**Lab 3**

**IMU Noise Characterization with Allan Variance**

**Deadline**

10:00 PM on Monday, October 23, 2023

**Goals**

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.

**Readings**

We have discussed sensor calibration and sensor stochastic characterization. The Vectornav primer on the subject is included so that you can refresh your memory.

**Hardware / Sensors:**Vectornav VN-100 IMU. The user manual for the sensor is provided.

**Hardware Setup:**

***Imu udev rules*** Standard Ubuntu is not designed for roboti application and for fast sensors 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. [*More here*](https://opensource.com/article/18/11/udev)

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

Connect your imu and type the following command in the terminal

*$ dmesg | grep vn100*

(If you do not know what those commands mean, check out Linux tutorial.)

You will see your sensor attributes including product id and vendor id. If these values differ from the values below, change them in your 50-VN-100.rules file accordingly. (if the file doesn’t exist:

*$ sudo nano /etc/udev/rules.d/50-VN-100.rules*)

1. As Sudo create a file under /etc/udev/rules.d called 50-VN-100.rules with the following

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

1. Ubuntu's default latency of 16ms cause issues with high rate sensors, this can be solved by adding the following UDEV rules as 49-USB-LATENCY.rules

ACTION=="add", SUBSYSTEM=="usb-serial", DRIVER=="ftdi\_sio", ATTR{latency\_timer}="1"

1. 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

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.

*$ 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**

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.

Parse $vnymr string, to get accel, gyro, orientation (roll, pitch, yaw) and magnetometer data. Refer to sensor user\_manual. Do not use pynmea library, it has very bad checksum handling problems.

Use the standard ROS [sensor\_msgs/IMU](http://docs.ros.org/en/api/sensor_msgs/html/msg/Imu.html) to publish the above data. Convert the above Yaw, Pitch, Roll data into quaternions and publish it as orientation in the same imu msg. Use ROS [sensor\_msgs/MagneticField](http://docs.ros.org/en/api/sensor_msgs/html/msg/MagneticField.html) to publish magnetometer 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).

Begin collecting a time series data (rosbag) for the 3 accelerometers, 3 angular rate gyros, 3- axis magnetometers. Collect at least 10-15 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.

Plot each time series 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 )

**Allan Variance Data collection (group work) and Analysis (individual work)**

For this part we only need to collect one set of data for each team – collect roughly 5 hours worth of stationary IMU data at a location that is not subject to vibrations (passing trains, building sway etc). Basements are ideal.

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/sources of noise are present?
2. How do we model them? Where do we measure them? Can you relate your measurements to the datasheet for the VN100?

**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 a new directory in your /home/user/ folder on linux (such as /home/user/test\_ws) & running catkin\_make + testing the driver.

**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.