DESIGN AND IMPLEMENTATION OF AN IMU CALIBRATION PLATFORM

DESIGN AND IMPLEMENTATION OF AN IMU CALIBRATION PLATFORM

BY

GE CHEN, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING & SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF ENGINEERING

Master of Engineering (2021) McMaster University
(Computing & Software) Hamilton, Ontario, Canada

TITLE:	DESIGN AND IMPLEMENTATION OF AN IMU CALIBRATION PLATFORM
AUTHOR:	Ge Chen M.Eng.
SUPERVISOR:	Dr. Rong Zheng
NUMBER OF PAGES:	vi, 41

Acknowledgements

I would like to thank my supervisor, Professor Rong Zheng, for her patience, guidance, and the continuous support of my study and research. I would also like to thank William Luders and Jiamin Zhou who designed the initial calibration platform. Thanks to Xinzhi Liu, who helped me to trouble shoot the issues in the experiments. Thanks to my wife Yu Wang, thanks for all your support and love !

Contents

List of Figures

Abstract

Inertial Measurements Units (IMU) are widely used in robotics, such as navigation and mapping tasks. Nowadays, many commercial off-the-shelf devices like smartphones and drones are mostly equipped with low-cost embedded IMU sensors. Nevertheless, systematic errors affect low-cost IMUs due to imprecise scaling factors and axes misalignments that decrease the accuracy in position and attitude estimation. Therefore, a procedure to calibrate these IMUs at reasonable costs is essential in many engineering applications. Traditionally the calibration of such IMUs has been done by using special mechanical platforms such as a robotic manipulator. However, such mechanical platforms used for calibration are usually costly. In this report, we propose a method to calibrate IMUs with the help of a low-cost platform. The procedure is based on a multi-position scheme, providing scale and misalignments factors for both the accelerometers and gyroscopes triads, as well as estimating sensor biases. The method only requires a sensor to be attached to the calibration platform. We use an Arduino Due board to control the motor on the platform and set different attitudes for the rotatable shaft. We design a data collection and calibration protocol that exploits an effective parameterless static filter to reliably detect the static intervals in the sensor measurements, where local stability of the gravity's magnitude can be assumed. In the protocol, the accelerometers triad is first calibrated from measurement samples in the static intervals. Next, these results are exploited to calibrate the gyroscopes through a robust numerical integration. The performances of the proposed calibration technique have been evaluated via actual experiments with a commercial high-precision IMU sensor.

Chapter 1

Introduction

IMUs in today's market, which are composed of an accelerometer, a gyroscope, and sometimes a magnetometer in one sensor package, are usually based on the MEMS (micro-electro-mechanical systems) technology. They have been used in navigation systems, attitude estimation, and consumer electronics like smartphone devices[1]. Their low costs make them attractive, but the advantages come at the expense of having high cross-sensitivity, significant offset biases, and drifts. In an ideal IMU, the tri-axial clusters should share the same three orthogonal sensitivity axes, with fixed and known scale factors that convert the digital quantities measured by each sensor to actual physical quantities (acceleration and angular velocity). Unfortunately, low-cost MEMS-based IMUs are usually affected by non-accurate scaling, sensor axis misalignments, cross-axis sensitivities, and non-zero biases. IMU calibration refers to the process of estimating and correcting these quantities.

1.1 Motivation

Many commercial IMUs in the price range from $1000 to $2000 are factory calibrated, and often they are also temperature compensated. Each sensor is shipped with its calibration parameters stored in the firmware or a non-volatile memory, providing accurate measurements off the shelf. Traditionally, calibrating such IMUs has been done by using unique mechanical platforms such as a robotic manipulator. However, the mechanical platforms used for calibration are usually costly, resulting in a calibration cost that often exceeds the cost of the IMU's hardware. On the other hand, low-cost IMUs ($20 to $100) and the IMU sensors employed in current smartphones are usually poorly calibrated[2]. They are affected by systematic errors from imprecise scaling factors and axes misalignments that decrease accuracy in the position and attitudes estimation. The ability to calibrate these low-cost IMUs at a reasonable cost is essential for many engineering applications.

1.2 Related Work

Traditionally, calibrating an IMU requires a unique mechanical platform that moves the IMU with known rotational velocities in a set of precisely controlled orientations[3, 4]. At each orientation, the output of the accelerometers is compared with the precomputed gravity vector, while during the rotations, the output of gyroscopes is compared with the precomputed rotational velocity. However, such mechanical platforms are usually costly.

In [5], a calibration procedure that exploits a marker-based optical tracking system was presented, while in [6], GPS readings are used to calibrate initial biases and misalignments. The accuracy of these methods strongly depends on the accuracy of the employed kinematic reference (i.e., the motion capture system or the GPS). A multi-position method was firstly introduced by the authors[7]: they proposed to calibrate the biases and the scale factor of the accelerometers using the fact that the magnitude of the static acceleration must equal the gravity's magnitude. This technique has been extended in [8] and [9] to correct accelerometer axis misalignment. The error model proposed for gyroscopes is similar to the one used for accelerometers, but the calibration procedure, in this case, requires a single axis turntable to provide a strong rotation rate signal, for high calibration accuracy. Unfortunately, these approaches not only require mechanical equipment, but since the two triads are independently calibrated, the misalignment between them can not be detected. In [10] and [11], two calibration schemes were presented that do not require any external mechanical equipment. Similar to our approach, in [10], the authors calibrate accelerometers by exploiting the high local stability of the magnitude of gravity. Then, gyroscope calibration is done by comparing the gravity vector measured by the calibrated accelerometer with the gravity vector obtained by integrating the angular velocities from the gyroscope. In [11], the authors also exploit the local stability of the magnetic field.

1.3 Goals

We aim to determine calibration parameters for low-cost IMUs with a low-cost calibration platform. We propose a practical and easy-to-implement calibration protocol that only requires collecting IMU data with the simple procedure described in Fig.3.5. William Luders and Jiamin Zhou designed the initial calibration platform.

The platform has 2-axis rotatable shafts, and we can attach IMU sensors on the shafts. The shafts are driven by two motors which are powered by 4 AA batteries. An Arduino board controls the rotation degrees. After an initialization period with no motion, the rotatable shaft of the platform moves the IMU sensor in different positions while the accelerometer and gyroscope can generate a set of readings. We use mobile app to collect sensor data, and use the collected dataset to calibrate the scale and misalignment factors for both the accelerometers and gyroscopes triads.

1.4 Organization of the Report

The rest of the report is organized as follows. Chapter 2 introduced the basic mechanism of a MEMS-based IMU sensor, the problems of the uncalibrated state. Then, the misalignment, non-orthogonality, scaling, and bias errors were described, including mathematical models and cost functions derived from them. In chapter 3, the hardware part of the calibration platform is illustrated at the beginning; then, the calibration algorithm is described, including the theoretical background. A data collection protocol that conforms to the calibration algorithm is also included. Finally, in chapter 4, two experiments that test with real IMU sensors are described in detail; all results are reported and lead to a positive conclusion. Future works are also exposed.

Chapter 2

Sensor Error Model

An IMU sensor can measure the attitude of the body which it is attached to. It consists of clusters of accelerometers and gyroscopes, sometimes also magnetometers[2]. It works by detecting the current rate of acceleration using the accelerometer cluster and detects changes in rotational attributes like pitch, roll, and yaw using the gyroscope cluster. In this report, we only consider accelerometer and gyroscope. The model used is shown in Fig.2.1

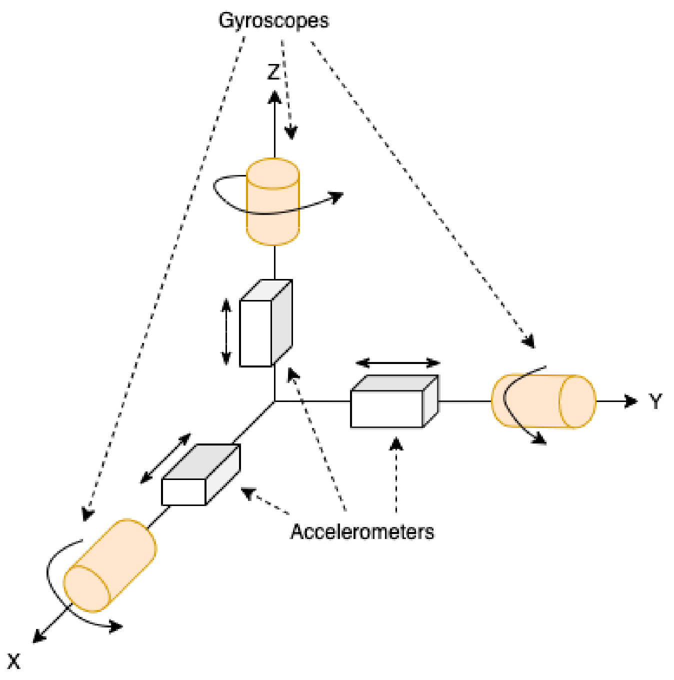


Figure 2.1: A simplified model of an IMU

Sensor errors consist of the non-zero bias, non-unit scale factor, non-orthogonal misalignment of the sensor axes, and the cross-axis sensitivity[7]. A calibration process proceeds by defining sensor error models, deriving the cost functions for accelerometer and gyroscope respectively, collecting raw sensor measurements, and fitting model parameters based on non-linear optimization of the cost functions.

All MEMS-based sensors are subject to deterministic and stochastic errors[12]. I summarized the classification of errors in Table 2.1. Stochastic errors can be identified by Allan Variance[13], and the coefficients of noise sources are obtained by the fitted curve to achieve the modeling process.

Table 2.1: Error Classification in MEMS sensors

Sensor Error Classification							
Deterministic Errors							**Stochastic Errors**
System Drift			**Scale Factor Error**		**Other Error**		
Zero Bias		**Environmental Sensitive Drift**	Nonlinear Error	Temperature Sensitive Error	Cross Coupling Error	Axia Misalignment Error	Quantization Noise, Angle Random Walk, Bias Instability, Rate Random Walk, Rate Ramp, Sinusoidal Noise
Preheat Instability	Equipment Running Instability	Temperture Sensitive Drift					

There are many imperfections due to the sensors themselves, the soldering of the sensors on the chip, and other reasons that cause data distortion so that the sensor's output becomes less accurate. In this work, we only deal with misalignment, non-orthogonality, scaling, and bias errors.

2.1 Misalignment and Non-Orthogonality Errors

For an ideal IMU, the three axes of the accelerometers triad and the gyroscopes triad define a single, shared, and orthogonal 3D frame[2]. Each accelerometer senses the acceleration along a distinct axis, while each gyroscope measures the angular velocity around one of the same axes. Unfortunately, in real IMUs, the two triads may form two misaligned and non-orthogonal frames due to assembly inaccuracy.

Since both the accelerometers frame (AF) and the frame of the gyroscope (GF) are usually non-orthogonal, we can define two associated orthogonal ideal frames (AOF and GOF, respectively) as follows:

- The x-axis of the AOF and the one of the AF coincide
- the y-axis of the AOF lies in the plan spanned by the x and y axes of the AF.

For the gyroscopes, it is sufficient to substitute the AF and AOF with GF and GOF, respectively. Finally, we defined a body frame (BF), which is an orthogonal frame that represents, for example, the coordinate frame of the IMU's chassis. The body frame usually differs from the AF and GF frames by small angles. However in general, there is no direct relation between them. For small angles, measurements S^S in a non-orthogonal frame (AF or GF) can be transformed into the orthogonal body frame as:

$$\mathbf{s}^B = \mathbf{t}\mathbf{s}^S, \quad \mathbf{t} = \begin{bmatrix} 1 & -\beta_{yz} & \beta_{zy} \\ \beta_{xz} & 1 & -\beta zx \\ -\beta_{xy} & \beta_{yx} & 1 \end{bmatrix}, \tag{2.1}$$

where $\mathbf{s}^B$ and $\mathbf{s}^S$ denote the (acceleration), or equivalently the (rotational velocity), in the body frame coordinates and accelerometers (gyroscopes) coordinates, respectively. Here β_{ij} is the rotation of the i-th accelerometer (gyrocsope) axis around the j-th BF axis as illustrated in Fig.2.2.

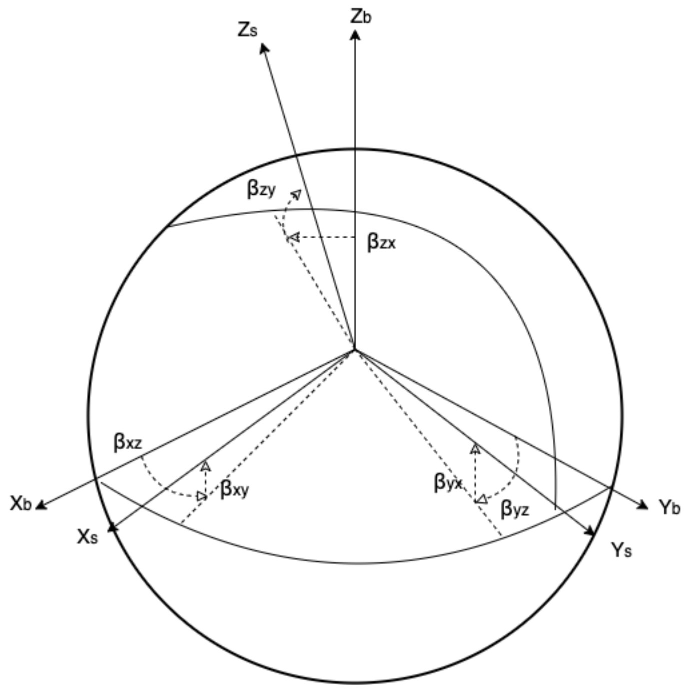


Figure 2.2: Sensor sensitivity axes x^S, y^S, z^S, and body frame coordinates axes x^B, y^B, z^B

On the other hand, the two orthogonal frames, BF and AOF (and, equivalently, BF and GOF), are related by a pure rotation. In our calibration method, we assume that the body frame BF coincides with the accelerometers orthogonal frame AOF. In such a case, the angles β_{xz}, β_{xy}, β_{yx} are zero, and in the case of the accelerometer Eq.(2.1) becomes:

$$\mathbf{a}^O = \mathbf{t}^a \mathbf{a}^S, \quad \mathbf{t}^a = \begin{bmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where letter β, referring to the general case, is changed to the letter α for accelerometer, while $\mathbf{a}^O$ and $\mathbf{a}^S$ denote the specific acceleration in AOF and AF, respectively.

We cannot apply the same simplification to the gyroscope coordinate frame because we want to obtain calibrated measurements coherent between the gyroscopes and the accelerometers[2]. Thus, we have to orthogonalize the gyroscopes' sensitivity coordinate frame and rotate it. For the gyroscopes, we have:

$$\boldsymbol{\omega}^O = \mathbf{t}^g \boldsymbol{\omega}^S, \quad \mathbf{t}^g = \begin{bmatrix} 1 & -\gamma_{yz} & \gamma_{zy} \\ \gamma_{xz} & 1 & -\gamma_{zx} \\ -\gamma_{xy} & \gamma_{yx} & 1 \end{bmatrix}, \tag{2.3}$$

where $\boldsymbol{\omega}^O$ and $\boldsymbol{\omega}^S$ denote the specific angular velocities in the orthogonal coordinate frame and IMU's gyroscopes sensitivity coordinate frame, respectively. $\boldsymbol{t}^g$ is the matrix that orthogonalizes the gyroscopes sensitivity axies and aligns them to these of the accelerometers.

2.2 Scaling Error and Bias

Both the accelerometers and the gyroscopes are affected by biases and scale errors. Two scaling matrices are introduced:

$$\boldsymbol{k}^a = \begin{bmatrix} s_x^a & 0 & 0 \\ 0 & s_y^a & 0 \\ 0 & 0 & s_z^a \end{bmatrix}, \quad \boldsymbol{k}^g = \begin{bmatrix} s_x^g & 0 & 0 \\ 0 & s_y^g & 0 \\ 0 & 0 & s_z^g \end{bmatrix}. \tag{2.4}$$

In the ideal case both $\boldsymbol{k}^a$ and $\boldsymbol{k}^g$ are identity matrix. Also, two bias vectors are introduced:

$$\boldsymbol{b}^a = \begin{bmatrix} b_x^a \\ b_y^a \\ b_z^a \end{bmatrix}, \quad \boldsymbol{b}^g = \begin{bmatrix} b_x^g \\ b_y^g \\ b_z^g \end{bmatrix}. \tag{2.5}$$

2.3 Complete Sensor Error Model

We should also consider the measurement noise to complete the sensor error model. Thus, the complete accelerometer error model is given by,

$$\boldsymbol{a}^O = \boldsymbol{t}^a \boldsymbol{k}^a \big(\boldsymbol{a}^S + \boldsymbol{b}^a + \boldsymbol{\upsilon}^a\big), \tag{2.6}$$

and the complete gyroscope error model is

$$\boldsymbol{\omega}^O = \boldsymbol{t}^g \boldsymbol{k}^g \big(\boldsymbol{\omega}^S + \boldsymbol{b}^g + \boldsymbol{\upsilon}^g\big), \tag{2.7}$$

where $\boldsymbol{\upsilon}^a$ and $\boldsymbol{\upsilon}^g$ are the accelerometer measurement noise and the gyroscope measurement noise, respectively.

2.4 Cost Function

In order to calibrate an accelerometer, nine unknown parameters of the error model presented in Eq.(2.6) need to be estimated. That is

$$\boldsymbol{a}^O = \begin{bmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x^a & 0 & 0 \\ 0 & s_y^a & 0 \\ 0 & 0 & s_z^a \end{bmatrix} \left(\boldsymbol{a}^S + \begin{bmatrix} b_x^a \\ b_y^a \\ b_z^a \end{bmatrix} \right). \tag{2.8}$$

Thus, the vectorized unknown parameters for the accelerometer ($\boldsymbol{\theta}^{acc}$) are

$$\boldsymbol{\theta}^{acc} = \left[\alpha_{yz}, \alpha_{zy}, \alpha_{zx}, s_x^a, s_y^a, s_z^a, b_x^a, b_y^a, b_z^a\right]. \tag{2.9}$$

We can define the function

$$\boldsymbol{a}^O = h(\boldsymbol{a}^S, \boldsymbol{\theta}^{acc}) = \boldsymbol{T}^a \boldsymbol{K}^a (\boldsymbol{a}^S + \boldsymbol{b}^a). \tag{2.10}$$

The cost function for accelerometer is

$$\boldsymbol{L}(\boldsymbol{\theta^{acc}}) = \sum_{k=1}^{N} (\parallel \boldsymbol{g} \parallel^2 - \parallel h(\boldsymbol{a}_k^S, \boldsymbol{\theta}^{acc}) \parallel^2)^2, \tag{2.11}$$

where N is the number of measurement sets from which acceleration vectors $\boldsymbol{a}_k^S$ are extracted, averaging the accelerometer readings in a temporal window inside each static interval. $\parallel \boldsymbol{g} \parallel$ is defined as the actual magnitude of the local gravity vector. In order to minimize Eq.(2.11), the Levenberg-Marquardt (LM) algorithm[14] is used with initial guess $\boldsymbol{\theta}_0^{acc} = [0, 0, 0, 1, 1, 1, 0, 0, 0]$.

We use a new bias-free system for gyroscopes simply by averaging the static gyroscope signals (Allan Variance method, to be discussed in Chapter 3). The nine unknown parameters are estimated in the same way as the accelerometer.

$$\boldsymbol{\omega}^O = \begin{bmatrix} 1 & -\gamma_{yz} & \gamma_{zy} \\ \gamma_{xz} & 1 & -\gamma_{zx} \\ -\gamma_{xy} & \gamma_{yx} & 1 \end{bmatrix} \begin{bmatrix} s_x^g & 0 & 0 \\ 0 & s_y^g & 0 \\ 0 & 0 & s_z^g \end{bmatrix} \left(\boldsymbol{w}^S\right). \tag{2.12}$$

Thus, the unknown parameter vector for the gyroscope ($\boldsymbol{\theta}^{gyro}$) is

$$\boldsymbol{\theta}^{gyro} = \left[\gamma_{yz}, \gamma_{zy}, \gamma_{xz}, \gamma_{zx}, \gamma_{xy}, \gamma_{yx}, s_x^g, s_y^g, s_z^g\right]. \tag{2.13}$$

$\boldsymbol{\Psi}$ can be any algorithm that converts a sequence of $\boldsymbol{\omega}_i^S$, from $i = 0$ to $i = n$, and the initial gravity vector $\boldsymbol{u}_0$, to the gyroscope computed gravity vector $\boldsymbol{u}_g$

$$\boldsymbol{u}_g = \boldsymbol{\Psi}\left[\omega_i^S, \boldsymbol{u}_0\right]. \tag{2.14}$$

We adopt the Runge-Kutta integration algorithm[15] to compute the orientation by integrating the angular velocity ω_i^S. Having all these definitions, we can define the cost function as

$$\boldsymbol{L}(\boldsymbol{\theta}^{gyro}) = \sum_{k=1}^{N} \parallel \boldsymbol{u}_{a,k} - \boldsymbol{u}_{g,k} \parallel^2, \tag{2.15}$$

where N is the number of measurement sets, $\boldsymbol{u}_{a,k}$ and $\boldsymbol{u}_{g,k}$ are the *k-th* acceleration vector measured in the calibrated accelerometer frame and the *k-th* acceleration vector computed using the gyroscope frame respectively. To minimize Eq.(2.15), the same algorithm is used with initial guess $\boldsymbol{\theta}_0^{gyro} = [0, 0, 0, 0, 0, 0, 1, 1, 1]$.

Chapter 3

Calibration Platform Design and Algorithm Implementation

We use two fundamental properties to set up the calibration method [10]. They put physical and mathematical constraints on the sensor outputs. In this way, we can calibrate an IMU easily without using high-precision mechanical equipment.

- **Property-1:** the magnitude of the acceleration measured in static states must equal the gravity.
- **Property-2:** the gravity vector measured using a triaxial accelerometer in static state must equal the gravity vector computed using the IMU orientation integration algorithm, which uses the angular velocities measured by the gyroscopes, and it starts the orientation integration from a direction given by the triaxial accelerometer itself.

Before we dive into the calibration procedure, let's first described the design of the calibration platform.

3.1 Hardware

William Luders and Jiamin Zhou designed the initial calibration platform. The mechanical design of the platform can be found in Fig. 3.1. They use a breadboard to connect the wires to the Arduino board. The disadvantage is that the wires may get loose and lose connection when the platform is moved. It is inconvenient when the tests are conducted in different places due to the need to reconnect the wires each time. To eliminate the need for reconnection, I design a circuit board and use sockets to fix the wires and make the connection more stable.

The calibration platform has two 2-axis rotatable shafts, allowing to place IMU sensors at different attitudes for raw sensor data collection. We require repeatable motions across trials with some accuracy ($\pm 10^{\circ}$ range) so that the sensors would be subject to similar conditions during calibration. In fact, we do not need knowledge of the motion from the calibration platform.

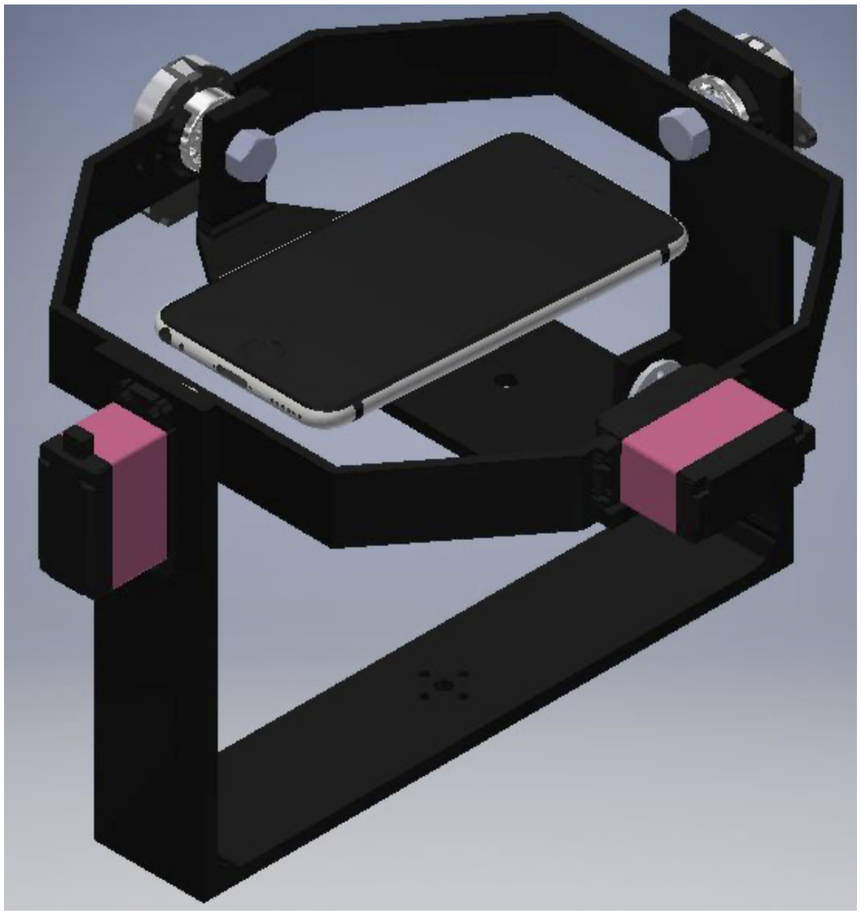


Figure 3.1: Calibration platform CAD design

- **Motors**

  We choose standard size servos with 270° range of motion. They have good repeatability and controllability, and are easy to integrate with an Arduino Due microcontroller. We use 4 AA batteries to power these motors.

- **Controller**

  We select an Arduino Due as the microcontroller. It has enough hardware timers and interrupt pins to run the servos. We can set multiple positions in a position table and let the controller control the rotations of the 2-axis shafts on the platform to collect sensor data in all different attitudes.

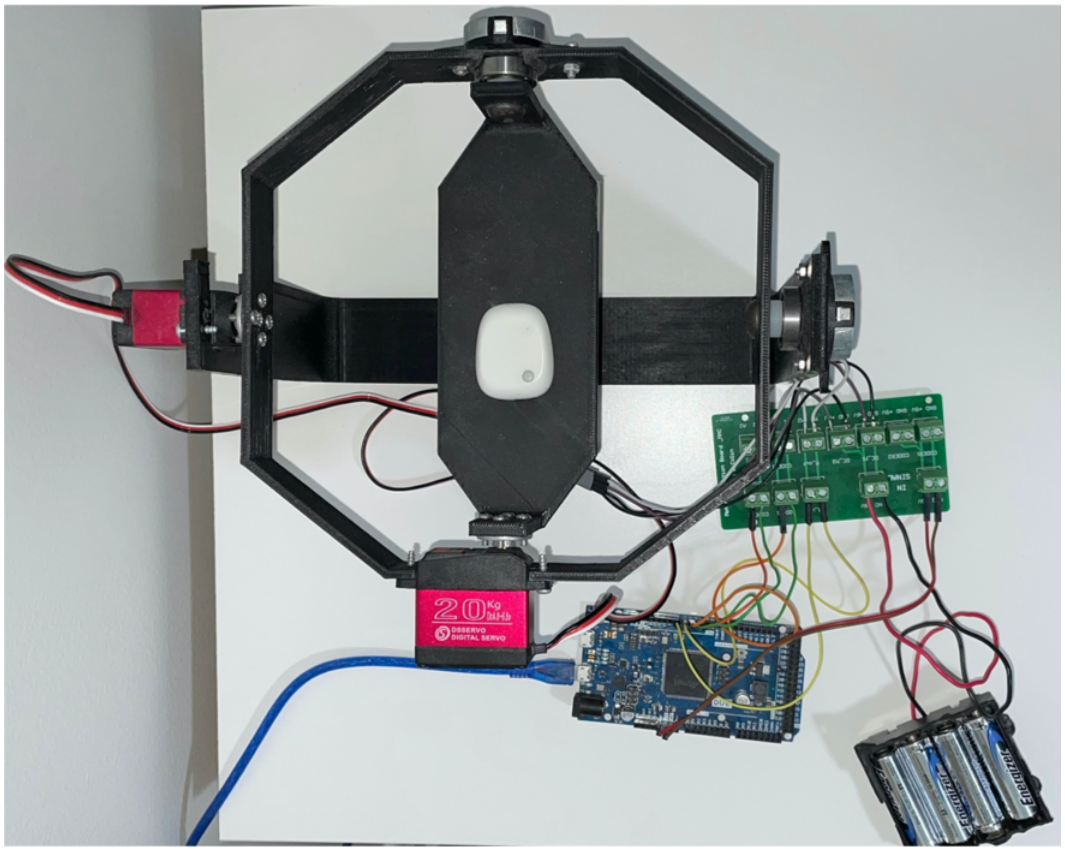


Figure 3.2: Calibration platform material object

3.2 Calibration Algorithm

The calibration algorithm requires four inputs, namely, T_{init}, t_{wait}, $\boldsymbol{a}^S$ and $\boldsymbol{\omega}^S$. T_{init} is the time that IMU sensors need to be placed in static at the beginning of a calibration. t_{wait} is the time that the rotatable shafts keep static before they start rotating again. $\boldsymbol{a}^S$ and $\boldsymbol{\omega}^S$ are raw sensor readings of a accelerometer and a gyroscope, respectively. T_{init} can be computed by Allan Variance method(more on 3.2.1). Our current implementation sets t_{wait} to 4 seconds. The outputs are the calibration parameters discussed in section 2.4. The calibration algorithm implements a motion detector(more on 3.2.2) that can identify static and motion intervals. To calibrate a accelerometer(gyroscope), raw sensor readings from static(motion) intervals are used. Using Ceres Solver library[16], cost functions discussed in 2.4 can be built and get minimized. A high level overview of the calibration algorithm can be described as:

Algorithm 1: IMU Calibration Algorithm

```
Input: $T_{init}$, $t_{wait}$; $\boldsymbol{a}^S$ and $\boldsymbol{\omega}^S$.
Output: Calibration parameters.

$\boldsymbol{b}^g \leftarrow$ average gyroscope signals over $T_{init}$;
$\boldsymbol{\omega}^S_{biasfree} \leftarrow \boldsymbol{\omega}^S$ - $\boldsymbol{b}^g$;
$M_{inf} \leftarrow$ empty matrix;
$\zeta_{init} \leftarrow \sqrt{[var_{tw}(a^t_x)]^2 + [var_{tw}(a^t_y)]^2 + [var_{tw}(a^t_z)]^2}$, with $t_w = T_{init}$ ;
for $i = 1 : k$ do
    $threashold \leftarrow i * \zeta^2_{init}$;
    $static\_intervals \leftarrow$ motion detector computed using $t_{wait}$ and $threshold$;
    $[Residual, Params] \leftarrow$ optimize Eq.(2.11), using $static\_intervals$ and $\boldsymbol{a}^S$
     averaging with $t_{wait}$ ;
    $M_{inf}(i) \leftarrow [Residual, Params, threshold, static\_intervals]$;
$index_{opt} \leftarrow$ index of the minimum residual in $M_{inf}$;
$Params_{acc} \leftarrow$ from $M_{inf}$ using $index_{opt}$ ;
$static\_intervals_{opt} \leftarrow$ from $M_{inf}$ using $index_{opt}$ ;
$\boldsymbol{a}^O \leftarrow$ calibrate $\boldsymbol{a}^S$ using $Params_{acc}$;
$Parameters_{gyro} \leftarrow$ optimize Eq.(2.15), using $static\_intervals_{opt}$, $\boldsymbol{\omega}^S_{biasfree}$ and
 $\boldsymbol{a}^O$ averaging with $t_{wait}$.
```

3.2.1 Allan Variance

Since the algorithm requires T_{init} as one of the inputs, it needs to be computed first. The random gyroscope bias drifts are characterized by using the Allan Variance method[13], which measures the variance of the difference between consecutive interval averages. The Allan Variance σ_a^2 is defined as:

$$\sigma_\alpha^2 = \frac{1}{2}\Big\langle\big(x(\widetilde{t},k) - x(\widetilde{t},k-1)\big)^2\Big\rangle = \frac{1}{2K}\sum_{k=1}^{K}(x(\widetilde{t},k) - x(\widetilde{t},k-1))^2, \tag{3.1}$$

where $x(\widetilde{t},k)$ is the average of the *k-th* interval of $\widetilde{t}$ seconds, and K is the number of segmented intervals. We compute the Allan Variance for each gyroscope axis, with $t_0 \leq \widetilde{t} \leq t_n$. We fix $t_0 = 1s$, $t_n = 225s$. The time in which the Allan Variance of the three axises converge to a small value represents a good choice for the initialization period T_{init}. In this T_{init} period, we compute the average of the static gyroscope signals to correctly determine the gyroscope biases used in the calibration. Fig. 3.3 shows that a good value for T_{init} is about 50 seconds.

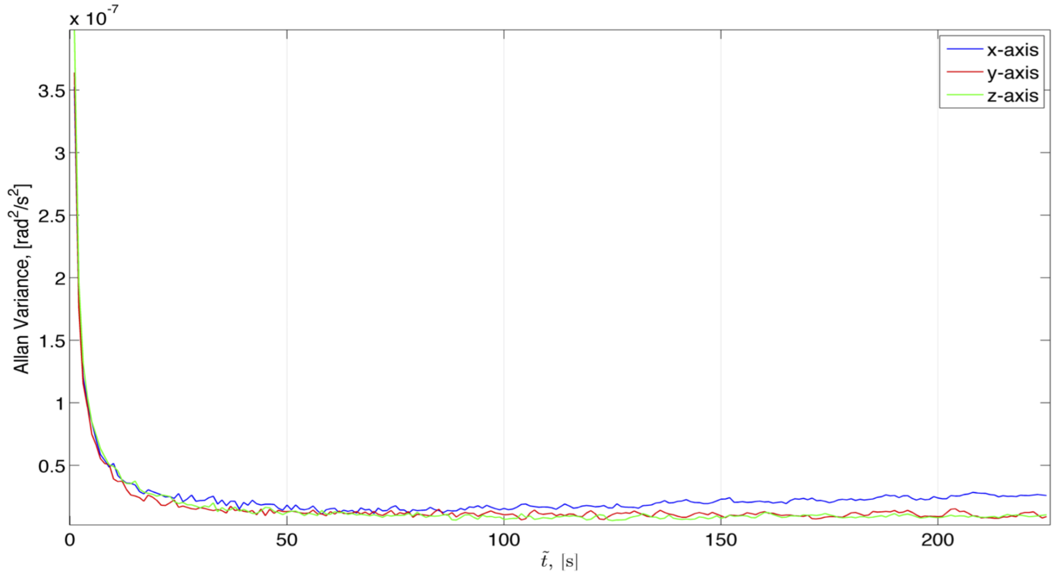


Figure 3.3: Allan Variance computed for the gyroscopes triad

3.2.2 Motion Detector

The calibration accuracy strongly depends on correct identification of static and motion intervals. Static intervals are used to calibrate the accelerometer, while the motion intervals between two consecutive static intervals are used for gyroscopes calibration. The design of the motion detector is based on the accelerometer's readings: given a time interval of length t_{wait} seconds, for time t, the variance magnitude is computed as:

$$\zeta_{init} = \sqrt{[var_{tw}(a_x^t)]^2 + [var_{tw}(a_y^t)]^2 + [var_{tw}(a_z^t)]^2}, \tag{3.2}$$

where $var_{tw}(\boldsymbol{a}^t)$ is the variance of a general signal $\boldsymbol{a}^t$ in a time interval of length t_w seconds centered in t. First, compute the variance of the data from the initial static interval(T_{init}), and its norm, denoted by $norm\_th$ which can be used as a threshold.

Next, use a sliding window of size 101 sample to find static intervals. If the variance of the accelerometers data in the window is less than *norm_th*, it is considered a static interval. Fig3.4 shows the plots of the raw data. The orange border of a rectangle represents a static interval and a motion interval lies in between two consecutive static intervals.

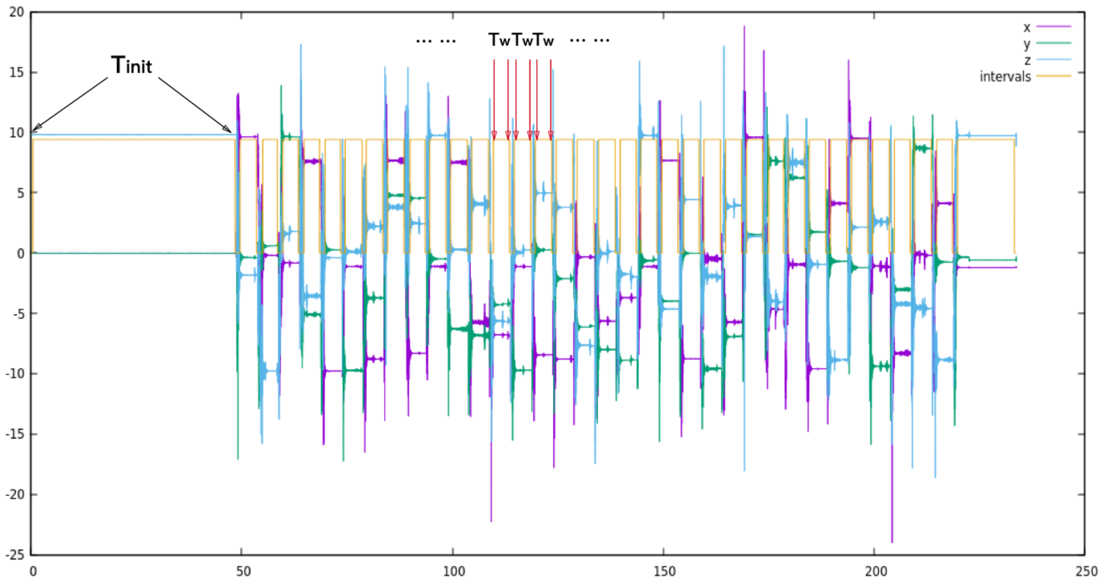


Figure 3.4: Accelerometer static interval detector

3.2.3 Calibration Procedure

We need to put sensors to at least nine different attitudes to correctly estimate the calibration parameters[9]. The higher number N of distinct attitudes we have, the better calibration results we may get. With $36 \leq N \leq 50$ and $1s \leq t_{wait} \leq 4s$[2], a good trade-off can be realized between calibration accuracy, biases stability and noise reduction. The calibration procedure is summarized in flow chart Fig.3.5.

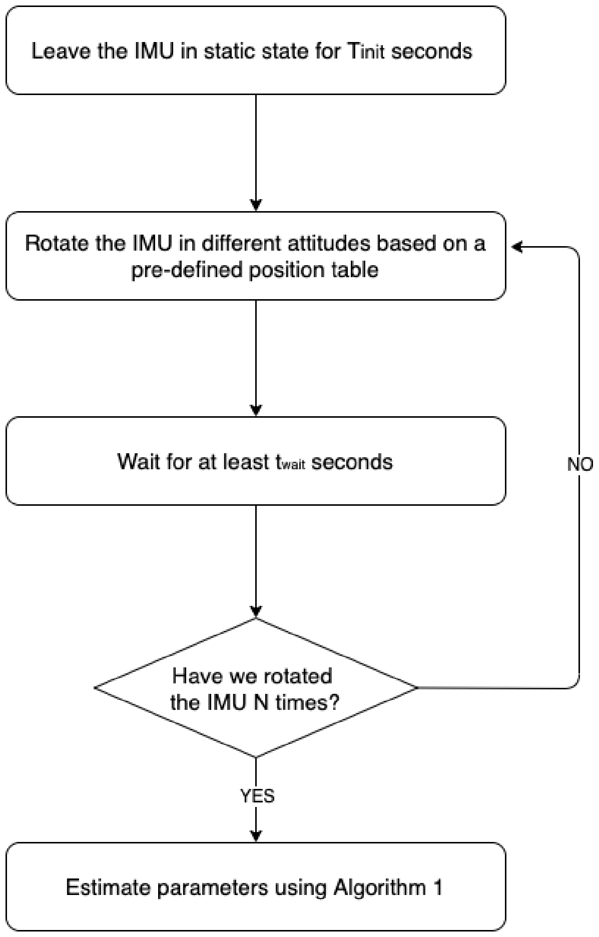

Leave the IMU in static state for Tinit seconds
Rotate the IMU in different attitudes based on a pre-defined position table
Wait for at least twait seconds
Have we rotated the IMU N times?
NO
YES
Estimate parameters using Algorithm 1

Figure 3.5: Calibration procedure

Steps in detail are listed below:

1. Set $t_{wait} = 4s$, $T_{init} = 50s$, the bias vector and scale vector as $\Big[0, 0, 0\Big]^T$, $\Big[1, 1, 1\Big]^T$, respectively.

2. Extract data from the first T_{init} time interval.

3. Compute the variance of the data from the initial static interval (step 2), and its norm, denoted by $norm\_th$.

4. Use a sliding window of 101 sample to find static intervals. If the variance of the accelerometers data in the window is less than *norm_th*, it is considered a static interval. Next, extract the accelerometers data from the static intervals.

5. We use the actual gravity magnitude and the static interval data samples to calculate the cost function defined in Eq.(2.11). If the IMU is in a static state, the accelerometer's reading should equal the gravity magnitude.

6. With the help of Google Ceres Solver library[16], we can minimize the cost function and obtain the nine unknown parameters in the error model.

7. From the calibrated results of the accelerometer, we extract the static samples for gyroscopes.

8. Compute the gyroscopes biases in the static initialization interval (T_{init}).

9. Remove the biases from raw gyroscopes data samples.

10. Find the start and end indices for the motion intervals using the static interval we extracted previously.

11. Build the cost function and minimize it to get the gyroscope calibration parameters.

3.3 Data Collection Protocol

To collect necessary data for calibration, we follow the calibration protocol defined in Fig 3.5.

1. Place the target sensor on the inner shaft of the platform.

2. Run the Ardunio program to drive the rotatable shafts and start collecting raw sensor data. Since we need an initial static interval (T_{init}) to compute the gyroscope's bias and the variance of the accelerometers' readings, the shafts will keep static for $T_{init} = 50s$.

3. After the first T_{init} static interval, the Ardunio program controls the motors to rotate the shafts. The rotation follows the position table defined in the Ardunio program. After each rotation, the shafts remains static for $T_{wait} = 4s$, then rotate again. We can collect a series of static and motion intervals data.

4. When we rotated the IMU $N = 36$ times, stop collecting data.

Fig.3.6 Shows the examples of the platform when collecting raw sensor data.

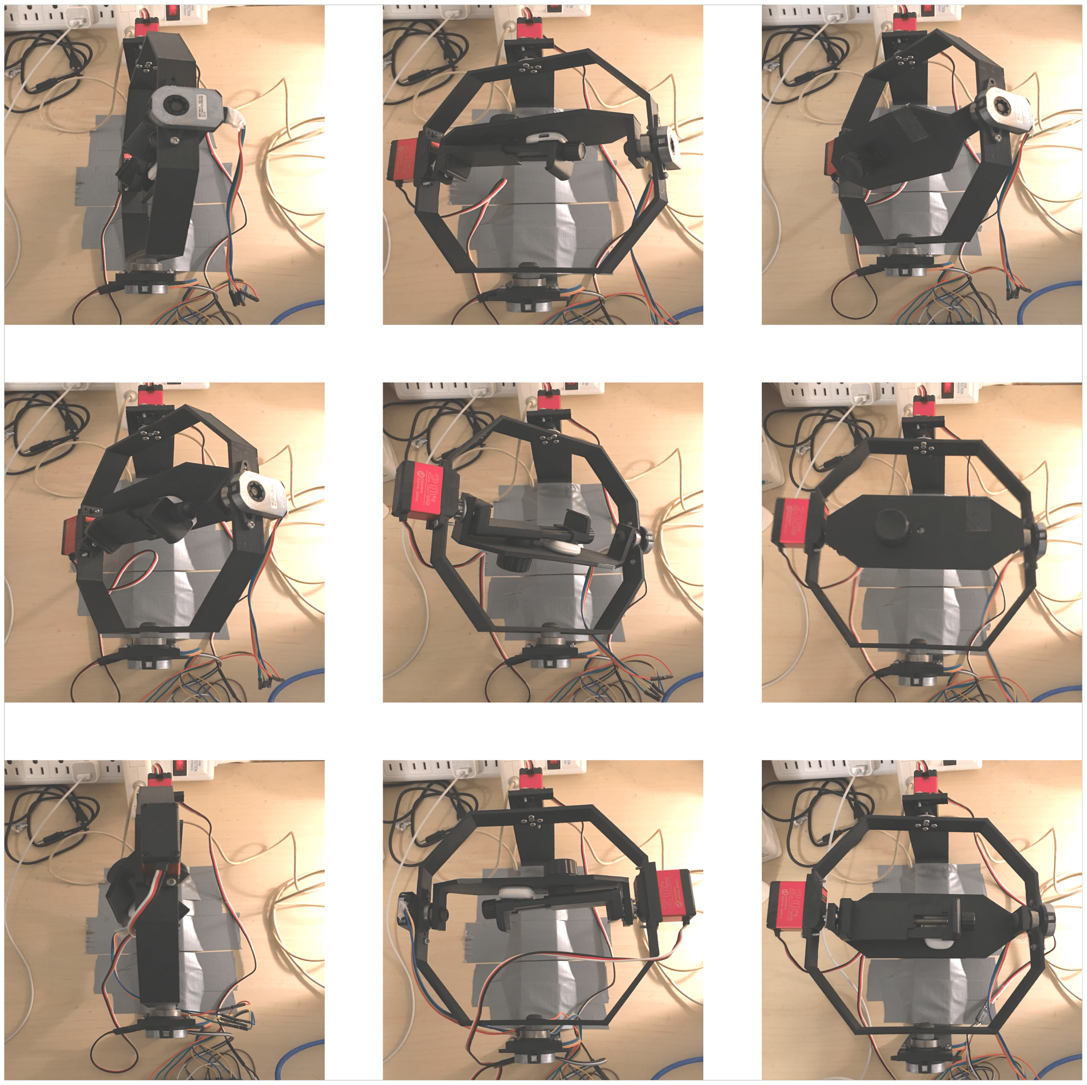


Figure 3.6: Examples of the mbient IMU sensor attached to the platform, placed in different attitudes

Some important aspects worth noting are:

- In the first T_{init} period, one needs to make sure the platform is completely static for a better estimation of the gyroscope's bias.
- Since we use a sliding window with size 101 to detect static intervals, each static

interval should at least have 100 frames data. If the sampling rate is $100Hz$, T_{wait} should be at least 1 second.

- The motion interval cannot be too short. The current implementation of calibration algorithm requires it to be at least 1 second to work. It is better to have a noticeable accelerate or decelerate motion, otherwise the integration may be hard to converge.
- The number of static intervals should be greater than 12. Otherwise, we do not have enough data to perform calibration. 30 to 40 static intervals can lead to optimized results[2].

Chapter 4

Evaluation And Conclusions

With the help of Ceres Solver library[16], we can minimize the cost function and compute the misalignment matrix, the scale matrix, and the bias vector for both the accelerometer and gyroscope. Using Eq.(2.8) and Eq.(2.12) we can get the calibrated sensor readings. For evaluation, use an iMeasureU high-precision IMU sensor[17] to provide the ground truth, and the IMU sensors that being tested are MbientLab MetaMotionR IMU[18] and the built-in IMU of Samsung Galaxy 9 smartphone.

There are two approaches to evaluate the calibration results:

Approach 1: Attach the iMeasureU IMU sensor and the IMU sensor being tested to the same rotatable shaft. Collect raw sensor data for both sensors simultaneously. Run the calibration program to get the calibration parameters. We first compare the similarity between the raw data of the tested sensor and the ground truth. Next, we apply the calibration parameters to get calibrated results and calculate the similarity measure again. If the degree of similarity improves after the calibration, we can conclude that the calibration algorithm and the platform are effective.

Approach 2: Some IMU sensors (e.g., the Xsens MTi IMU) provide factory-calibrated calibration parameters in their datasheets. It is sufficient to compare the computed calibration parameters with the factory-calibrated parameters.

Since all the IMU sensors tested do not provide such factory-calibrated parameters, we adopt approach 1 on the evaluation test. Before we dive into the test, we need to first handle two problems, namely, defining similarity measures and aligning time series data from multiple sensors.

4.1 Data Similarity Measures

To assess how close the data from the IMU sensor being tested to the ground truth before and after the calibration, we choose the Pearson Correlation Coefficient[19] to calculate the data similarity. Unlike the Euclidean Distance similarity score, PCC measures how correlated two data sets is in the range from -1 to 1. A score of 1 indicates that the data sets are perfectly correlated, a score of -1 means that the data sets are negatively correlated, and a score of 0 indicates that the two data sets are not correlated at all.

The Pearson Correlation Coefficient is calculated as the ratio between the covariance and the standard deviation of both data sets. In mathematical form, the score is:

$$Pearson(x, y) = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n} (y_i - \overline{y})^2}}, \tag{4.1}$$

where n is sample size, x_i, y_i are the individual sample points indexed with i, $\overline{x}$ is the sample mean (same for $\overline{y}$).

The Pearson correlation is implemented in multiple Python packages, including Numpy, Scipy, and Pandas, making it much easier to use.

4.2 Aligning Time Series Data

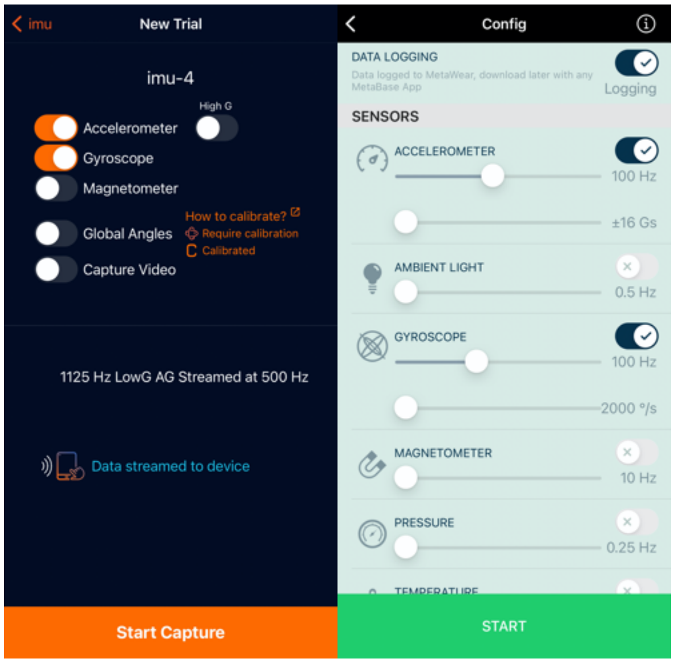
To compute the Pearson Correlation Coefficient between the ground truth and an IMU sensor being tested, we first need to align the time-series data from the two sensors. Even though the data are collected simultaneously, due to different clock crystals used in various sensors and the inability to start recording data for two sensors at the same time, two time-series data can have some delay offsets.

Computing the cross-correlation function is useful for finding the time-delay offset between two time-series data. Python has the numpy.correlate function, but a FFT-based implementation which has much lower complexity with an $O(nlogn)$ running time complexity (compared to $O(n^2)$ running time of the naive implementation) is a better choice.[20][21]

The cross-correlation function has two input parameters X and Y, which are the two sensors readings. It gives us an integer output denoted as t_{shift}. If t_{shift} is less than zero, indicates that Y is t_{shift} frames ahead of X, and if t_{shift} is greater than zero, shows Y is t_{shift} frames behind of X. Shifting one of the sensor data based on t_{shift} aligned two time-series data sets.

4.3 Calibrating a MetaMotionR IMU

Both the iMeasureU and MetaMotionR have a corresponding mobile app that can configure data recording settings (Fig.4.1). We collect raw data for MetaMotionR at a 100Hz sampling rate. The sensitivity for the accelerometer is $\pm 16g$, and the sensitivity for the gyroscope is $2000°/s$. Since iMeasureU streams measurements at 500Hz, the data needs to be downsampled to 100Hz afterward.



imu-4

High G

Accelerometer

Gyroscope

Magnetometer

How to calibrate?

Global Angles

Require calibration

Calibrated

Capture Video

1125 Hz LowG AG Streamed at 500 Hz

Data streamed to device

DATA LOGGING

Data logged to MetaWear, download later with any MetaBase App

Logging

SENSORS

ACCELEROMETER 100 Hz

±16 Gs

AMBIENT LIGHT 0.5 Hz

GYROSCOPE 100 Hz

2000 °/s

MAGNETOMETER 10 Hz

PRESSURE 0.25 Hz

TEMPERATURE

Figure 4.1: iMeasureU and MetaMotionR data recording setting

Before attaching IMU sensors to the platform, we need to know the three axises direction for the iMeasureU and the MetaMotionR sensors. The direction of the x, y, and z-axis for two IMU sensors should be the same during data collection. This information can be retrieved from their datasheets. (Fig.4.2)

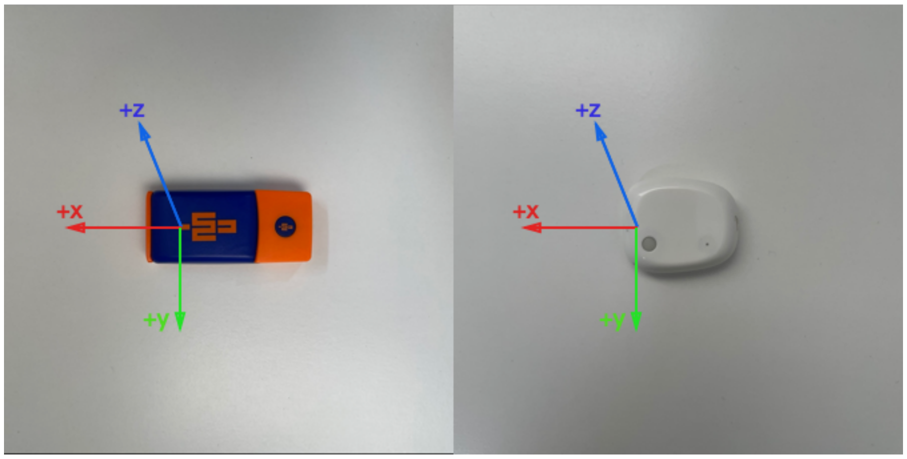
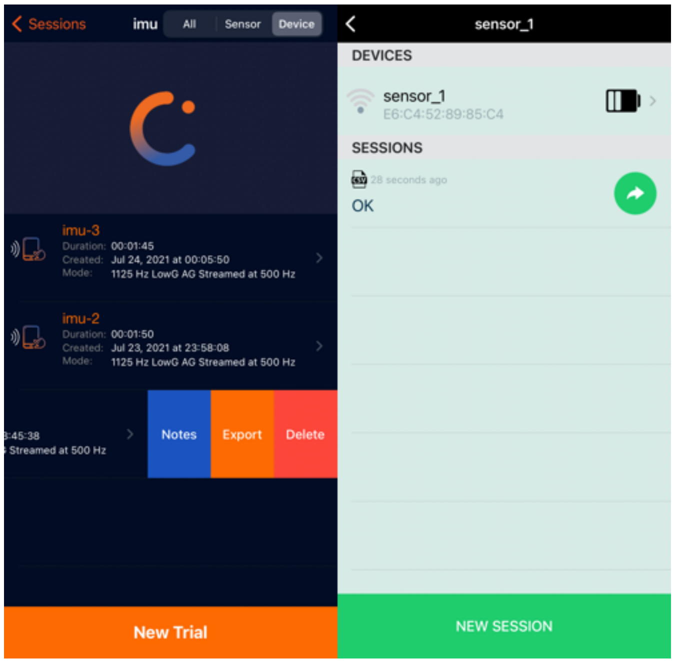


Figure 4.2: X,Y,Z axis direction for iMeasureU (left) and MetaMotionR (right) sensors

Following the data collection protocol in Chapter 3.3, we attach the two sensors to the rotatable shaft and start collecting raw sensor data. The collected data is stored on a mobile phone and can export to a computer for further processing. (Fig.4.3)



imu-3
Duration: 00:01:45
Created: Jul 24, 2021 at 00:05:50
Mode: 1125 Hz LowG AG Streamed at 500 Hz

imu-2
Duration: 00:01:50
Created: Jul 23, 2021 at 23:58:08
Mode: 1125 Hz LowG AG Streamed at 500 Hz

:45:38
Streamed at 500 Hz

sensor_1
E6:C4:52:89:85:C4

28 seconds ago
OK

Figure 4.3: Data export options on mobile app

The calibration program processes the raw data and calculates the calibration parameters of a MetaMotionR sensor as follows:

$$t_{acc} = \begin{bmatrix} 1 & 0 & -0.01 \\ 0 & 1 & -0.003 \\ 0 & 0 & 1 \end{bmatrix}, k_{acc} = \begin{bmatrix} 0.997 & 0 & 0 \\ 0 & 1.005 & 0 \\ 0 & 0 & 0.993 \end{bmatrix}, b_{acc} = \begin{bmatrix} -0.07 \\ -0.655 \\ 0.06 \end{bmatrix},$$

$$t_{gyro} = \begin{bmatrix} 1 & -0.02 & -0.01 \\ 0 & 1 & -0.001 \\ 0.01 & 0 & 1 \end{bmatrix}, k_{gyro} = \begin{bmatrix} 0.981 & 0 & 0 \\ 0 & 1.02 & 0 \\ 0 & 0 & 1.00 \end{bmatrix}, b_{gyro} = \begin{bmatrix} 0 \\ -0.004 \\ -0.005 \end{bmatrix}.$$

Given a raw sensor reading x (e.g., the acceleration), the calibrated reading x' is obtained by $x' = t * k * (x + b)$ where t is the calculated misalignment matrix, k is the scale matrix, and b is the bias vector.

These two time-series data need to be synchronized to measure their similarity. After calculating the time delay offsets between them, we plot in Fig.4.4 the data for x, y, and z-axis and can see the data is now aligned.

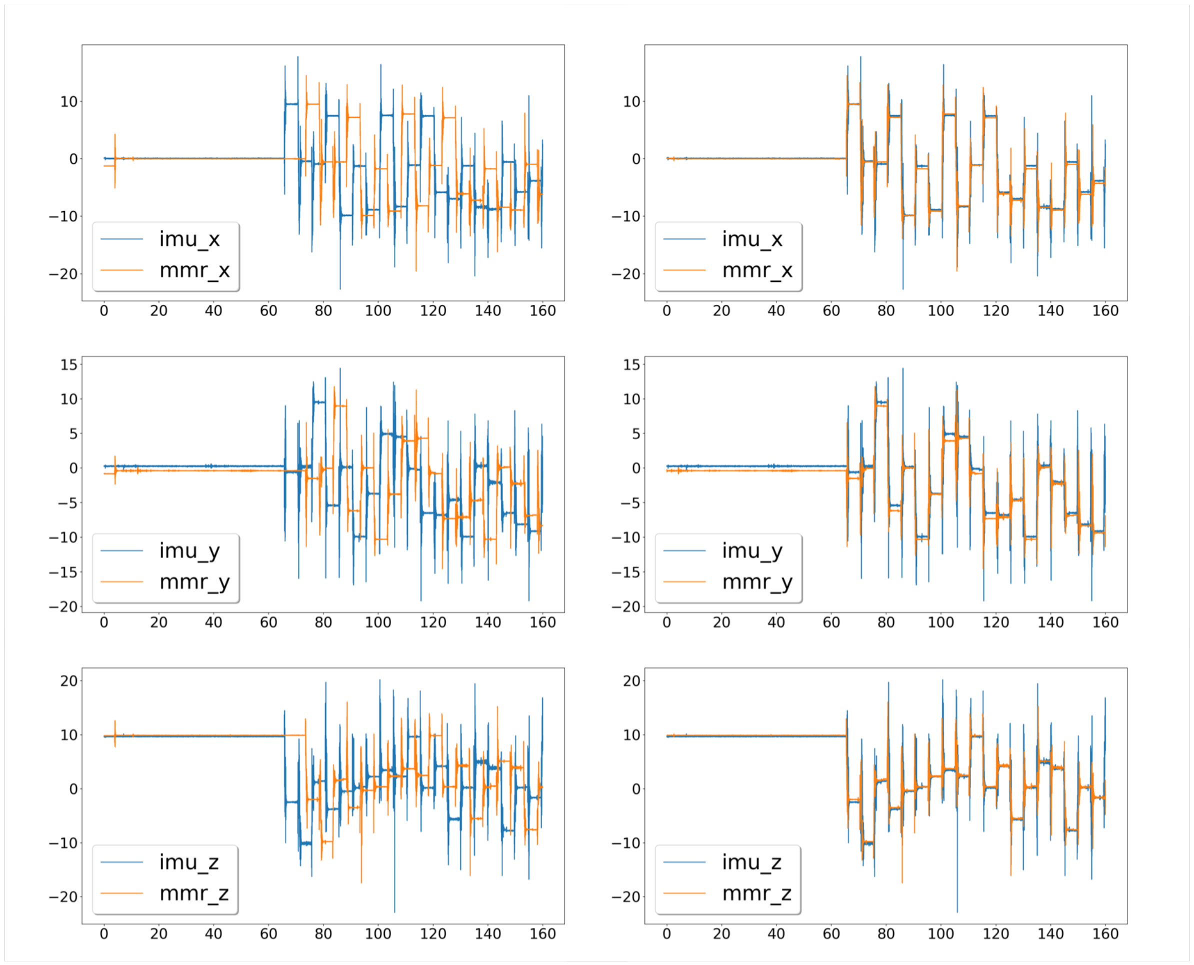


Figure 4.4: Unaligned (left) and aligned (right) time-series data

After applying Eq.(4.1), the Pearson correlation score before and after the calibration are summarized in Table 4.1.

Table 4.1: Pearson Correlation score before and after calibration

MMR	**Before**	**After**
acc_x	0.830	0.851
acc_y	0.843	0.905
acc_z	0.866	0.892
gyro_x	0.730	0.813
gyro_y	0.803	0.840
gyro_z	0.768	0.821

From the table, we can see from the results that sensor readings are getting better after the calibration.

4.4 Calibrating Galaxy S9 Built-in IMU

The calibration process for the Galaxy S9 built-in IMU follows the same steps. Except for this time, we need to build a data collection app using Android's motion sensor API[22].

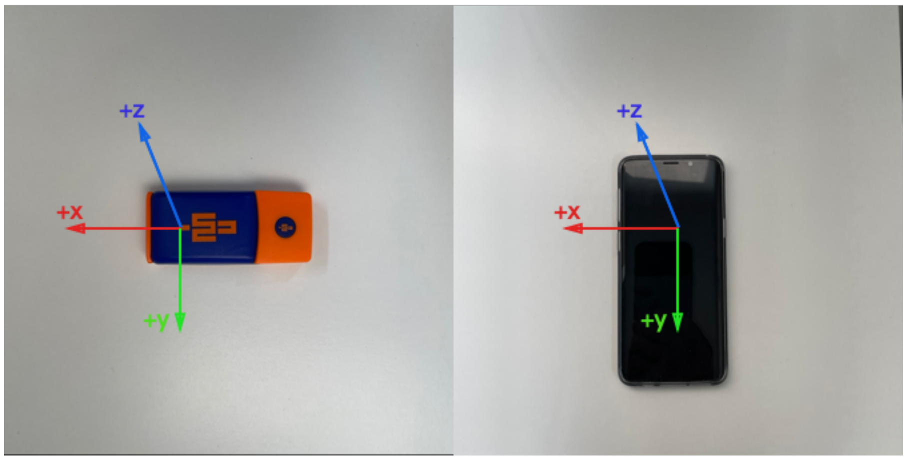


Figure 4.5: X,Y,Z axis direction for iMeasureU and Galaxy S9 built-in IMU

The collected data is stored on the mobile phone's local storage. A HTTP server is set up on a PC, so that the data can upload to server via the HTTP protocol. (Need to make sure the mobile phone and the PC are connected to the same LAN)

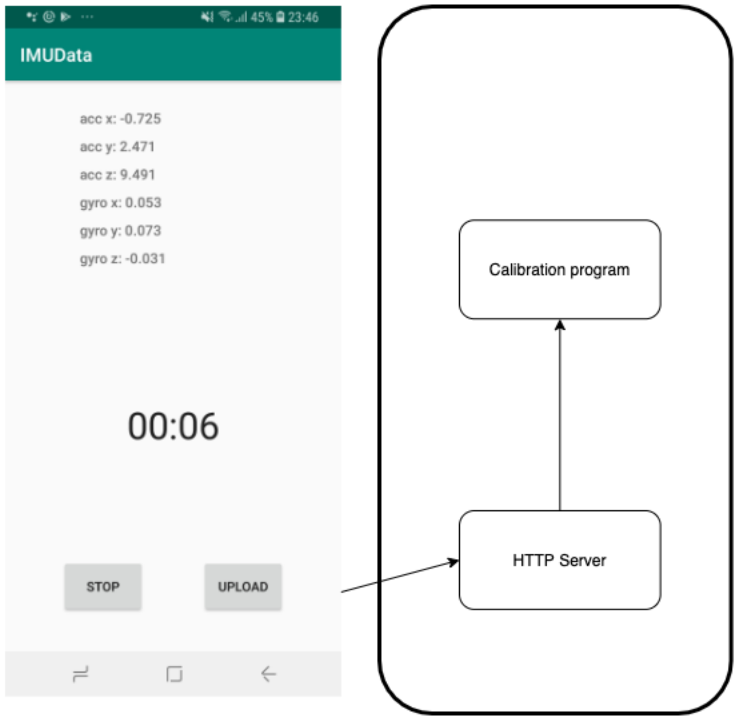

acc x: -0.725
acc y: 2.471
acc z: 9.491
gyro x: 0.053
gyro y: 0.073
gyro z: -0.031
00:06

Figure 4.6: Galaxy S9 collected data flow

The calibration program processes the raw data and calculates the calibration parameters as follows:

$$t_{acc} = \begin{bmatrix} 1 & 0.13 & -0.01 \\ 0 & 1 & 0.16 \\ 0 & 0 & 1 \end{bmatrix}, k_{acc} = \begin{bmatrix} 0.997 & 0 & 0 \\ 0 & 0.98 & 0 \\ 0 & 0 & 0.992 \end{bmatrix}, b_{acc} = \begin{bmatrix} -0.02 \\ 0.018 \\ -0.029 \end{bmatrix},$$
$$t_{gyro} = \begin{bmatrix} 1 & -0.08 & -0.01 \\ 0 & 1 & -0.021 \\ 0.01 & 0 & 1 \end{bmatrix}, k_{gyro} = \begin{bmatrix} 0.997 & 0 & 0 \\ 0 & 1.01 & 0 \\ 0 & 0 & 0.985 \end{bmatrix}, b_{gyro} = \begin{bmatrix} -0.06 \\ -0.018 \\ -0.018 \end{bmatrix}.$$

Given a raw sensor reading x (e.g., the acceleration), the calibrated reading x' is obtained by $x' = t * k * (x + b)$ where t is the misalignment matrix, k is the scale matrix, and b is the bias vector.

After applying Eq.(4.1), the Pearson correlation score before and after the calibration are:

Table 4.2: Pearson Correlation score before and after calibration

S9	**Before**	**After**
acc_x	0.751	0.793
acc_y	0.805	0.812
acc_z	0.826	0.84
gyro_x	0.730	0.752
gyro_y	0.776	0.79
gyro_z	0.741	0.772

We can see from the results that correlation has improved after the calibration.

Comparing the results from Tbl 4.3 and Tbl 4.4 shows MetaMotionR has a better result. Because the built-in IMU sensor in Galaxy S9 has much lower sensitivity ($\pm 8g$ for accelerometer, and $1000°/s$ for gyroscope) than iMeasureU and MetaMotionR (both support $\pm 16g$, $2000°/s$).

4.5 Future Works

From the experiments we conducted previously, the calibration platform and algorithm are effective. But there are some rooms to improve.

1. On the hardware side, the platform is not stable. It needs to be fixed on a flat surface using tapes. Otherwise, it may fall due to the inertial force when rotating.

2. The rotation is not linearly accelerated/decelerated. When the shaft moves from rotation to static, it has some slight vibration movements caused by inertial force, which has a negative impact on the quality of the collected data. Fig.4.4 shows that even for static intervals, the plotted data has some glitches. Since the calibration algorithm is highly dependent on the quality of the data, the

calibration may fail due to poor quality data. The rotatable shafts need a better driving mechanism which can support linear acceleration/deceleration.

3. Since smartphones are typically equipped with built-in thermometers, the calibration parameters can be compensated using the temperature readings[11].

Bibliography

[1] M. Li, B.H. Kim, and A. I. Mourikis. Real-time motion estimation on a cell-phone using inertial sensing and a rolling-shutter camera. *IEEE International Conference on Robotics and Automation, Karlsruhe, Germany*, pages 4697–4704, 2013.

[2] David Tedaldi, Alberto Pretto, and Emanuele Menegatti. A robust and easy to implement method for imu calibration without external equipments. *2014 IEEE International Conference on Robotics & Automation (ICRA)*, 2014.

[3] R.M. Rogers. Applied mathematics in integrated navigation systems. *American Institute of Aeronautics and Astronautics, AIAA education series*, 2003.

[4] J. J. Hall and R. L. Williams II. Inertial measurement unit calibration platform. *Journal of Robotic System*, 17(11):623–632, 1998.

[5] A. Kim and M.F. Golnaraghi. Initial calibration of an inertial measurement unit using an optical position tracking system. *Position Location and Navigation Symposium*, pages 96–101, 2004.

[6] Eduardo Mario Nebot and Hugh F. Durrant-Whyte. Initial calibration and alignment of low-cost inertial navigation units for land vehicle applications. *Journal of Robotic Systems*, 16(2):81–92, 1999.

[7] J.Schipper J.Lotters, P.Veltink, W.Olthuis, and P.Bergveld. Procedure for in-use calibration of triaxial accelerometers in medical applications. *Sensors and Actuators A: Physical*, 68(1–3):221–228, 1998.

[8] Skog I. and Proc Händel P. Calibration of a mems inertial measurement unit. In *17th IMEKO World Congress*, 2006.

[9] Syed Z F, Aggarwal P, Goodall C, Niu X, and El-Sheimy N. A new multi-position calibration method for mems inertial navigation systems. *Meas. Sci. Technol.*, 18:897–907, 2007.

[10] W.Fong, S.Ong, and A.Nee. Methods for in-field user calibration of an inertial measurement unit without external equipment. *Measurement Science and Technology*, 19, 2008.

[11] Chi Ming Cheuk, Tak Kit Lau, Kai Wun Lin, and Yun hui Liu. Automatic calibration for inertial measurement unit. In *Proc. of International Conference on Control, Automation, Robotics and Vision*, 2012.

[12] Yitong Zhang, Shuli Guo, Qiming Chen, Lina Han, and Quanjin Si. Noise identification and analysis in mems sensors using an optimized variable step allan variance. In *Proceedings of the 38th Chinese Control Conference*, 2019.

[13] Angelo M. Sabatini. A wavelet-based bootstrap method applied to inertial sensor

stochastic error modeling using the allan variance. *Measurement Science and Technology*, 17(11):2980–2988, 2006.

[14] Kenneth Levenberg and Donald Marquardt. Levenberg–marquardt algorithm.

[15] Runge Kutta Integration methods. `https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods`.

[16] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[17] iMeasureU. `https://imeasureu.com/capture-u/`.

[18] MetaMotionR. `https://mbientlab.com/metamotionrl/`.

[19] Pearson Correlation Coefficient. `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient`.

[20] Lex Fridman, Daniel E. Brown, William Angell, Irman Abdić, Bryan Reimer, and Hae Young Noh. Automated synchronization of driving data using vibration and steering events. *Pattern Recognition Letters*, 75:9–15, May 2016.

[21] Fast Fourier Transform. `https://en.wikipedia.org/wiki/Fast_Fourier_transform`.

[22] Android Developers Documentation. `https://developer.android.com/guide/topics/sensors/sensors_overview`.