Lab Guide

Sensor Interfacing

Background

Sensor Interfacing guides students through the process of acquiring data and calculating the statistical properties used by the sensor fusion lab. Prior to starting Lab Guide – Sensor Interfacing please read through the following concept documents:

- Concept Review – Sensor Models

Lab Guide – Sensor Interfacing will ask you to complete the following pipelines:

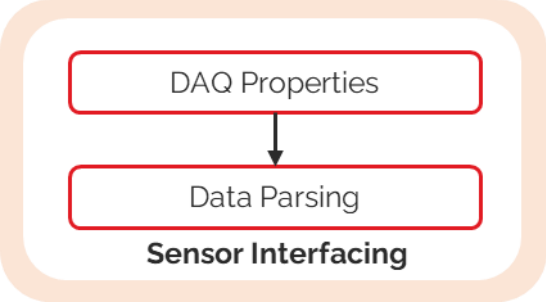

Figure 1: Sensor Interfacing Pipeline

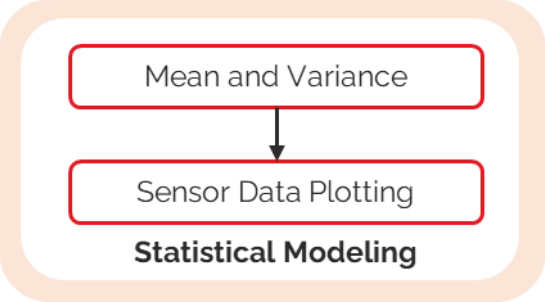

Figure 2: Statistical Modeling Pipeline

- **Sensor Interfacing** is designed give students insight into what a sensor measurement looks like when a QCar is moving around and how to define the read method for accessing data.
- **Statistical Modeling** is designed for students to understand how accurate a sensor measurement is by quantifying the uncertainty in measurement data.

Skills Activity 1 - Sensor Interfacing

Robotic systems make use of Micro Electro-Mechanical Systems (MEMS) to sense information related to the rigid body. Categorized as interoceptive sensors, these include accelerometers, gyroscopes, encoders, voltage sensors, temperature sensors, etc. Autonomous vehicles use these sensors to complement the kinematic properties of the system and estimate motion using sensor fusion techniques.

The objective of this lab is to understand how to visualize sensor data and the type of information it measures within a self-driving car. Sensors being looked at for this skills activity are:

- Accelerometers (x,y,z axis) m/s^2
- Gyroscope (x,y,z axis) rad/s

self.myQCar is the variable used to interact with either the virtual or physical QCar. Subfunctions like **self.myQCar.read()** can be used to specify when to request sensor data from inside of a timing loop. **taskRate** is the variable used to specify the frequency at which data is requested.

Step 1 – Class Initialization

The first step is configuring the number of samples and the rate at which you will request data from the QCar.

- Specify the read rate using **taskRate**
- Specify the number of samples being recorded using **specifiedSamples**

```
# ============  SECTION A - Student Inputs for Image Interpretation ===============
sensorLab = sensorInterfacing(taskRate = 120,
                    specifiedSamples = 600)

''' Students decide the activity they would like to do in the
sensor interfacing lab
List of current activities:
- read (sensor interfacing skills activity)
- sensor_stats (sensor characterization skill activity)
'''
sensorMode = "read"
```

Codeblock A – Student Inputs for sensor interfacing for reading QCar sensors

Note: For the sensor **read** mode the QCar will move in space. The sample rate and number of samples defines the length of time for which the simulation is running for. As an example, if the QCar has a task rate of 120 and samples of 600 the experiment will run for a combines 5 seconds. The QCar will move along a counter-clockwise arc for 2.5s and a clockwise arc for 2.5s.

Step2 – Reading and parsing data

Using the **self.myQCar.read()** and the buffers **self.myQCar.accelerometerData** and **self.myQCar.gyroscopeData** define **self.accelerometerData** and **self.gyroscopeData** to visualize the data in real-time using the preconfigured scope class.

```
    # ============  SECTION B - Sensor Reading and Data Parsing =============
            self.timeStamp = self.elapsedTime(t0)
            '''
```

Codeblock B – Data Reading and Parsing

Physical setup -

Run **sensor_interfacing.py.** 2 python windows will show up with the data visualizing the QCar accelerometer and gyroscope data.

Virtual setup -

Run **sensor_interfacing.py** to start the skills activity code. 2 python windows will show up with the data visualizing the QCar accelerometer and gyroscope data.

Results:

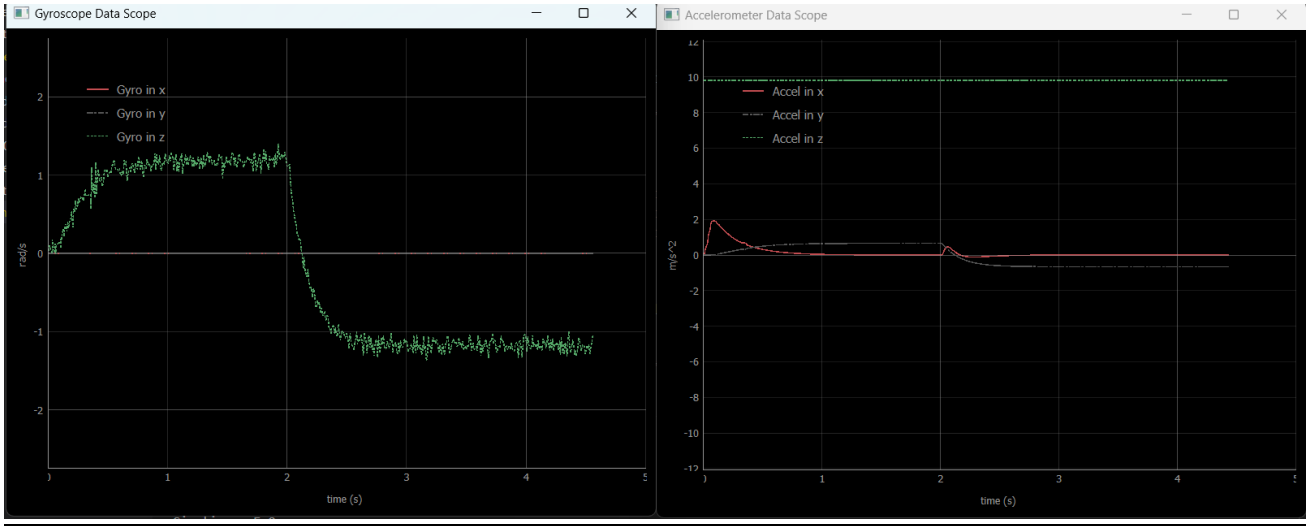

Figure 3: Sample Accelerometer and Gyroscope scope data for virtual QCar.

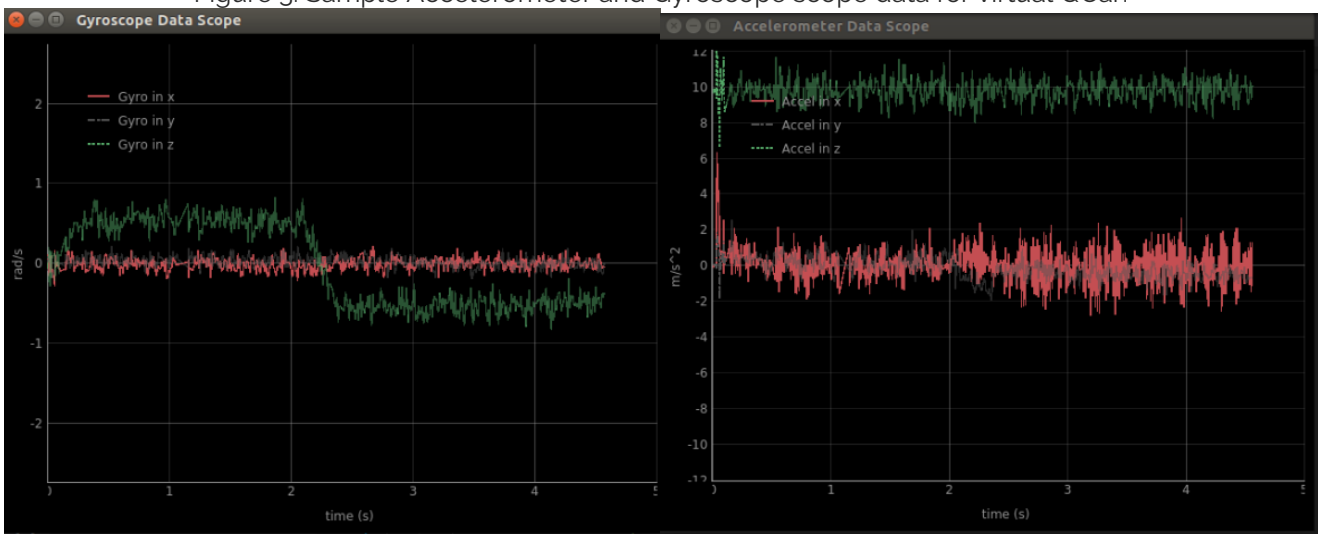

Figure 4: Sample Accelerometer and Gyroscope scope data for physical QCar.

Skills Activity 2 – Statistical Modeling

Sensor information gives access to the dynamic properties of the car. Part of understanding how sensors can be used is to identify the amount of noise present in the sensor measurement. In this skills activity the noise of the sensors on the QCar will be calculated by keeping the vehicle stationary and identifying the average measurement and variance for both the gyroscope and IMU. This information will be used in the Lab Guide – State Estimation.

Step 1 – Class Initialization

The first step is configuring the number of samples and the rate at which you will request data from the QCar.

- Specify the read rate using **taskRate**
- Specify the number of samples being recorded using **specifiedSamples**

```
# ============  SECTION A - Student Inputs for Image Interpretation ===============
sensorLab = sensorInterfacing(taskRate = 120,
                    specifiedSamples = 600)

''' Students decide the activity they would like to do in the
sensor interfacing lab
List of current activities:
- read (sensor interfacing skills activity)
- sensor_stats (sensor characterization skill activity)
'''
sensorMode = "read"
```

Codeblock A – Student Inputs for sensor interfacing for reading QCar sensors

Step2 – Statistical Properties

Using the variable **self.plotData** calculate the mean (**mu**) and variance (**var**) of the samples collected from the gyroscope and accelerometer

```
# ========== SECTION C - Statistical Properties ==========
mu = np.zeros(6)
var = np.zeros(6)

'''
print("Mean: ", mu)
print("Variance: ",var)
```

Codeblock C – calculation for mean (mu) and variance(var) in SECTION C

Step3 – Data Plotting

Using the variable **self.plotData** visualize the overall data collected for this skill activity and visualize the histogram representing the distribution of samples. Provided is an example of using the package matplotlib to visualize a histogram and data over time for the gyroscope z-data.

```
        # ========== SECTION D - Data Plotting  ==========
        # Example plot for gyroscope z values:
        fig1 = plt.figure()
        plt.plot(self.time, self.plotData[:,5])
        plt.title("Gyro z-axis data")
        plt.xlabel('time (s)')
        plt.ylabel(r'$\omega_z$ (rad/s)')
        plt.grid()

        fig2 = plt.figure()
        plt.hist(self.plotData[:,5], bins=9)
        plt.title("Gyro z-axis histogram")
        plt.xlabel(r'$\omega_z$ (rad/s)')
        plt.ylabel('samples')
        plt.grid()
        plt.show()
```

Codeblock D – Data plotting

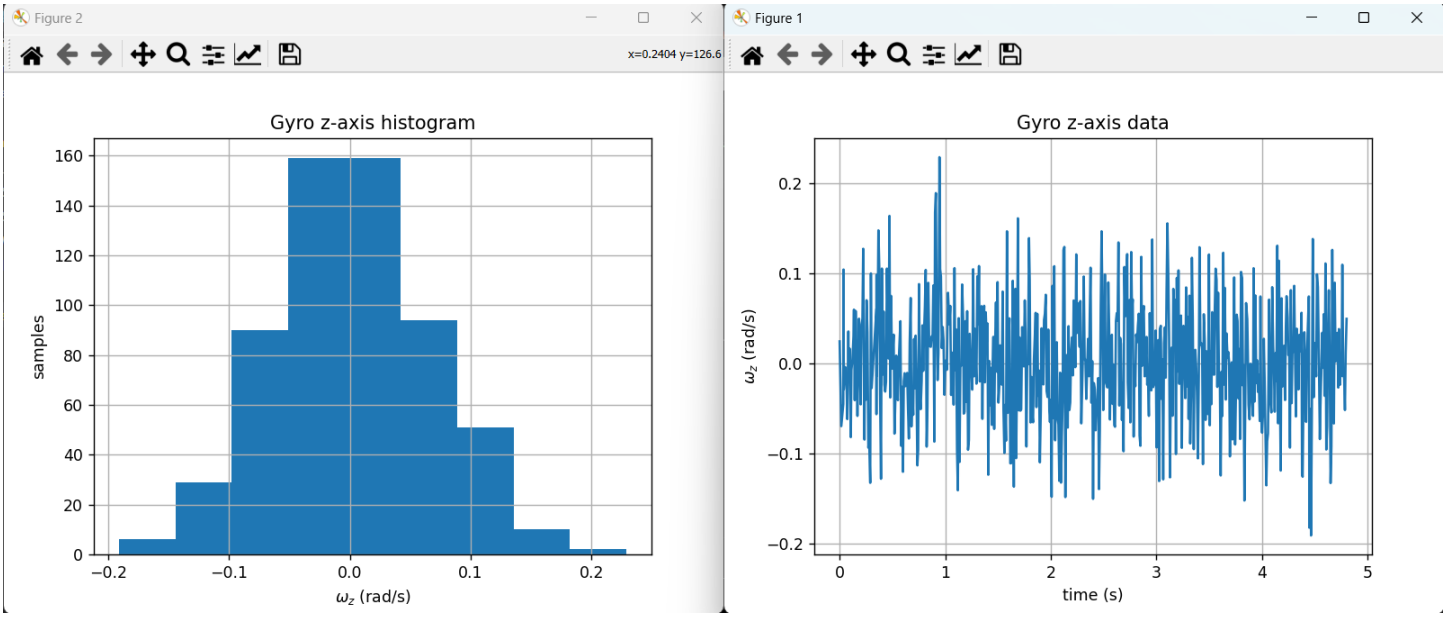

Figure 5: Histogram and sensor plot for gyro-z axis on virtual QCar

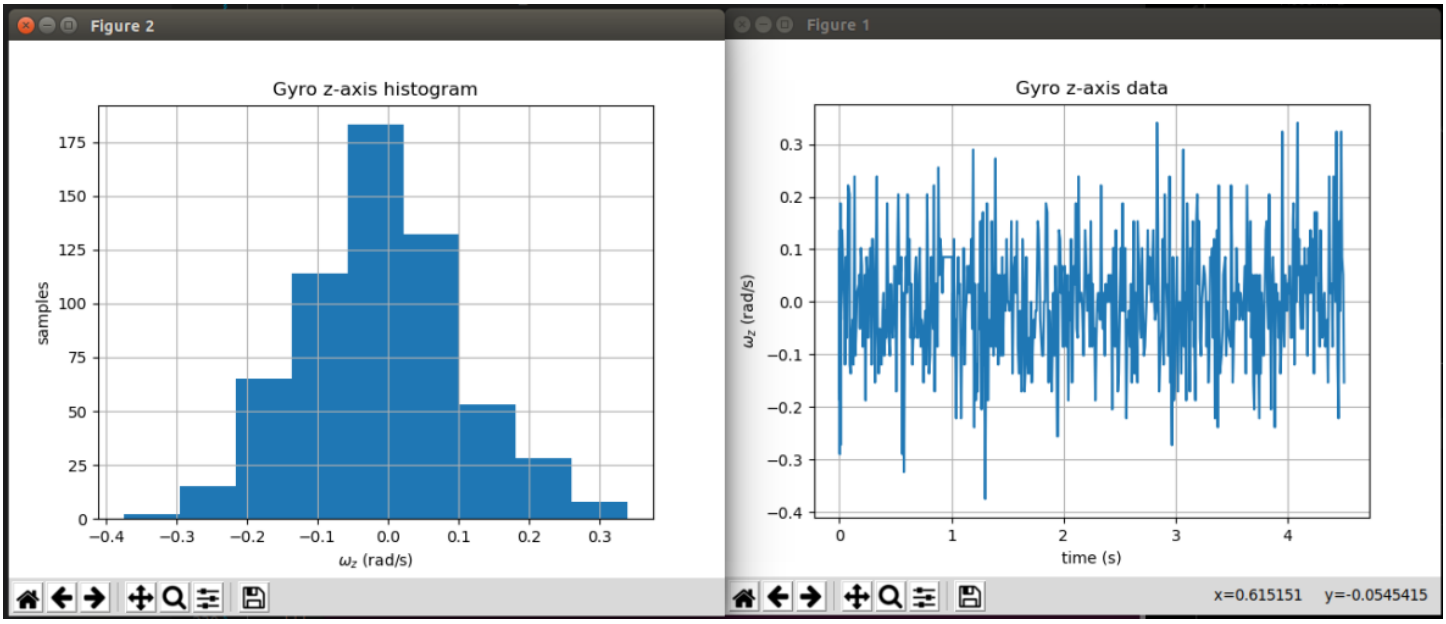

Figure 6: Histogram and sensor plot for gyro-z axis on physical QCar

Physical setup -

Run **sensor_interfacing.py.** 2 python windows will show up with the data visualizing the QCar accelerometer and gyroscope data.

Virtual setup -

Run **sensor_interfacing.py** to start the skills activity code. 2 python windows will show up with the data visualizing the QCar accelerometer and gyroscope data.

Results:

Distributions, data plot, mean and variance values for the physical QCar.

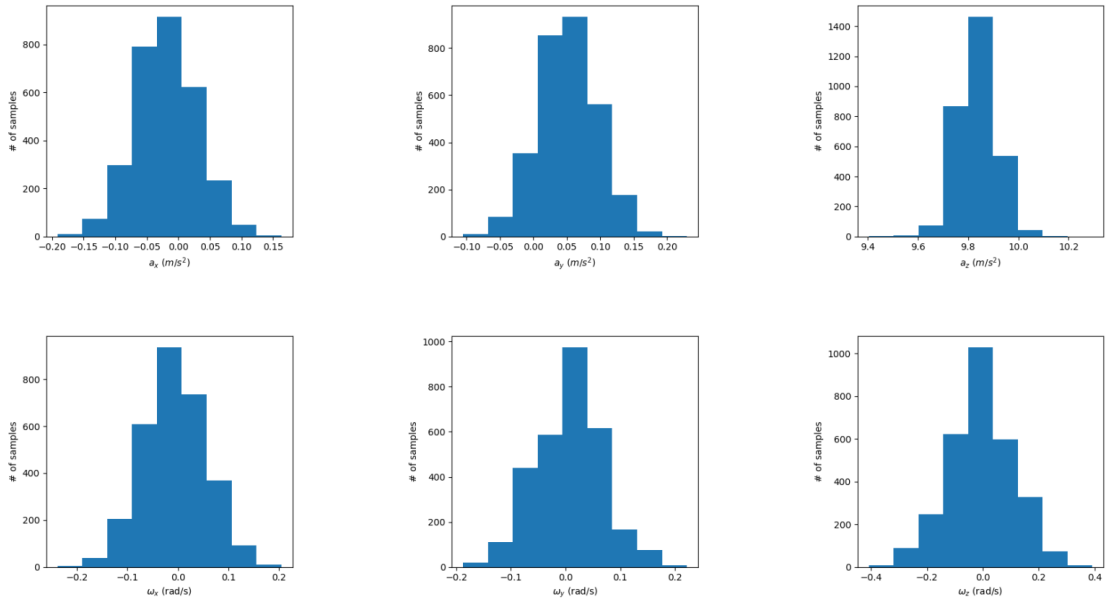

Figure 7: Histograms for Accelerometer and Gyroscope on physical QCar

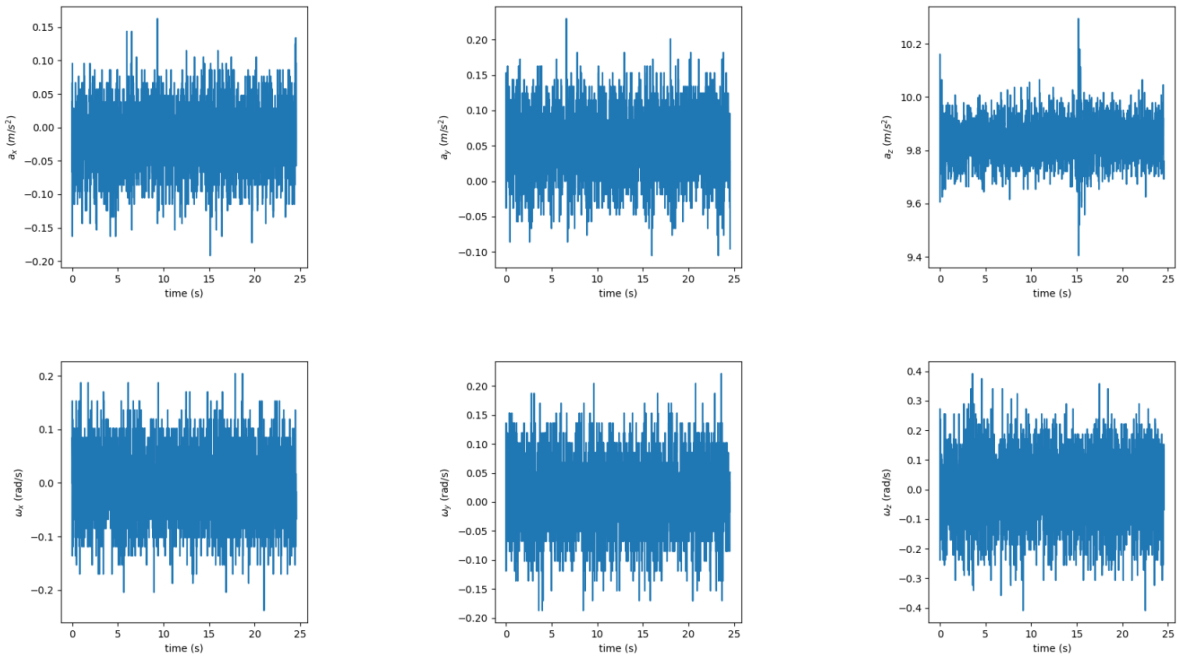

Figure 8: Sensor plots for Accelerometer and Gyroscope on physical QCar

Mean	$[-1.943e-03 \quad 5.1405e-02 \quad 9.8376 \quad -6.2495e-03 \quad 9.2096e-03 \quad -5.9087e-03]$
Variance	$[2.223e-03 \quad 2.07e-03 \quad 5.554e-03 \quad 4.144e-03 \quad 3.6586e-03 \quad 1.378e-02]$

Be mindful Mean and Variance will vary across physical QCars.

Distributions, data plot, mean and variance values for the virtual QCar.

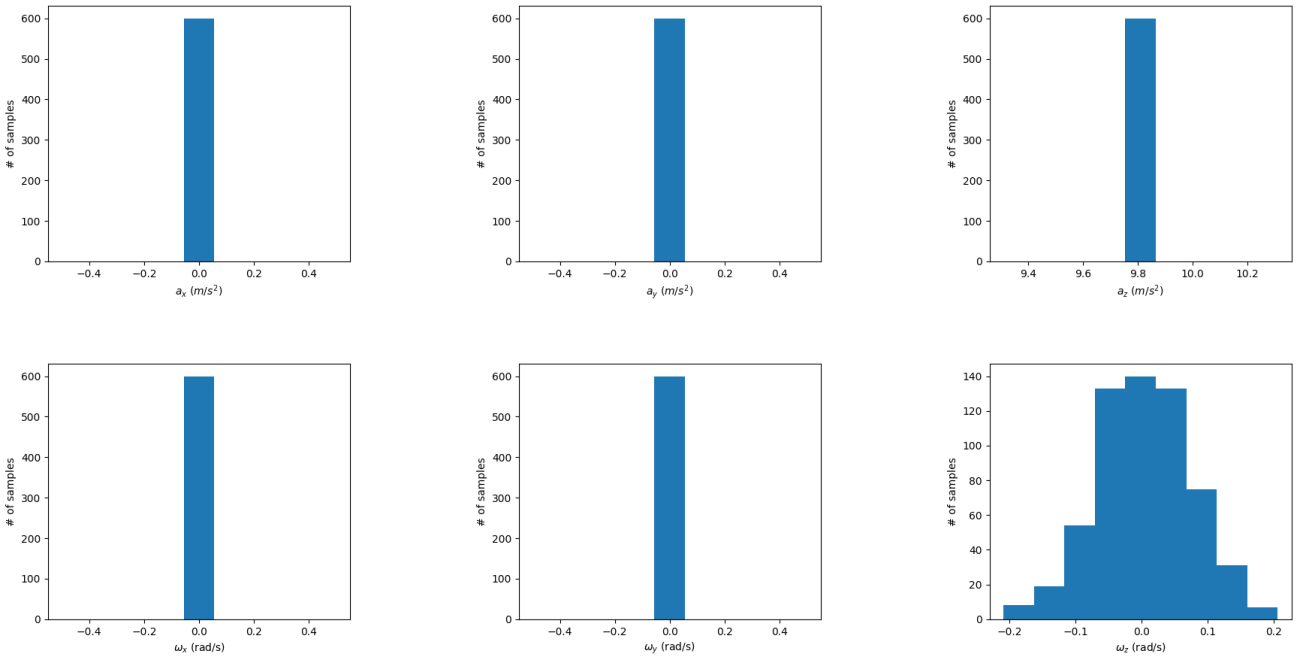

Figure 9: Histograms for Accelerometer and Gyroscope on virtual QCar

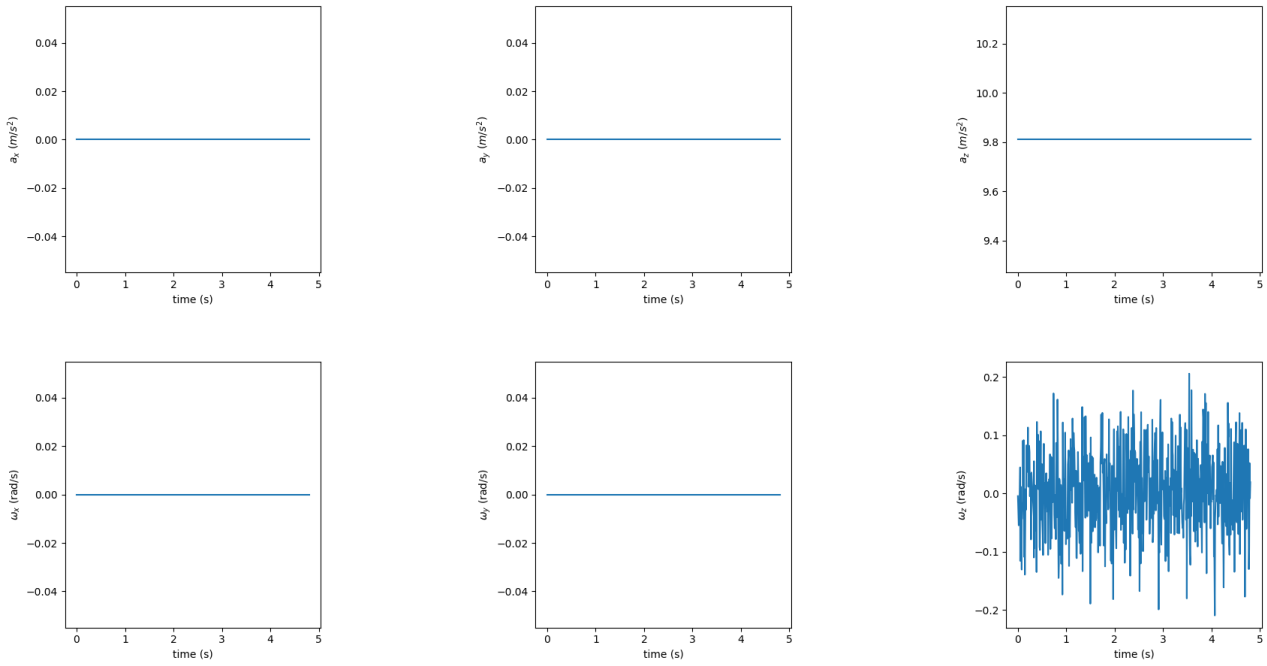

Figure 10: Sensor plots for Accelerometer and Gyroscope on virtual QCar

Mean $[0.0 \quad 0.00 \quad 9.8100 \quad 0.00 \quad 0.00 \quad 3.3024\text{e}-03]$

Variance $[0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 5.1024\text{e}-03]$