are interested in.

```
rostopic echo /scan
```

These dump of messsages might not make sense now, but these values correspond to the distance sensed by LiDAR at different angles.

We can visualize these laser scans by a GUI interface called Rviz. Unfortunately due to the driver issues Rviz doesn't work on Jetson. If you had setup ROS over Network explained in previous tutorial properly, you can run Rviz over your system that has been setup for the network.

AS you can see these are the laser scans. It might not show up on its own on your system. Insert it using the add button visible here.

Now to install the packages for IMU, go back to Jetson and install the package using command :

```
sudo apt-get install ros-indigo-razor-imu-9dof
```

Lets keep the default parameters untouched, and take a copy for us to make some tweaks

```
roscd razor_imu_9dof/config
cp razor.yaml my_razor.yaml
```

You can calibrate the IMU by following the link provided in the reference. and mention the measured offset values in the newly made file. To confirm the communication of IMU with ROS interface use the launch file:

```
roslaunch razor_imu_9dof razor-pub.launch
```

You can now run "rostopic list" in another window and look for the /imu topic. "rostopic echo /imu" will allow you to see the incoming data stream. So the packages for both these sensors are ready. You can follow similar steps to install it on any ROS system. Its better not to clog Jetson's memory with useless packages and hence you install any extra packages on your laptop.

For example to visualize the IMU data, you need to install the python-visual package. Install the IMU packages on your ROS system as well. Then install the python- visual using :

```
sudo apt-get install python-visual
```

Once done, run the launch file :

```
roslaunch razor_imu_9dof razor-pub-and-display.launch
```

The red board is following the motion of my IMU. Feel free to change the config parameters and play around with this to improve the performance.

Introduction to Rosbags

This tutorial covers data logging tools provided by the Robot Operating System. These tools allow users to capture data from various sensors and then play it back offline. This tutorial goes over some basic commands to utilise these logging tools. This should be ideally read after you go through the **Sensor Integration Video**.

Rosbags: Tutorial 5.1

In this tutorial we are going to learn about **rosbags**. You would now know how dependant we are on the various sensors for testing and running our algorithms. The problem arises when we don't always have access to these sensors. **rosbag** is the tool provided by ROS that lets us log sensor data and play it back as per our need.
It stores this sensor data as the **topics** on which it was published and when the data is played back, its published again on those same topics. Therefore the remaining part of our system need not change when we need to test anything ofline.

You could go ahead and download the files located at https://github.com/mlab-upenn/f1_10_code_public/ - Git Repo

In a terminal after initialising **roscore** run this command,

```
$ rosbag play lidarBag.bag – –clock
```

This **.bag** file contains the data from our LIDAR. After this command, on another terminal, use rostopic list to verify that you are able to see the **/scan** topic. This is the topic that the LIDAR was publishing on when it was actually used. The **clock** arguement here is a very important paramter. It provides a simulated 'clock' for the ROS network. The data playing back from the bags would have time stamps highly offset from the current time. This discrepency in time between the incoming data and the rest of the system often leads to unpredictable and generally unwanted behaviour. For this reason, the **'clock'** is used - It provides all the ROS systems with a common time stamp - a simulated time stamp. If you perform a rostopic list in a terminal, you will see the **/clock** topic that is being published. The other subsystems of your ROS framework need to be told that they must follow the simulated time and not actual time. To do this we must set the ros parameter **use_sim_time** to **true**. To do this, run the following command on a terminal

```
$ rosparam set use_sim_time
```

To check the value of the paramter use,

```
$ rosparam get use_sim_time
```

Ensure this configuration of use_sim_time every time you playback a bag.

Now we will go through the process of actually taking recording a bag. Lets use the Hokuyo and the IMU as our sensors. Launch the sensors using,

```
$ rosrun hokuyo_node hokuyo_node
$ roslaunch razor imu_launch
```

For bagging all current topics, use the **-a** command line argument with the command

```
$ rosbag record –a
```

You can choose to terminate the bagging process at any time by simply killing that process - **Ctrl + C** should do it.
You will notice now that you will have a bag in your current folder. Use

```
$ rosbag <bag name>.bag info
```

to view the contents of the bag. This data can be then played with the command shown earlier to simulate the LIDAR data ofline.

Hopefully you guys understood the need and usefullness of the bagging process. This tutorial like the others was a very concise version of the Official tutorials on the ROS Wiki. Please visit [1] for more commands and options regarding the rosbag tool.

References

[1] http://wiki.ros.org/rosbag - Rosbags Wiki

[2] Calibrating the Inertial Measurement Unit : Section 7 in http://wiki.ros.org/razor_imu_9dof