Lab 3: IMU Noise Characterization with Allan Variance

Deadline: As listed on Piazza

Goals and Learning Objectives

We would like to understand how to characterize and choose IMU sensors for different robotic applications. As part of the exercise, we will write a device driver for the IMU and use that to collect data and analyze it.

After this lab, a successful student will be able to:

- Write a device driver that communicates with the VectorNav IMU over USB serial
- Identify device parameters (e.g., angle random walk) and sources of error from Allan variance data
- Identify and discuss sources of noise in IMU and magnetometer measurements

Collaboration policy for this lab

- Everyone in the team needs to write their own device driver for the IMU.
- Each team should collect one 5 h data set per team.
- Analysis on the collected dataset should be done individually.

Hardware / Sensors

Vectornav VN-100 IMU

Individual work: hardware setup

Standard Ubuntu is not designed for robotic application, so for sensors with a high sampling rate, it may not behave as we want it to. There may be latency in terms of its response to the sensor. There are a set of files under /etc/udev that dictate how Ubuntu will react to certain sensors. These files instruct Linux what to do when a particular device is plugged in. More here

To make the VN-100 plug & play and to address Ubuntu USB latency issues:

1. Plug in or emulate your IMU and in the terminal: `$ dmesg | tail -20`
You will see your sensor attributes including product id and vendor id. These will be listed in the most recent logs after dmesg once you plug the device in.

2. Use a text editor to see the contents of `/etc/udev/rules.d/50-VN-100.rules`. If the values in your file differ from the values below, change them accordingly and if the file doesn't exist,
`$ sudo nano /etc/udev/rules.d/50-VN-100.rules`
with the contents

```
KERNEL=="ttyUSB[0-9]*", ACTION=="add", ATTRS{idVendor}=="1d6b",
ATTRS{idProduct}=="0002", MODE="0666", GROUP="dialout"
```

4. Ubuntu's default latency of 16 ms cause issues with high-rate sensors. This problem can be solved by adding the file in `/etc/udev/rules.d` called 49-USB-LATENCY.rules with the contents:

```
ACTION=="add", SUBSYSTEM=="usb-serial", DRIVER=="ftdi_sio", ATTR{latency_timer}="1"
```

5. Once the rules have been added, to get udev to recognize the rule, run the following command:

```
sudo udevadm control --reload-rules && sudo service udev restart && sudo udevadm
trigger
```

6. Finally, unplug and replug the VN-100 to have it work with the new rules.

You can verify the applied settings with below command, should report 1 on success. (Please verify each time you collect data to be safe).

```
$ cat /sys/bus/usb-serial/devices/<ttyUSB0  your device path>/latency_timer
```

Individual work: write a device driver for IMU and do stationary noise analysis

1. Open a serial port with 115200 baudrate. Configure your IMU to output data at 40 Hz. This is a non-trivial step and should not be solved by just sleeping your driver to collect at 40 Hz. You will need to use the instructions presented in class to talk to your IMU.

2. Parse $VNYMR string to get accel, gyro, orientation (roll, pitch, yaw) and magnetometer data. Refer to the user_manual for more information. *Do not use pynmea library, it has very bad checksum handling problems.*

3. Use the standard ROS sensor_msgs/Imu and sensor_msgs/MagneticField to publish the data.
   - You need to code the Euler to quaternion conversions by yourself, do not use libraries. Your custom msg file called "imu_msg.msg" needs to contain the following message variables:
     - ROS Header with the name of Header (The frame_id will be "IMU1_Frame". You may use the system time for the Header time stamp but keep the highest precision by using both the secs and nsecs)
     - ROS sensor_msgs/IMU with the name IMU
     - ROS sensor_msgs/MagneticField with the name MagField
     - (optional) A string with any name containing the raw IMU string (this can help if you do not have the IMU and find a mistake in your driver after data collection).

4. Begin a rosbag for the 3 accelerometers, 3 angular rate gyros, 3- axis magnetometers. Collect at least 5 min of data with the *instrument stationary and as far away as possible from your computer or other computers, moving objects, etc.* You do not need to do any magnetometer calibration here.

5. Plot time series & frequency distribution of each axis of data in your report, and figure out the noise characteristics of each of the values (mean and standard deviation) reported by the IMU. Can you describe the distribution it follows? Please convert the quaternions back to Euler angles for your analysis (we cannot visualize in 4D)

Group work: Allan Variance data collection (group work) and analysis (individual work)

1. For one data set per team, collect roughly 5 hours worth of stationary IMU data at a location that is not subject to vibrations (passing trains, sway from wooden buildings, jumping children, etc). Basements are ideal.

Individual work: Allan Variance analysis

The following MathWorks webpage provides code that you can use to analyze your data for Allan variance. We do not need any further analysis for Allan Variance other than that illustrated in this code. https://www.mathworks.com/help/nav/ug/inertial-sensor-noise-analysis-using-allan-variance.html

You need to analyze this data using the Matlab functions and should be able to answer the following questions.

1. What kind of errors and sources of noise are present?
2. How do we model them? Where do we measure them? How do your measurements compare to the performance listed in the VN100 datasheet?

How to Submit Lab_3

1. In your class repo 'EECE5554', create a directory called LAB3

2. Copy the ROS driver package used for this assignment with the src folder under LAB3.

3. Inside LAB3, create a sub-directory called 'analysis'

4. Place your report in pdf format also in the analysis directory. Your report includes analysis for both the stationary data you collected individually, as well as the MATLAB analysis on the data on Allan variance collected as a team.

5. Place any MATLAB / python code you used to generate and plot noise parameters in this directory as well.

6. Your repo structure should look like
   '<Path_to_repo>/EECE5554/LAB3/src/imu_driver/<all ROS files>'
   '<Path_to_repo>/ EECE5554/LAB3/src/analysis/<your analysis files>'
   '<Path_to_repo>/ EECE5554/LAB3/src/analysis/report.pdf'

7. Push your local commits to (remote) gitlab server. You can verify this by visiting gitlab.com and making sure you can see the commit there.

8. Ensure you have submitted the correct files & folders by cloning a **fresh** copy of your repository in some new directory in your /home/user/ folder on linux (such as /home/user/test_ws) & running catkin_make + testing the driver.

We will need your ROS bag files too but if the 5 hour long bag file is too big for you, please only submit the .csv file from that bag file in the Analysis folder.

Checklist For a Correct ROS Package Structre

- ☐ The ROS package name is *imu_driver*
- ☐ The driver is in a folder called *python* with the name *driver.py*
- ☐ The message file name is *imu_msg.msg* with the names of message variables & types described in the section for writing a device driver.
- ☐ You do not have a fixed USB port for the driver & it can take any USB port given by the user at run time.
- ☐ Your topic name is "*imu*"
- ☐ We can run your driver with the command
  roslaunch imu_driver driver.launch port:="/dev/tty"* # Where * can be any port

Note : If time permits, we will provide a structure checker as we did for LAB1 but that is only for verification. We expect you to have a good upper hand on folder naming conventions in a ROS package by now & will not reiterate them.