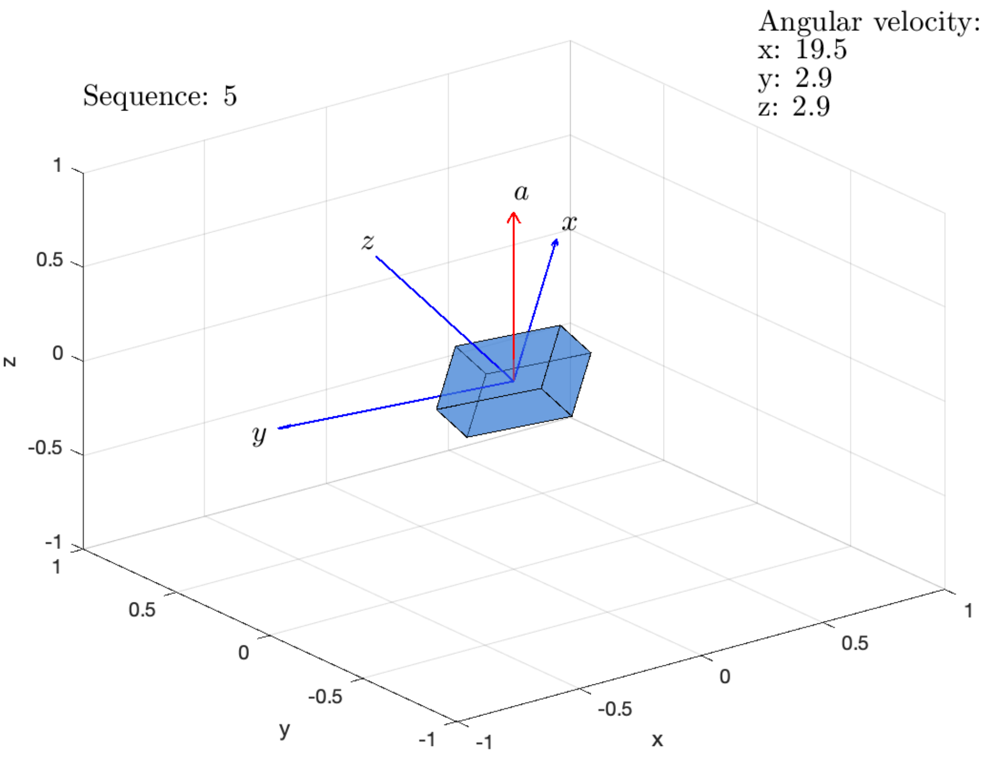
CHALMERS
UNIVERSITY OF TECHNOLOGY



Calibration of IMUs using Neural Networks and Adaptive Techniques

-Targeting a Self-Calibrated IMU

Master's thesis in System, Control and Mechatronics

ELLINOR CLAESSON
SARA MARKLUND

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019:30

Calibration of IMUs using Neural Networks and Adaptive Techniques

-Targeting a Self-Calibrated IMU

ELLINOR CLAESSON
SARA MARKLUND

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Applied Artificial Intelligence
Chalmers University of Technology
Gothenburg, Sweden 2019

Calibration of IMUs using Neural Networks and Adaptive Techniques
-Targeting a Self-Calibrated IMU
ELLINOR CLAESSON
SARA MARKLUND

Supervisors: David Oom, Mikael Tulldahl, CPAC Systems AB
Examiner: Peter Forsberg, Applied Artificial Intelligence

Master's Thesis 2019:30
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Applied Artificial Intelligence Research Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Simulation of a tri-axis IMU rotating around the center of its coordinate frame. The gravitational acceleration that the IMU is subjected to is denoted a.

Typeset in LaTeX
Gothenburg, Sweden 2019

Calibration of IMUs using Neural Networks and Adaptive Techniques
-Targeting a Self-Calibrated IMU
ELLINOR CLAESSON
SARA MARKLUND
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

This thesis presents an investigation of different sensor models for calibrations of IMUs and suggests an approach for adaptive calibration which is intended to be utilized during use of the sensor. The results show that using a neural network as sensor model can reduce the calibration error compared to if a linear sensor model is used. The results also show that the suggested adaptive algorithm can be used to approximate the sensor model but is not sufficiently accurate.

An IMU is a set of inertial sensors that can, among other applications, be used to estimate position and orientation of a body. An accurate calibration of the IMU improves the performance of the estimations and is hence an essential operation. The most commonly used calibration algorithms today use linear sensor models although the characteristics of the sensor are often argued to be nonlinear.

The thesis contains an investigation of whether neural networks efficiently can be used for nonlinear sensor model approximation. It was found that, for offline calibration, neural networks can approximate the sensor model accurately but needs more data than the linear model. More precisely, using a neural network trained on 400 data points, the calibration error was halved compared to a calibration using a linear model.

A problem with MEMS gyroscopes and accelerometers is that their characteristics change over time and are affected by parameters in their surroundings. To maintain an accurate approximation of the sensor model, the sensor would need to be re-calibrated. Most commonly used calibration methods today are time consuming and require expensive hardware, which make them complicated to perform in field.

An algorithm for adaptive online calibration of IMU sensors has been introduced. This method is designed to be used continuously to update the calibration and thereby adapt the sensor model to changes in the sensor's characteristics. The algorithm is iterative and alternates between calibrating the accelerometer using the gyroscope as reference and calibrating the gyroscope using the accelerometer as reference. To approximate the sensor model, neural networks are used. The result from evaluation of the developed adaptive algorithm shows that it can be used to approximate the sensor model but does not obtain as high accuracy as the offline algorithms. Hence it needs further development to be useful in practice.

Keywords: IMU, calibration, neural networks, adaptive algorithms, self-calibration.

Acknowledgements

First of all we would like to thank our supervisors, David Oom and Mikael Tulldahl, for their help and valuable advise during the project. We would also like to thank our examiner, Peter Forsberg for being committed and pushing us to think outside the box. Finally we would like to thank CPAC Systems AB, for having us as thesis workers and all employees for showing interest and appreciation in our work.

Ellinor Claesson, Sara Marklund, Gothenburg, June 2019

Supervisors: David Oom, Mikael Tulldahl, CPAC Systems AB
Examiner: Peter Forsberg, Applied Artificial Intelligence

Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
- 1.1 Background 2
- 1.2 Purpose 3
- 1.3 Scope 3
- 1.4 Outline of the Thesis 3

2 Theoretical Framework **5**
- 2.1 Sensor Modeling 5
  - 2.1.1 Linear Sensor Models 6
  - 2.1.2 Nonlinear Sensor Models 7
- 2.2 Artificial Neural Networks 7
  - 2.2.1 Structure of the Feed Forward Neural Network 8
  - 2.2.2 Supervised Learning 10
  - 2.2.3 Validation of the Networks Performance 10

3 Offline Calibration of Accelerometer **13**
- 3.1 Calibration Assumptions 13
- 3.2 Implementation of Linear Sensor Model Algorithm 13
- 3.3 Implementation of Nonlinear Neural Network Algorithm 14
  - 3.3.1 Preprocessing of Accelerometer Measurements 15
  - 3.3.2 Structure and Training Parameters 15
  - 3.3.3 Using the Neural Network for Calibration 16
- 3.4 Evaluation of Offline Calibration Algorithms 16
  - 3.4.1 Result from Calibration using 18 Orientations 17
  - 3.4.2 Result from Calibration using 1000 Orientations 19
  - 3.4.3 Result from Calibration using 18-1000 Orientations 20
- 3.5 Discussion 21

4 Online Calibration of Accelerometer and Gyroscope **23**
- 4.1 Calibration Assumptions 24
- 4.2 Dataset 2 - Simulated Dataset 25
- 4.3 Initial Training 27

4.4 Adaptive Gyroscope Calibration 28
4.5 Adaptive Accelerometer Calibration 29
4.6 Evaluation of Online Calibration Algorithm 31
4.6.1 Result During Calibration 31
4.6.2 Result After Calibration 34
4.7 Discussion 36
4.7.1 Convergence of Adaptive Algorithm 36
4.7.2 Accuracy of Adaptive Algorithm 37
4.7.3 Practical Aspects 37
4.8 Further Development 38
4.8.1 Finer Tuning of Training Parameters 38
4.8.2 Eliminate Assumptions 38
4.8.3 Alternative Sensor Model 38

5 Conclusions **39**

List of Figures

1.1 Schema of dead-reckoning 1

2.1 Description of scale and bias 5
2.2 Description of non-orthogonality 6
2.3 Structure of a feed forward neural network 8
2.4 The activation functions ReLU and Leaky ReLU 9
2.5 The concept of overfitting and underfitting 11

3.1 Overview of the nonlinear neural network algorithm 15
3.2 Architecture of NN used in the nonlinear neural network algorithm . 16
3.3 Utilization of the nonlinear neural network algorithm 16
3.4 Training samples in Dataset 1 17
3.5 Eighteen of the training samples in Dataset 1 17
3.6 Calibrated measurements vs ground truth using 18 calibration orientations 18
3.7 Histogram of error using 18 calibration orientations 18
3.8 Calibrated measurements vs ground truth using 1000 calibration orientations 19
3.9 Histogram of error using 1000 calibration orientations 19
3.10 Mean error as a function of number of calibration samples 20

4.1 Overview of the adaptive algorithm 24
4.2 Training samples in Dataset 2 25
4.3 Modeling of angular rates for training set in Dataset 2 26
4.4 Modeling of angular rates for test set in Dataset 2 27
4.5 Architecture of NN used in the adaptive algorithm 27
4.6 Overview of the adaptive algorithm of gyroscope 28
4.7 Visualization of coordinate frames for loss calculation, gyroscope . . . 29
4.8 Visualization of separated coordinate frames in loss 29
4.9 Overview of the adaptive algorithm of accelerometer 30
4.10 Visualization of coordinate frames for loss calculation, accelerometer . 31
4.11 Calibrated accelerometer measurements vs ground truth after initial training 32
4.12 Calibrated gyroscope measurements vs ground truth after initial training 32
4.13 Error during online adaptive calibration 33
4.14 Comparison of the convergence of the two sensors 34

4.15 Calibrated accelerometer measurements vs ground truth 34
4.16 Histogram of the error after online calibration, accelerometer 35
4.17 Calibrated gyroscope measurements vs ground truth 35
4.18 Histogram of the error after online calibration, gyroscope 36

List of Tables

3.1 Mean error and error standard deviation after offline calibration with l calibration orientations 20

4.1 Mean error and error standard deviation after online calibration, accelerometer 35

4.2 Mean error and error standard deviation after online calibration, gyroscope 36

1
Introduction

In many applications it is desirable to know the position, orientation and motion of a body. To obtain such information inertial sensors are widely used. An inertial measurement unit (IMU) is a combination of inertial sensors. A common configuration of the IMU contains three orthogonally mounted accelerometers and three orthogonally mounted gyroscopes [1]. The accelerometers measure the specific force, in g, acting on the body and the gyroscopes measure the angular rate, in rad/s, of the body [2]. The IMU is often used in Inertial Navigation Systems (INS) [3], which utilize the raw IMU measurements and convert the data into information about position, velocity and attitude. The IMUs are also commonly used as orientation sensors in consumer products, such as smart phones and gaming remote controls [2].

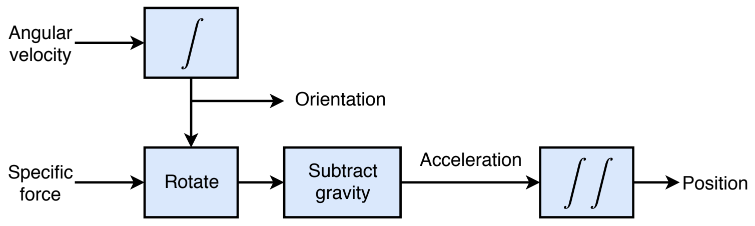


Figure 1.1: Process of estimating position and orientation from IMU. Integration of the angular velocity, given by the gyroscope, yields the orientation. The orientation is used to rotate the accelerometer and subtract the gravity before integrating the acceleration twice into a position.

When using an IMU for the purpose of navigation, information about position and orientation is desired, which can be estimated by fusing the accelerometer and gyroscope measurements. In the process of estimating position and orientation, the registered angular rates are integrated into information about orientation of the body. When the orientation is known, the gravitational acceleration can be deducted from the registered acceleration. The remaining acceleration is in turn integrated twice into information about position. A schema of the process, often called *dead-reckoning*, is shown in Figure 1.1. If the sensors provide the true angular velocity and the true acceleration, and if position and orientation were known at the initial state, perfect new estimates would be obtained from dead-reckoning.

In practice, the sensors are not perfect and the measurements are altered by noise. Due to integration of the measurements, errors will accumulate and grow over time. This means that even a small disparity between the interpretation of the measurements and the true acceleration/angular rate will, over time, cause a significant error in position, orientation and velocity [2].

Calibration aims to find a sensor model that maps the sensor output to ground truth. The sensor model is assumed to have the property

$$\boldsymbol{q}_t = f(\boldsymbol{q}_m) + \boldsymbol{n}, \tag{1.1}$$

where $\boldsymbol{q}_t$ is the true quantity, f is a static, deterministic and bijective function of the measured quantity $\boldsymbol{q}_m$ and $\boldsymbol{n}$ is stochastic noise. The noise cannot be predicted and is thus impossible to compensate for. However, if f could be estimated, then the function could be applied to all measurements. In a statistical sense, applying the function, f, could improve the performance of the applications that utilize the best known sensor data.

1.1 Background

There are a wide variety of approaches for collection of calibration data and identification of the sensor model. Algorithms that are said to operate *offline* have disjointed processes for sampling data and performing the calibration, i.e the calibration is performed on data that is collected in the past. Offline calibration results in a sensor model which is used to process the sensor output in future use. Algorithms that are said to operate *online* perform the calibration based on the data that is being sampled at real-time. Online calibration can hence be adaptive to changes in the sensor's behaviour.

A common approach of offline calibration, as seen in [3]–[6], is to predefine a sensor model and then use some optimization method to find the parameters in the chosen model. To identify the values of the parameters the sensor output is measured under known conditions. A different approach could be to train an artificial neural network to approximate the sensor model. Neural networks are known to perform well when used for function approximation and are capable of approximating functions with highly nonlinear behaviour [7]. Calibration of IMU sensors using neural networks have successfully been tested by [8]–[10]. Using a neural network as sensor model enables the optimizer to find other properties and nonlinearities than if a predefined sensor model, with e.g. a given order of polynomial and a given set of parameters, is used.

Regardless of whether the sensor model is predefined or if a neural network is used to approximate it, the calibration algorithm requires labeled data, i.e. data that has been collected under known conditions. The amount of data needed to find the parameters in the sensor model depends on the choice of model and the choice of optimization algorithm. However, the process of collecting raw data from the sensors, that is to be used in the calibration algorithms, involves sampling the sensor under known conditions. Accelerometer data is usually gathered by placing the sensor in a number of static positions where the expected output of each sensor axis is known. How accurate the sensor is positioned has impact on the result of the calibration and hence, external equipment, like a robot and a rig, is widely used. Gathering of gyroscope data often includes the use of a rate table. With a rate table, the IMU can be rotated with a specified angular velocity and hence the expected output of the gyroscope is known [3], [6], [10].

The mentioned methods have proven to accurately map measured quantity into true quantity at the time of calibration. Yet, it is known that the sensor characteristics tend to change over time and are highly dependent on temperature changes [11]–[15]. To maintain accurate interpretations of the sensor measurements, the IMU needs to be re-calibrated in course of time. Since most calibration methods require equipment with high precision, as a robot and a rate table, re-calibration is, if even possible, demanding and time consuming to perform in field. It would therefore be desirable if the calibration of the IMU could be performed in field and be adaptive to changes in the sensors nature and local environment.

1.2 Purpose

The purpose of this thesis is to construct a calibration algorithm for the sensors in an IMU which, by itself, can find a sensor model using neural networks and investigate how accurate a neural network can approximate the sensor characteristics compared to a linear model. The project also aims to extend the algorithm to circumvent the need for labeled sensor data, and hence make the IMU self-calibrated in the sense that it only needs information from itself to perform the calibration.

1.3 Scope

This thesis focuses on calibration of microelectromechanical systems (MEMS) sensors. The project will make use of simulated data and already existing IMU-data. The simulated measurements, $q_{measurements}$, will be modeled as

$$q_{measured} = sq_{true} + b \tag{1.2}$$

where q_{true} is the true signal, s is a scale factor and b is bias. Throughout the project the IMU is not subjected to linear accelerations while measurements are collected, and is hence only affected by gravitational acceleration. In addition the temperature is kept constant.

1.4 Outline of the Thesis

The workflow in this thesis can mainly be divided into the development of three separate algorithms. The theoretical framework used for the development of all three algorithms is presented in chapter 2. It contains a description of common characteristics of sensor models and introduces theory behind neural networks and machine learning algorithms.

Chapter 3 presents two algorithms for offline calibration of an accelerometer based on data collected with external equipment. One is based on well known theories about sensor calibration, which utilizes a linear sensor model, and one utilizes a nonlinear sensor model approximated by a neural network. The derivation and implementation of the algorithms are introduced separately, followed by a joint evaluation where the the linear sensor model is used as a benchmark. The performance of the two algorithms are shortly discussed in order to chose what model to use for the development of an adaptive calibration algorithm.

In chapter 4 an approach for simultaneous online calibration of an accelerometer and a gyroscope, that does not require any external equipment, is introduced. The algorithm is evaluated on simulated data. Interpretations of the result from evaluation of the algorithm are discussed and ideas for future work are presented. Conclusions from the project, including both offline and online calibration, are found in chapter 5.

2
Theoretical Framework

This chapter presents theory that has been essential for research and development of the algorithms introduced in this thesis. The chapter comprises two main topics. The first part explains some characteristics of the sensors that entails the need of calibration and how these characteristics can be modeled linearly. The second part describes theory about neural networks and how they are trained.

2.1 Sensor Modeling

The two main properties of the sensor model, that are the most essential to compensate for, are scale and bias. The bias is a constant offset between the true and the measured quantity and the scale is the ratio between true and measured quantity [1]. Scale and bias can be described by looking at a linear function

$$y = kx + m, \tag{2.1}$$

and is illustrated in Figure 2.1. Here y represents the measured specific force/ angular rate on the vertical axis, x represents the true specific force/angular rate on the horizontal axis, k represents the scale and m represents the bias.

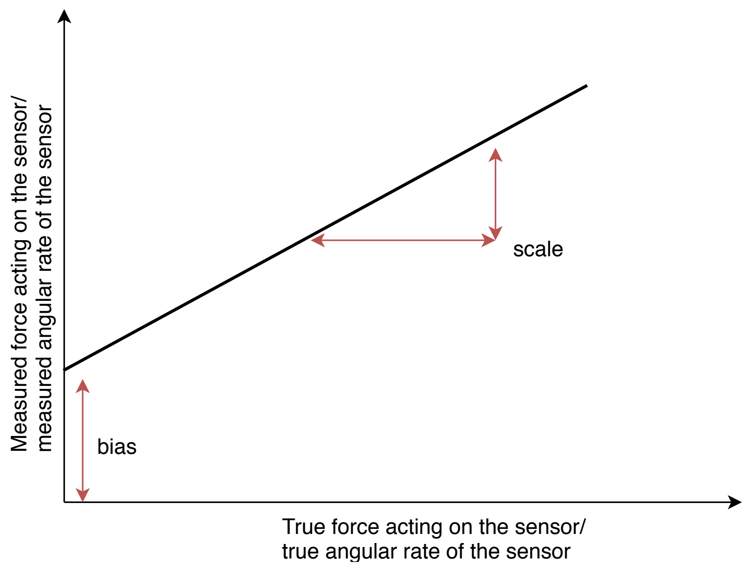

Measured force acting on the sensor/
measured angular rate of the sensor
scale
bias
True force acting on the sensor/
true angular rate of the sensor

Figure 2.1: Explanation of scale and bias. The linear function is the relation between the measured output of the sensor and the true specific force/angular rate. The bias is where the line cuts the y-axis and the scale is the inclination of the line.

A three-axis accelerometer or gyroscope consists of three individual sensors in total, one for each orthogonal axis represented in the three dimensional euclidean space (x, y, z). The precision of the mounting of the three sensors is of great importance, and if the axes are not mounted perfectly orthogonal, the sensor model will be affected. Figure 2.2 shows an illustration of two coordinate frames. The dashed grey frame is orthogonal while the black frame is non-orthogonal as $\alpha \neq 90°$, $\beta \neq 90°$, $\gamma \neq 90°$. If the black frame represents the sensor frame, it will introduce cross-talk between the sensor's axes, i.e the output of one axis will impact the other axes. To compensate for non-orthogonality, the measurements, taken in the sensor's coordinate frame, need to be mapped into an euclidean coordinate frame [1].

Misalignment is another property that has impact on the sensor model. It occurs if the sensor is not mounted such that the coordinate frame of the sensor's sensitivity axes coincide with the coordinate frame of the body that the sensor is mounted on. [1].

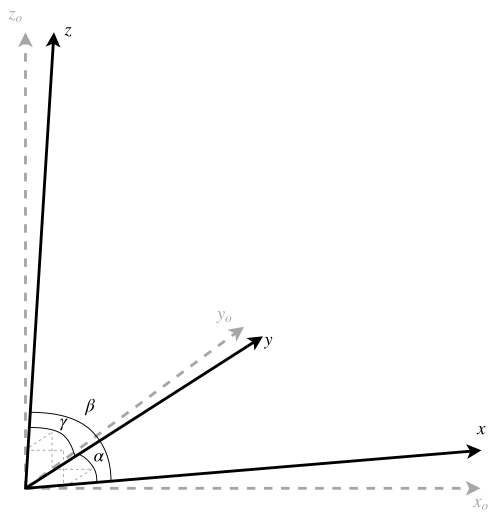


Figure 2.2: Illustration of the error caused by non-orthogonal sensitivity axes of the sensors. The grey dashed axes x_o, y_o and z_o builds an orthogonal coordinate system, i.e there is 90 degrees between all axes. The axes of the sensor x, y and z (in black) are not orthogonal since the angles $\alpha \neq 90°$, $\beta \neq 90°$, $\gamma \neq 90°$.

2.1.1 Linear Sensor Models

For an accelerometer, one of the simplest models that can be used for calibration is described in [4] as

$$\underbrace{\begin{bmatrix} \tilde{q}_x \\ \tilde{q}_y \\ \tilde{q}_z \end{bmatrix}}_{\tilde{\boldsymbol{q}}} = \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}}_{\boldsymbol{S}} \underbrace{\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}}_{\boldsymbol{q}} + \underbrace{\begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}}_{\boldsymbol{b}}, \tag{2.2}$$

where $\tilde{\boldsymbol{q}}$ represents the true quantity, $\boldsymbol{q}$ is the measured quantity, $\boldsymbol{S}$ is a matrix containing the sensor scaling errors and $\boldsymbol{b}$ is the sensor bias. In [4], the true and measured quantity is acceleration such that $\tilde{\boldsymbol{q}}$ is true acceleration and $\boldsymbol{q}$ is measured acceleration. Gyroscopes in general have similar characteristics as the accelerometer and thus equation (2.2) can be used to describe the sensor model for a gyroscope, such that $\tilde{\boldsymbol{q}}$ is true angular velocity and $\boldsymbol{q}$ is measured angular velocity.

The sensor models introduced by [5] and [6] compensate, apart from scale and bias, for non-orthogonality and misalignment. The equation that describes the sensor model is expressed as

$$\underbrace{\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}}_{\boldsymbol{q}} = \underbrace{\begin{bmatrix} s_x & m_{xy} & m_{xz} \\ m_{yx} & s_y & m_{yz} \\ m_{zx} & m_{zy} & s_z \end{bmatrix}}_{\boldsymbol{S}} \underbrace{\begin{bmatrix} \tilde{q}_x \\ \tilde{q}_y \\ \tilde{q}_z \end{bmatrix}}_{\tilde{\boldsymbol{q}}} + \underbrace{\begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}}_{\boldsymbol{b}}, \tag{2.3}$$

where $\tilde{\boldsymbol{q}}$ represents the true quantity (acceleration or angular velocity), $\boldsymbol{q}$ is the measured quantity (acceleration or angular velocity), $\boldsymbol{b}$ is the sensor bias and $\boldsymbol{S}$ is a matrix containing the scale factor (diagonal elements) and multiplicative errors due to non-orthogonality within the sensor axes and misalignment caused by mounting of the sensors (non-diagonal elements).

2.1.2 Nonlinear Sensor Models

It has been argued that the mapping between measured quantity and true quantity for IMU sensors is not completely linear and hence a nonlinear sensor model could be a more accurate approximation. For this purpose a polynomial of second order or higher can be used to predefine the sensor model. Another tested approach to nonlinear modeling of the sensor is to use neural networks (NN). Among those who have presented calibration methods using NN the usage of feed forward NN is widely adopted. Studied approaches for training such NNs include backpropagation [9], [10] or extended versions of backpropagation that also use the Levenberg–Marquardt algorithm for optimization of the networks weights [8]. The NN suggested in [8]–[10] all use one single hidden layer but what sets the methods apart is what activation function that is used in the hidden layer and what algorithm is used for updating the weights in the network.

2.2 Artificial Neural Networks

Artificial neural networks, usually referred to as simply neural networks (NN), are computational structures that are inspired by the architecture of the biological brain [16]. The networks are collections of units called neurons with weighted connections. The neurons are gathered together in layers, where the first layer is called the input layer and the last layer is called the output layer. Layers between the input and the output layer are called hidden layers.

Neural networks have been commonly used in applications including function approximations, pattern recognition, classification and control [16]. The process of calibrating a inertial sensor involves approximation of a sensor model, i.e function approximation. In function approximation the neural network is trained to represent an arbitrary function, $f(\boldsymbol{x})$ for any $\boldsymbol{x}$ in the domain of f. Normally, the network is trained using training data that only covers a subset of the domain and then it is up to the network to predict the output for the rest of the points in the domain [16].

As the theory of neural networks has been adopted to solve a wide variety of tasks, numerous types of network structures have been introduced including feed forward neural network (FFNN), recurrent neural network (RNN) and convolutional neural network (CNN). CNNs are often used in image analysis. RNNs have internal memory and are often used in applications where the context is important, such as speech recognition or translation. FFNNs are less complex than CNNs and RNNs, and are sufficient for approximation of one-to-one mapping functions [17].

2.2.1 Structure of the Feed Forward Neural Network

The FFNN has an input layer and an output layer and in between there can be any number of hidden layers. The more hidden layers the network has, the deeper it is said to be [18]. The information in the FFNN transfers in one direction, forward, from input layer to output layer. All the nodes in the network are fully connected which means that each neuron in one layer is connected to all neurons in the next layer [17], as depicted in Figure 2.3.

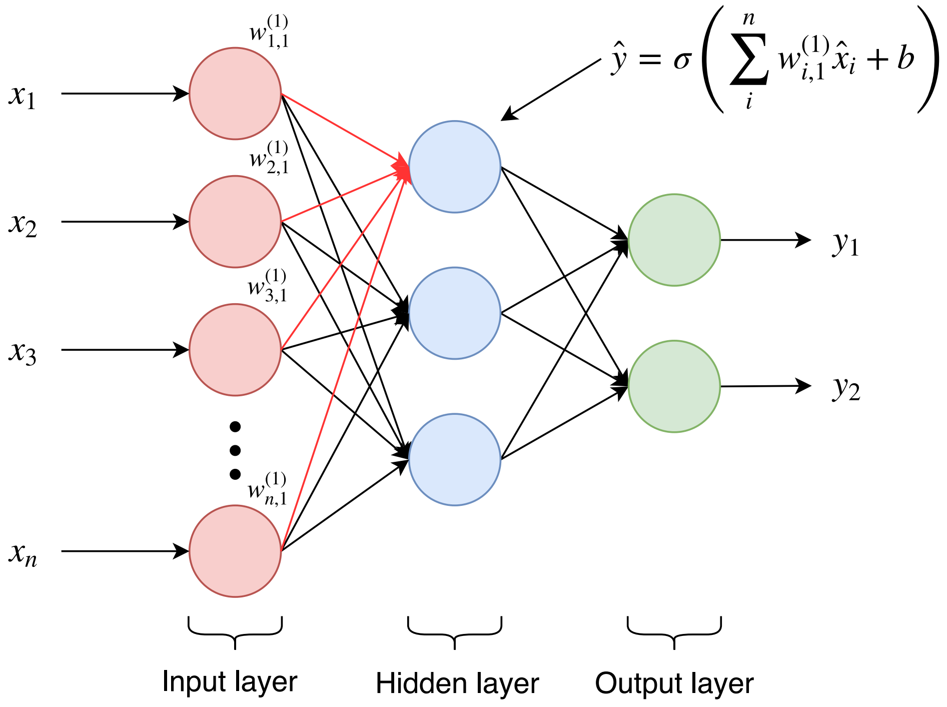


Figure 2.3: Structure of a feed forward neural network. In each neuron the weighted sum of the output from the previous layers is calculated. The sum is given as input to an activation function which determines the final output of the neuron. The equation in the image is utilized by the highlighted neuron which takes the connections marked with red into account.

Figure 2.3 shows an example of the structure in a FFNN. This neural network has n inputs, one hidden layer with three neurons and two outputs. In the figure,

x is the input to the network and y is the output of the network. Each neuron of the network is influenced by an activation function. The activation function, which often is a nonlinear function, decides whether the neuron should be activated or not. The output, $\hat{y}_j^{(k)}$, of each neuron, is calculated as

$$\hat{y}_j^{(k)} = \sigma\left(\sum_i^n w_{i,j}^{(k)} \hat{x}_i + b_j\right), \tag{2.4}$$

where $\sigma(.)$, is the activation function, $w_{i,j}^{(k)}$ is the weight for the i-th connection to the j-th neuron in the k-th layer, $\hat{x}_i$ is the output from neuron i in the previous layer and b is the bias.

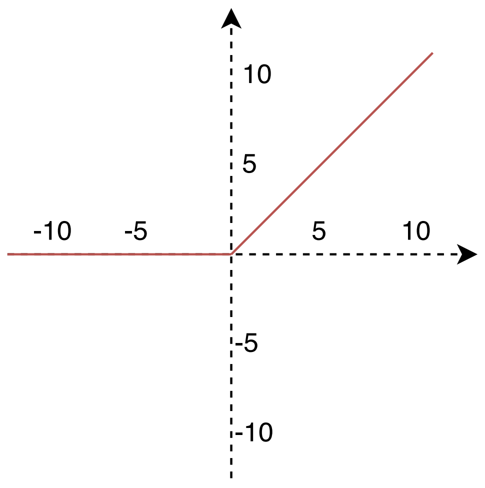
One commonly used activation function, due to its simplicity to train and its good performance [19], is the rectified linear unit function, *ReLU*

$$\sigma(x) = \max(0, x), \tag{2.5}$$

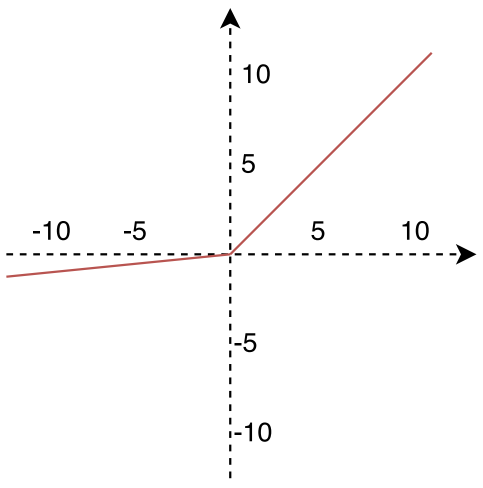
which maps the input, x, to the output if the input is positive and outputs zero otherwise. Figure 2.4a shows the behaviour of the ReLU function. If the ReLU is used on ill conditioned data, an issue called the "dying ReLU problem" can occur. The phenomena appears when the output of the majority of the neurons have zero output, causing the gradients to also be zero. It results in the weights corresponding to the zero gradient not being updated. To overcome this problem a number of different versions of the ReLU function have been introduced. One such variant is the *Leaky ReLU* function [20],

$$\sigma(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x > 0 \end{cases}, \tag{2.6}$$

where α is a small constant which makes the horizontal line to be slightly inclined as shown in Figure 2.4b.



(a) The standard ReLU function



(b) The variant Leaky ReLU function

Figure 2.4: Visualization of two non-linear activation functions, in a) the standard ReLU function and in b) a variant of the ReLU function called Leaky ReLU.

2.2.2 Supervised Learning

Supervised learning is an iterative process that is often used as training approach when applied to a dataset that contains labels. The labels denote the expected output of the network for the given input. In image classification the labels might indicate whether a photo contains a cat or a dog. In the case of this report the labels are the true acceleration and angular rate of the IMU.

The labels are given to the network as targets and the training process aims to adjust the network into an accurate mapping between the input data and the corresponding targets [21]. Inside the network the training data is propagated through every neuron, which results in a forward cascade of computations, using the current set of parameters [22]. The predicted output of the network is compared to the intended target output in a loss function. A commonly used loss function for function approximation is the *mean squared error* (MSE) function [21],

$$\mathcal{L} = \mathrm{MSE}(y, \hat{y}) = \frac{1}{JK} \sum_{j=1}^{J} \sum_{k=1}^{K} (y_{j,k} - \hat{y}_{j,k})^2, \tag{2.7}$$

where J is the total number of samples, K is the total number of neurons in the output layer, y is the target and $\hat{y}$ is the predicted output of the network.

The objective of the training process is to minimize the loss function, which is done by backpropagation. In backpropagation, the gradient of the loss function with respect to the parameters is calculated. The parameters are then updated in the steepest descent direction according to

$$\theta = \theta - \eta \nabla \mathcal{L}(\theta), \tag{2.8}$$

where θ represents the parameters, η represents the learning rate which determines how much the parameters are adjusted and $\nabla \mathcal{L}(\theta)$ is the gradient of the loss function [23].

Finding the optimal set of parameters can be a costly process that takes a lot of time and computational power. To speed up the process different algorithms called *optimizers* have been developed, which all are based on gradient descent. One of the most used optimizer is the *stochastic gradient descent* (SGD) algorithm, where a minibatch of m samples are drawn from the training set in each iteration and the parameters are updated using the average gradient of the minibatch [24]. One problem with the SGD algorithm is the choice of learning rate. Another frequently used optimizer is the *Adam* algorithm, which is an extension to the SGD algorithm. The method uses individual learning rates for different parameters that are adaptive and changes during the training process to efficiently find the optimal set of parameters [25]. One repetition of forward propagation and backpropagation of all minibatches is called an epoch and the training continues until a certain termination condition is fulfilled [21].

2.2.3 Validation of the Networks Performance

If the training continues too long, there is a risk that the network learns the training data too detailed and the network will not be able to perform well on unseen data.

This is called overfitting. The opposite to overfitting is called underfitting and means that the network is not able to learn the training data good enough. To explain the concept of over- and underfitting Figure 2.5 illustrates a set of data points in three different plots. The red line illustrates a function that maps input to output, approximated by a neural network. In the left plot, the function is both sufficiently detailed and generalizing. In the middle plot, the function has overfitted to the data, while in the right plot, the function is underfitted [26].

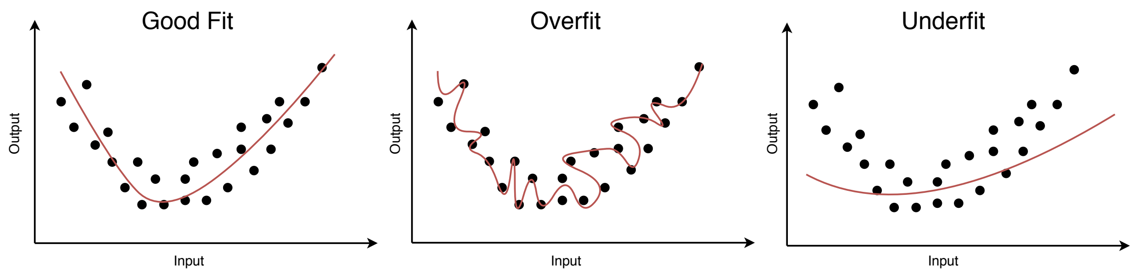


Figure 2.5: Illustration of the concept of overfitting and underfitting.

To be able to monitor the network's performance and prevent the problem with overfitting and underfitting, the data is often split into a training set, a validation set and test set. It is common to use 70-80% of all available data for training, 10-15% for validation and 10-15% for test [21]. The training set is then used during the training and the validation set is used as unseen data after each epoch in the training to continuously validate how the network perform on unseen data. The test set is used to evaluate the network after training.

Having a training loss that is significantly smaller than the validation loss is an indication of an overfitted NN. To avoid overfitting a network, the training can be executed until the validation loss stops improving. If the validation loss is still much worse than the training loss, the network is overfitted. The problem can be solved by either using more training data or change the structure of the network. If the training loss does not reach the desired degree it is an indication of an underfitted NN, which can be solved by either train for a longer time or change the structure of the network [21].

3 Offline Calibration of Accelerometer

This chapter describes the development and implementation of two calibration algorithms for offline calibration. One is based on a commonly used linear sensor model and is therefore used as benchmark in this thesis. The other algorithm uses neural networks to approximate a nonlinear sensor model. The two algorithms are then used to calibrate measurements with the purpose to investigate whether the linear or the nonlinear model provides the best calibration.

The algorithms run on labeled datasets, i.e datasets that consist of pairs of measurements and corresponding ground truth. Such data is obtained by sampling the sensors under known conditions. For accelerometers, labeled data was provided at the beginning of the project. In addition, a robot for accurate positioning has been available. Corresponding data has not been available for gyroscopes and neither has a rate table been available. Therefore, the gyroscope is not studied in this chapter and it is considered sufficient to evaluate the algorithms' performance on an accelerometer only. Since neural networks are known to perform well when trained on a large set of data, the two models are evaluated on datasets of different sizes.

3.1 Calibration Assumptions

The robot, which is used to collect stationary accelerometer data, positions the accelerometer in the requested orientations with exact precision. When the accelerometer is stationary, the only acceleration affecting it is the gravitational acceleration. The ground truth can hence be calculated as the gravitational acceleration measured in the sensors local coordinate frame given by the robot.

3.2 Implementation of Linear Sensor Model Algorithm

The linear sensor model algorithm uses the sensor model presented in equation (2.3), which is constructed to compensate for scale, bias, non-orthogonality and misalignment. Equation (2.3) was rewritten into

$$\underbrace{\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}}_{\boldsymbol{a}} = \underbrace{\begin{bmatrix} s_x & m_{xy} & m_{xz} & b_x \\ m_{yx} & s_y & m_{yz} & b_y \\ m_{zx} & m_{zy} & s_z & b_z \end{bmatrix}}_{\boldsymbol{M}} \underbrace{\begin{bmatrix} \tilde{a}_x \\ \tilde{a}_y \\ \tilde{a}_z \\ 1 \end{bmatrix}}_{\tilde{\boldsymbol{a}}}, \tag{3.1}$$

where $\boldsymbol{a}$ represents measured acceleration, $\tilde{\boldsymbol{a}}$ represents true acceleration, $\boldsymbol{M}$ represents scale, misalignment, non-orthogonality and bias terms.

To solve equation (3.1), i.e to find the calibration parameters in $\boldsymbol{M}$, the algorithm needs accelerometer measurements, $\boldsymbol{a}_l$, and the associated ground truth, $\boldsymbol{a}_{gt,l}$, where $l \in [1, L]$ and L is the number of labeled data pairs. Finding the optimal set of parameters in the sensor model is an optimization problem and in this case the least squares method [27] has been used. The calibration was done with the sensor set in L known stationary orientations, which yields the system

$$\underbrace{\begin{bmatrix} a_{1x} & a_{2x} & \dots & a_{Lx} \\ a_{1y} & a_{2y} & \dots & a_{Ly} \\ a_{1z} & a_{2z} & \dots & a_{Lz} \end{bmatrix}}_{\boldsymbol{U}} = \underbrace{\begin{bmatrix} s_x & m_{xy} & m_{xz} & b_x \\ m_{yx} & s_y & m_{yz} & b_y \\ m_{zx} & m_{zy} & s_z & b_z \end{bmatrix}}_{\boldsymbol{M}} \underbrace{\begin{bmatrix} a_{gt,1x} & a_{gt,2x} & \dots & a_{gt,Lx} \\ a_{gt,1y} & a_{gt,2y} & \dots & a_{gt,Ly} \\ a_{gt,1z} & a_{gt,2z} & \dots & a_{gt,Lz} \\ 1 & 1 & \dots & 1 \end{bmatrix}}_{\boldsymbol{A}}, \tag{3.2}$$

where $\boldsymbol{U}$ is a $3 \times L$ matrix containing the measured acceleration in x-axis, y-axis and z-axis in L stationary orientations, $\boldsymbol{M}$ is a 3×4 matrix containing the calibration parameters and $\boldsymbol{A}$ is a $4 \times L$ matrix representing the ground truth. The parameters in $\boldsymbol{M}$ are then isolated using the method of least squares as

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{A}^T(\boldsymbol{A}^T\boldsymbol{A})^{-1}. \tag{3.3}$$

The matrix $\boldsymbol{M}$ can be separated into one matrix representing the scale, misalignment and non-orthogonality, $\boldsymbol{S}$, and one vector representing the bias, $\boldsymbol{b}$,

$$\boldsymbol{S} = \begin{bmatrix} s_x & m_{xy} & m_{xz} \\ m_{yx} & s_y & m_{yz} \\ m_{zx} & m_{zy} & s_z \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}. \tag{3.4}$$

Now $\boldsymbol{S}$ and $\boldsymbol{b}$ can be used to calibrate the accelerometer measurements, $\boldsymbol{a}_l$ by

$$\boldsymbol{a}_{c,l} = \boldsymbol{S}^{-1}(\boldsymbol{a}_l - \boldsymbol{b}), \tag{3.5}$$

where $\boldsymbol{a}_{c,l}$ are the calibrated accelerometer measurements.

3.3 Implementation of Nonlinear Neural Network Algorithm

The approach presented in this section approximates the sensor model with a neural network. Using a neural network to find the sensor model, enables the algorithm to find other properties than those that are stated in equation (3.4). Also the neural network can e.g. model nonlinearities.

An overview of the structure of the developed algorithm using a neural network is shown in Figure 3.1. The idea is to collect sensor data and preprocess it using normalization. The neural network is then fed with the preprocessed measurements in x-axis, y-axis and z-axis as input, and trained to output the corrected sensor values in each axis. The training stops when some termination criterion is fulfilled at which the network should be ready to be used for calibration of new sensor data.

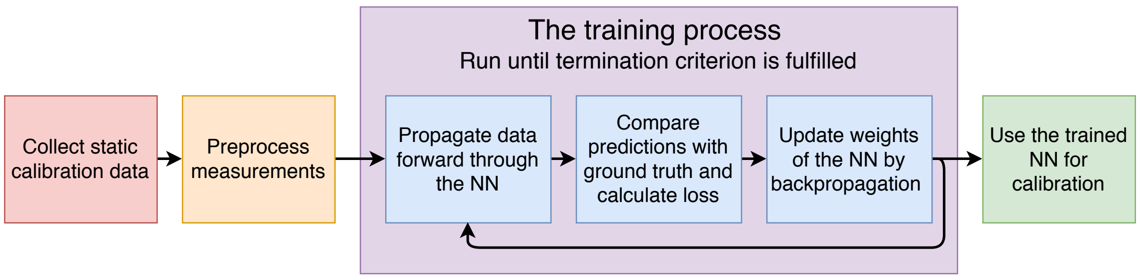

The training process
Run until termination criterion is fulfilled
Collect static calibration data
Preprocess measurements
Propagate data forward through the NN
Compare predictions with ground truth and calculate loss
Update weights of the NN by backpropagation
Use the trained NN for calibration

Figure 3.1: Illustration of the offline algorithm used for calibrating an accelerometer with a neural network using known orientations as target.

3.3.1 Preprocessing of Accelerometer Measurements

Normalizing the input data, i.e. manipulating the accelerometer measurements, $\boldsymbol{a}_l$, such that they are scaled into a standardized interval, can help the neural network to converge to a solution more quickly [28]. In this algorithm the accelerometer measurements, $\boldsymbol{a}_l$, have been scaled into the interval $[-1, 1]$. The accelerometer measures no more than ± 2g. Using the accelerometer scale, s_a, and bias, b_a, the maximum and minimum accelerometer readings a_{max} and a_{min} are obtained using

$$\begin{aligned} a_{max} &= 2s_a + b_a, \\ a_{min} &= -2s_a + b_a. \end{aligned} \tag{3.6}$$

Every accelerometer measurement, $\boldsymbol{a}_l$, is then scaled into the interval $[-1, 1]$ using

$$\boldsymbol{a}_{norm,l} = 2 \cdot \frac{\boldsymbol{a}_l - a_{min}}{a_{max} - a_{min}} - 1, \tag{3.7}$$

where $\boldsymbol{a}_{norm,l}$ are the normalized measurements [28].

3.3.2 Structure and Training Parameters

To find a suitable architecture of the network a wide variety of structures were tested, from shallow to very deep networks, and using only a few neurons to many neurons in each layer. The architecture that achieved the best result had one hidden layer with 7500 neurons, which can be seen in Figure 3.2. Inputs to the network are the normalized accelerometer measurements, $\boldsymbol{a}_{norm,l}$, the targets during training are the ground truth accelerations $\boldsymbol{a}_{gt,l}$ and the outputs of the network are the calibrated accelerometer measurements, $\boldsymbol{a}_{c,l}$. Each neuron in the hidden layer is activated by the Leaky ReLU function with $\alpha = 0.1$.

In the training process the loss is calculated as the MSE($\boldsymbol{a}_{gt,l}, \boldsymbol{a}_{c,l}$) and to update the weights of the network the optimizer Adam is used. The training is executed until the training loss stops improving.

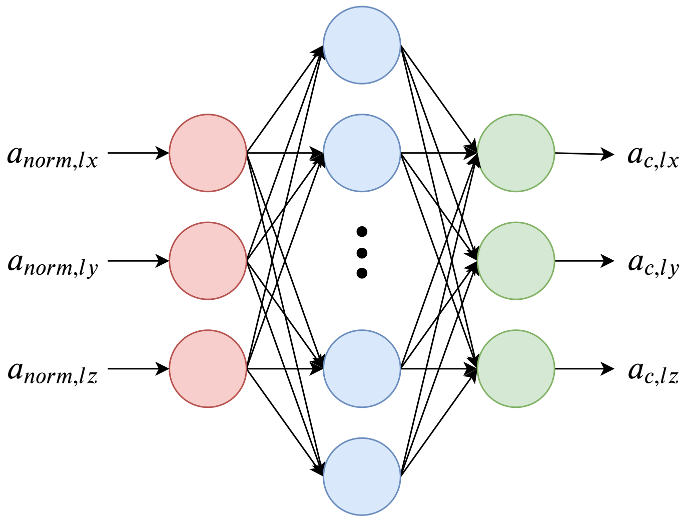


Figure 3.2: Architecture of the neural network used to calibrate the accelerometer. The input, $\boldsymbol{a}_{norm,l}$, to the network is fed into the input layer displayed in red. The blue neurons represents the hidden layer and the green neurons are the output layer. The output of the network is the calibrated accelerometer measurements, $\boldsymbol{a}_{c,l}$.

3.3.3 Using the Neural Network for Calibration

Once the training has terminated the network should have learned the characteristics of the sensor model and can hence be used to calibrate new measurements. Figure 3.3 shows an illustration of how the accelerometer measurements, $\boldsymbol{a}_l$, are preprocessed according to equation (3.7) and propagated through the network which outputs the calibrated measurements, $\boldsymbol{a}_{c,l}$.

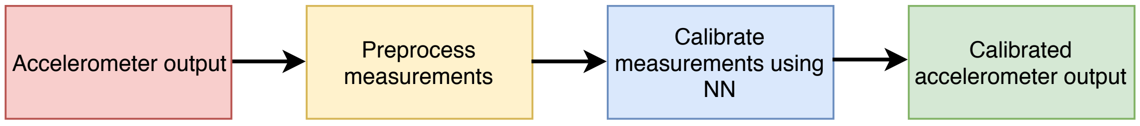

Accelerometer output
Preprocess measurements
Calibrate measurements using NN
Calibrated accelerometer output

Figure 3.3: Illustration of how the neural network is used for calibrating new accelerometer measurements after training.

3.4 Evaluation of Offline Calibration Algorithms

For validation of the developed offline calibration methods Dataset 1 was used. The dataset contains pairs of accelerometer measurement, $\boldsymbol{a}_l$, in Least Significant Bit (LSB) and true acceleration, $\boldsymbol{a}_{gt,l}$, in g for $l = 1, 2, ..., L$ and $L = 1386$, sampled from an accelerometer. All of the samples, l, represents a stationary orientation of the IMU. For each orientation the sensor has oversampled measurements and $\boldsymbol{a}_l$ is set to the average of the samples. The 1368 data points are divided into a training set of 1000 samples, a validation set of 184 samples and a test set of 184 samples. The ground truth, $\boldsymbol{a}_{gt,l}$, in the training set, i.e. the calibration orientations, are distributed as shown in Figure 3.4.

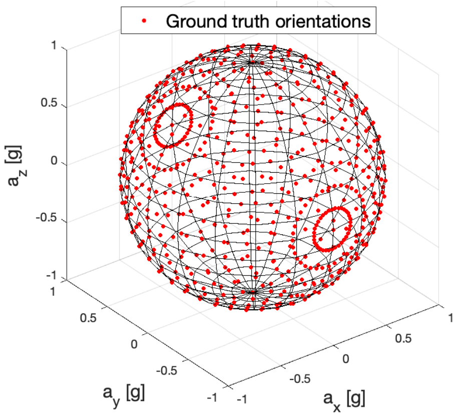


Figure 3.4: Distribution of the 1000 training samples. Each training sample generates a calibration orientation.

3.4.1 Result from Calibration using 18 Orientations

An accelerometer was calibrated by applying both of the offline algorithms to 18 training samples, taken from Dataset 1. In Figure 3.5, the distribution of the 18 calibration orientations are shown.

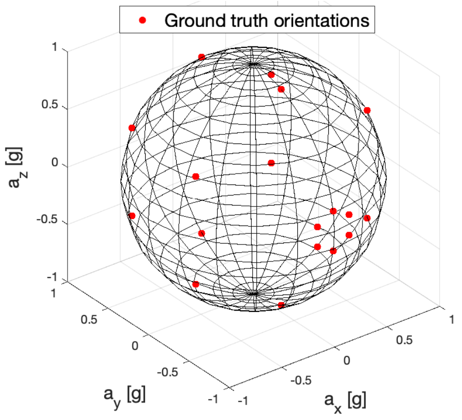


Figure 3.5: Distribution of the 18 training samples. Each training sample generates a calibration orientation.

After the calibration, the algorithms were evaluated on the test set, which contains 184 samples. A comparison of the calibrated measurements in the test set and corresponding ground truth is shown in Figure 3.6. In the figure ground truth is shown as red circles and the calibrated measurements as blue points. The left plot is the result after calibration using the linear sensor model algorithm and the one to the right is the result after calibration using the nonlinear neural network algorithm. The blue dots are centered inside the red circles to a greater extent for the linear sensor model.

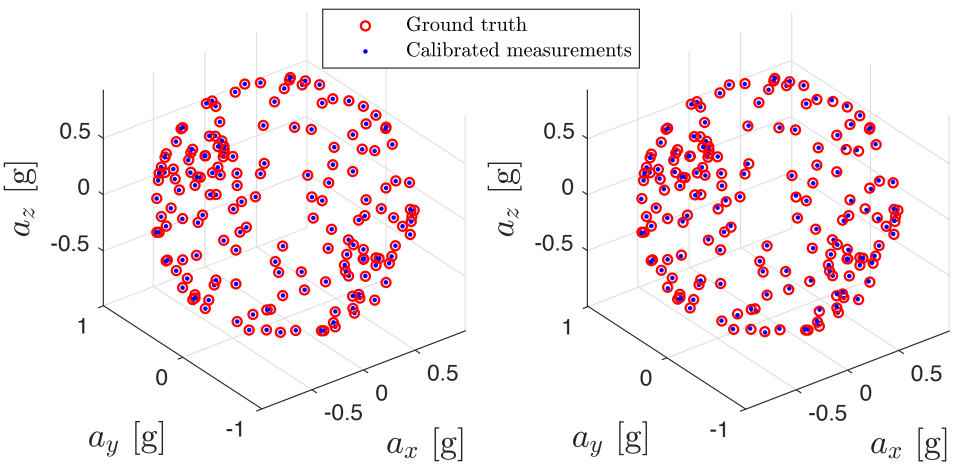


Figure 3.6: Plots of calibrated measurements, $\boldsymbol{a}_{c,l}$, and ground truth, $\boldsymbol{a}_{gt,l}$, in the test set. The plot to the left represents the linear sensor model algorithm and the one to the right represents the nonlinear neural network algorithm.

The error, e_l, between the ground truth, $\boldsymbol{a}_{gt,l} = [a_{gt,lx}, a_{gt,ly}, a_{gt,lz}]$, and the calibrated measurements, $\boldsymbol{a}_{c,l} = [a_{c,lx}, a_{c,ly}, a_{c,lz}]$, is for each sample of the test set calculated as the euclidean distance

$$e_l = \sqrt{(a_{gt,lx} - a_{c,lx})^2 + (a_{gt,ly} - a_{c,ly})^2 + (a_{gt,lz} - a_{c,lz})^2}. \tag{3.8}$$

A histogram containing the errors, e_l, is shown in Figure 3.7. In the upper histogram of the figure, which illustrates the result from using the linear sensor model algorithm, the distribution of the errors has both smaller mean and spread than the distribution of the errors in the lower histogram, which is the result of using the nonlinear neural network algorithm.

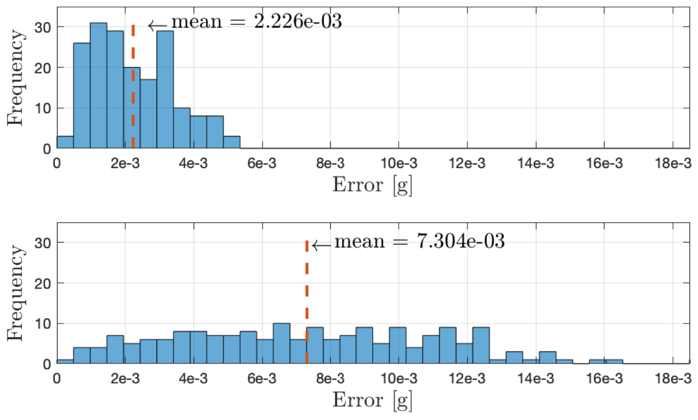


Figure 3.7: Distribution of e_l when evaluating in 184 test samples. The upper histogram is the result after calibration using linear sensor model algorithm and the lower is the result after calibration using the nonlinear neural network algorithm. The red dashed line represents the mean error.

3.4.2 Result from Calibration using 1000 Orientations

An accelerometer was also calibrated by applying both of the offline algorithms to all of the 1000 training samples in Dataset 1. After the calibration the algorithms were evaluated on the test set of 184 orientations. A comparison of the calibrated measurements in the test set and corresponding ground truth is shown in Figure 3.8. The left plot is the result after calibration using the linear sensor model algorithm and the one to the right is the result after calibration using the nonlinear neural network algorithm. The blue dots are centered inside the red circles for both models.

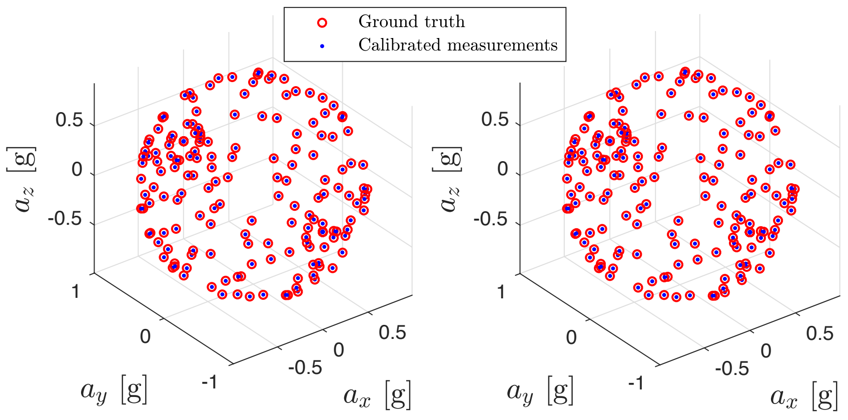


Figure 3.8: Plots of calibrated measurement, $\boldsymbol{a}_{c,l}$, and ground truth, $\boldsymbol{a}_{gt,l}$, in the test set. The plot to the left represents the linear sensor model algorithm and the one to the right represents the nonlinear neural network algorithm.

The errors, e_l, was again calculated according to equation (3.8). The distribution of e_l is visualized in Figure 3.9. In the upper histogram of the figure, which shows the error after the linear sensor model calibration, the mean error is larger relative the mean error in the lower histogram, which shows the error after the nonlinear neural network calibration.

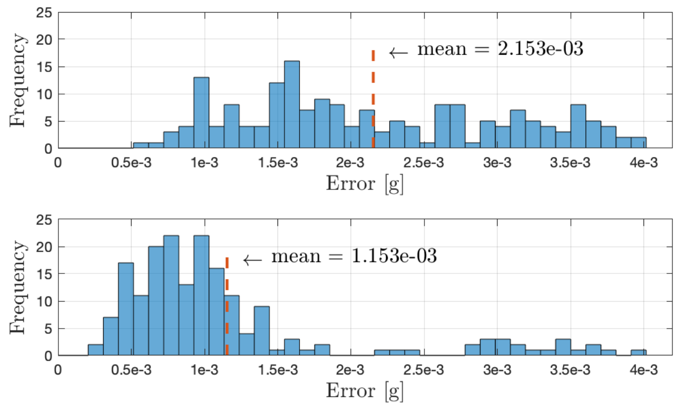


Figure 3.9: Distribution of e_l when evaluating in 184 test samples. The upper histogram is the result after calibration using the linear sensor model algorithm and the lower is the result after calibration using the nonlinear neural network algorithm. The red dashed line represents the mean error.

3.4.3 Result from Calibration using 18-1000 Orientations

To investigate the relation between the algorithms' performance and number of data points available for calibration, the algorithms were tested on different numbers of calibration samples between 18 and 1000, taken from the training set in Dataset 1. After calibration, the algorithms were tested on the test set of Dataset 1. The mean of the errors, e_l, calculated according to equation (3.8), is visualized as a function of the number of calibration samples in Figure 3.10.

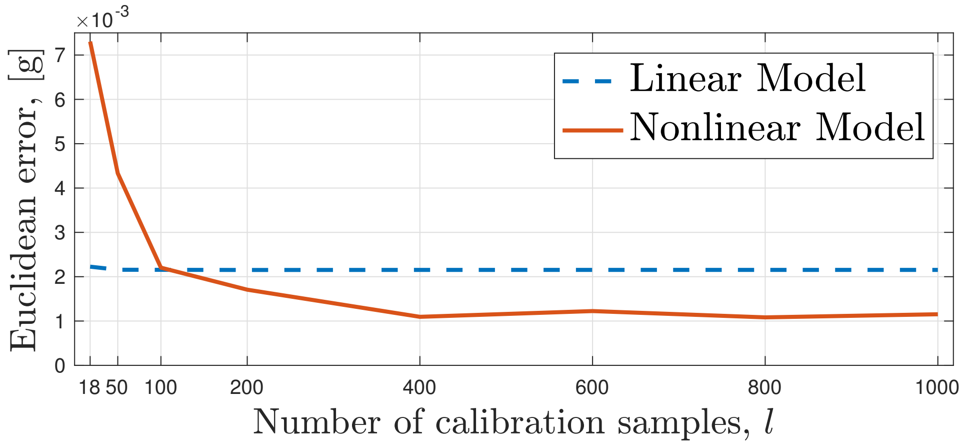


Figure 3.10: Mean error as a function of number of calibration samples, l.

The figure shows that the linear model has a constant mean error of about 2.2 mg, while the nonlinear model converges to a mean error of about 1.1 mg. Using more than 100 calibration orientations, the nonlinear neural network algorithm perform better than the linear sensor model algorithm. Table 3.1 summarises the mean error and error standard deviation after calibration with l samples. It can be seen from the table that the error standard deviation is smaller for the nonlinear model when calibration is performed using 400 data points or more.

Table 3.1: Mean error and error standard deviation after calibration with l samples.

	Mean $[mg]$		Standard deviation $[mg]$	
Calibration samples, l	Linear	Nonlinear	Linear	Nonlinear
18	2.226	7.304	1.175	3.832
50	2.157	4.332	0.933	2.561
100	2.155	2.204	0.897	1.063
200	2.152	1.707	0.911	1.032
400	2.153	1.095	0.905	0.880
600	2.154	1.224	0.901	0.860
800	2.153	1.083	0.905	0.839
1000	2.153	1.153	0.904	0.829

3.5 Discussion

Two different approaches for sensor model estimation have been implemented, one linear model and one nonlinear model approximated by a neural network. The result from the performed tests shows that the linear sensor model performs a better calibration when only a few data points are available. However, the neural network performs a better calibration when a sufficiently large amount of data points is available. As seen in Table 3.1 and Figure 3.10, 400 samples have proved to be sufficient in order for the neural network to generate a sensor model that halves the mean error against the benchmark model. Since the neural network has significantly more parameters than the linear model, it can fit better to the variations in the data. This seems to be beneficial when the algorithm is exposed to dense data. On the other hand, when only a few data samples are available, a thorough fit to the data might be disadvantageous and may result in overfitting. For a small dataset it is important that the model can interpolate between samples, which the linear model seems to be better at.

The result shows that the neural network can perform a better calibration than the linear model when a large dataset with a wide spread is available. Using a robot to position the accelerometer in 400 different stationary orientations or, if the algorithm is applied to a gyroscope, using a rate table to collect data for 400 angular rates, is expensive in the sense that it is time consuming. It also requires that the robot and rate table have high precision.

4 Online Calibration of Accelerometer and Gyroscope

This chapter describes the development and implementation of an online calibration method of both an accelerometer and a gyroscope. The algorithm is intended to calibrate both of the sensors simultaneously in field without use of any external equipment. The fact that the calibration can be utilized in field, enables the calibration to be continuously updated and to compensate for changes in the sensors' nature and the local environment. For implementation and evaluation of the online algorithm, a simulated dataset of accelerometer and gyroscope samples was created.

The idea behind the algorithm is to utilize the fact that, if an accelerometer and a gyroscope is calibrated perfectly, the measurements from the two sensors will be coherent with each other. Starting with a set of sensors which generates measurements that are not coherent, the algorithm alternately calibrates the gyroscope to fit the accelerometer data, assuming that the accelerometer is perfect, and calibrates the accelerometer to fit the gyroscope data, assuming that the gyroscope is perfect.

During evaluation of the offline calibration algorithms, the results showed that neural networks can accurately approximate the sensor model but at the cost of the need of a sufficiently large dataset. Using a large dataset is expensive since it is time consuming to collect and demand accurate precision of the external equipment used. Taken into account that the IMUs' characteristics change over time, which require a re-calibration or update of the current calibration, the gained accuracy from using neural networks in offline calibration may not be significantly valuable in a practical sense. However, if no external equipment or information are needed, and the data does not have to be collected in advance but can be collected during use, the disadvantages of using a neural network can be turned into advantages.

For the algorithm introduced in this chapter, neural networks are used to represent the sensor models and hence two neural networks are needed, one for calibration of the gyroscope (NN_φ) and one for calibration of the accelerometer (NN_a). The two networks are trained alternately until their outputs are coherent. Designing a proper loss function for the problem at hand is a crucial part of structuring a neural network. To be able to calibrate the sensors without any labeled training data, the loss can be based on how well the calibrated gyroscope and accelerometer estimates match each other. The developed loss functions are used to train the NN_a to compensate for the characteristics in the accelerometer measurements using the gyroscope as target, and to train the NN_φ to compensate for the characteristics in the gyroscope measurements using the accelerometer as target. An illustration of the introduced calibration algorithm, with a few steps explaining the overall structure, is shown in Figure 4.1.

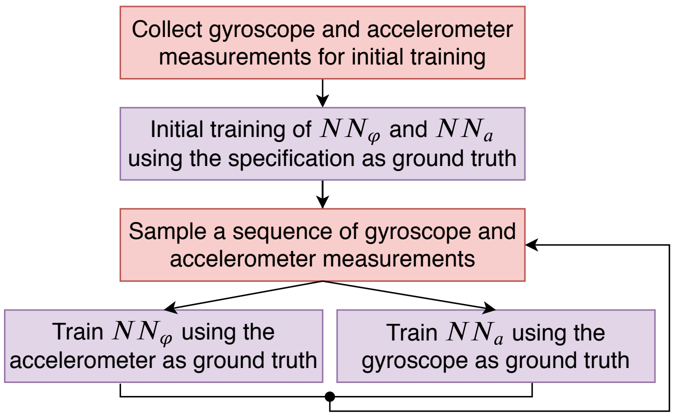

Collect gyroscope and accelerometer measurements for initial training
Initial training of NN_φ and NN_a using the specification as ground truth
Sample a sequence of gyroscope and accelerometer measurements
Train NN_φ using the accelerometer as ground truth
Train NN_a using the gyroscope as ground truth

Figure 4.1: Overview of the structure of the online adaptive calibration algorithm.

4.1 Calibration Assumptions

The gyroscope has characteristics that are known to be dependent of time and temperature. However, for short periods of time, such as 1-2s, the time-variant and temperature-variant drift can be neglected. During calibration, the IMU is only rotated around its own center and hence the accelerometer will not measure any linear acceleration except for the gravitational acceleration. Between two samples, n and $n+1$, the angular velocity is constant. The IMUs' orientation of sample $n+1$ can hence be estimated from the orientation of sample n using Euler forward method.

Each sample of gyroscope measurements contains an angular rate in x, y and z, i.e. at time sample n the gyroscope outputs the vector $\boldsymbol{\varphi}_n = [\varphi_{nx}, \varphi_{ny}, \varphi_{nz}]^T$. The sensors operate at a frequency f and the sample period between two consecutive discrete samples is $dt = 1/f$. The infinitesimal rotation of the sensor can be calculated as

$$\boldsymbol{\theta}_n = \boldsymbol{\varphi}_n \cdot dt, \tag{4.1}$$

where $\boldsymbol{\varphi}_n$ is the angular rate measured by the gyroscope at sample n and $\boldsymbol{\theta}_n$ is the angle of which the sensor has rotated between sample n and sample $n+1$.

The rotation matrix, $\boldsymbol{R}_n = \boldsymbol{R}(\boldsymbol{\varphi}_n, dt)$, that represents the rotation $\boldsymbol{\theta}_n$, can be expressed through the axis-angle representation which states that any 3D rotation can be represented by an axis, $\boldsymbol{v}_n$, and an angle, Θ_n [29]. The axis-angle representation is calculated as

$$\begin{aligned}\boldsymbol{v}_n &= \begin{bmatrix} v_{nx} \\ v_{ny} \\ v_{nz} \end{bmatrix} = \frac{1}{\sqrt{\theta_{nx}^2 + \theta_{ny}^2 + \theta_{nz}^2}} \begin{bmatrix} \theta_{nx} \\ \theta_{ny} \\ \theta_{nz} \end{bmatrix}, \\ \Theta_n &= \sqrt{\theta_{nx}^2 + \theta_{ny}^2 + \theta_{nz}^2},\end{aligned} \tag{4.2}$$

and yields the rotation matrix

$$\boldsymbol{R}_n = \begin{bmatrix} c + v_{nx}^2(1-c) & v_{nx}v_{ny}(1-c) - v_{nz}s & v_{nx}v_{nz}(1-c) + v_{ny}s \\ v_{nx}v_{ny}(1-c) + v_{nz}s & c + v_{ny}^2(1-c) & v_{ny}v_{nz}(1-c) - v_{nx}s \\ v_{nx}v_{nz}(1-c) - v_{ny}s & v_{nz}v_{ny}(1-c) + v_{nx}s & c + v_{nz}^2(1-c) \end{bmatrix} \tag{4.3}$$

where $c = \cos(\Theta_n)$ and $s = \sin(\Theta_n)$ [30].

4.2 Dataset 2 - Simulated Dataset

When simulating this dataset the concept of sequences was introduced. One sequence is defined as the measurements sampled by an IMU that is rotated continuously between two stationary orientations. The dataset contains a training set of 500 sequences and a validation set of 100 sequences with starting and finish point distributed over the whole sphere. Each sequence consists of 101 samples, where the first and last sample represent stationary conditions. The accelerometer and gyroscope measurements are sampled simultaneously at a frequency of 100 Hz. Figure 4.2 shows the distribution of the 500 training sequences of accelerometer samples.

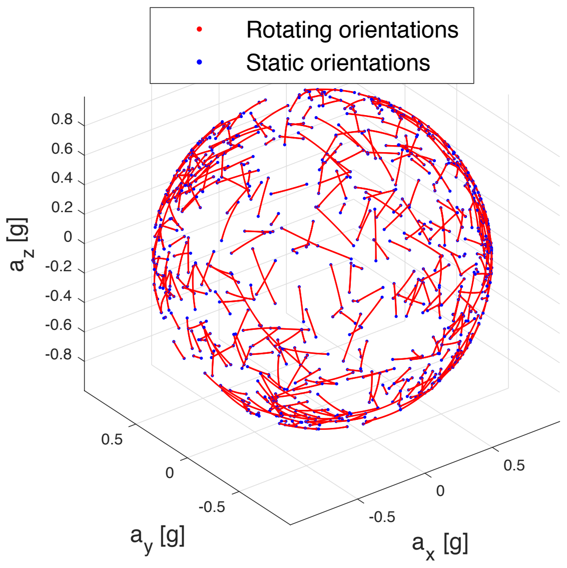


Figure 4.2: Distribution of the 500 training sequences used in calibration. The blue points represent the static orientations at the start and end of the sequence and the red points are the rotating orientations.

The true angular velocity and the true acceleration is denoted $\boldsymbol{\varphi}_{gt,n}^{m}$ and $\boldsymbol{a}_{gt,n}^{m}$ respectively, where the subscript gt is ground truth, $m \in [1, M]$, M is the number of sequences in the dataset, $n \in [1, N]$ and N is the number of samples in one sequence. To simulate a sequence of accelerometer and gyroscope measurements, the $\boldsymbol{\varphi}_{gt,n}^{m}$, are modeled as a quadratic function

$$\varphi_k(t) = 4\varphi_{max}(-t^2 + t), \quad t \in [0, 1], \tag{4.4}$$

where the subscript $k = \{x, y, z\}$ and φ_{max} represents the highest angular velocity reached in each sequence. For each axis, φ_{max} was randomly chosen in the interval $[-0.35, 0.35]$ rad/s. In Figure 4.3 the true angular velocity is visualized. If φ_{max} is negative the curve in the figure will be inverted.

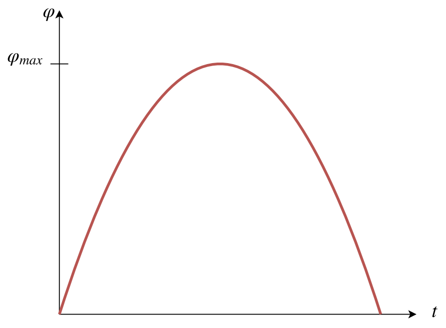


Figure 4.3: Quadratic function representing the true angular velocity, $\boldsymbol{\varphi}^m_{gt,n}$, in each rotation. If φ_{max} is a negative value the curve will be inverted.

From $\boldsymbol{\varphi}^m_{gt,n}$ a rotation matrix, $\boldsymbol{R}^m_n = \boldsymbol{R}(\boldsymbol{\varphi}^m_{gt,n}, dt)$, is calculated according to equation (4.1)-(4.3). Starting with an initial orientation the following true accelerations, $\boldsymbol{a}^m_{gt,n}$, are created using

$$\boldsymbol{a}^m_{gt,n+1} = \boldsymbol{R}^m_n \boldsymbol{a}^m_{gt,n}, \tag{4.5}$$

where the current acceleration frame, $\boldsymbol{a}^m_{gt,n}$ is rotated with $\boldsymbol{R}^m_n$ into the next acceleration frame, $\boldsymbol{a}^m_{gt,n+1}$.

From the true angular rate, $\boldsymbol{\varphi}^m_{gt,n}$, in rad/s and the true specific force, $\boldsymbol{a}^m_{gt,n}$, in g, measurements, $\boldsymbol{\varphi}^m_n$ and $\boldsymbol{a}^m_n$, in LSB values are constructed by element-wise multiplication of scale, $\boldsymbol{s}$, and addition of bias, $\boldsymbol{b}$,

$$\begin{aligned} \boldsymbol{\varphi}^m_n &= \boldsymbol{S} \odot \boldsymbol{\varphi}^m_{gt,n} + \boldsymbol{b}, \\ \boldsymbol{a}^m_n &= \boldsymbol{S} \odot \boldsymbol{a}^m_{gt,n} + \boldsymbol{b}, \end{aligned} \tag{4.6}$$

where $\odot$ denotes the element-wise multiplication. To resemble the output in LSB of a real sensor, the measurements, $\boldsymbol{\varphi}^m_n$ and $\boldsymbol{a}^m_n$, are rounded to integers.

A test set was also created, containing one set for the accelerometer, and another set for the gyroscope. The test set for the accelerometer consists of ground truth, $\boldsymbol{a}^{test}_{gt,n}$, in g and accelerometer measurements, $\boldsymbol{a}^{test}_n$, in LSB in 500 stationary orientations. The ground truth samples are distributed over the whole sphere and the measurements are then created according to equation (4.6). In the test set for the gyroscope, which consists of 500 samples of ground truth, $\boldsymbol{\varphi}^{test}_{gt,n}$ in rad/s and gyroscope measurements, $\boldsymbol{\varphi}^{test}_n$ in LSB, the ground truth is simulated as

$$\varphi_k(t) = 0.35 \sin(t), \quad t \in [0, 2\pi], \tag{4.7}$$

where $k = \{x, y, z\}$. Figure 4.4 shows the true angular velocity simulated in the test set. The gyroscope measurements, $\boldsymbol{\varphi}^{test}_n$, are then created according to equation (4.6).

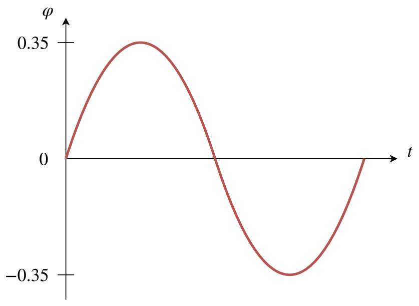


Figure 4.4: Sinusoidal function representing the true angular velocity, $\boldsymbol{\varphi}_{gt,n}^{test}$, in the test set.

4.3 Initial Training

In the first part of the algorithm, an initial training of both the (NN_a) and the (NN_φ) is executed. The purpose of the initial training is to help the networks converge in a later stage of the algorithm. In the initial training phase the measurements, $\boldsymbol{\varphi}_n^m$ and $\boldsymbol{a}_n^m$, are converted from LSB values to angular rates and accelerations using the scale and bias specified in the sensors' specification. The measurements according to the specification are denoted $\boldsymbol{\varphi}_{s,n}^m$ and $\boldsymbol{a}_{s,n}^m$. Knowing the scale, s, and bias, b, $\boldsymbol{\varphi}_n^m$ and $\boldsymbol{a}_n^m$ is converted using

$$\begin{aligned} \boldsymbol{\varphi}_{s,n}^m &= \frac{\boldsymbol{\varphi}_n^m - b_g}{s_g}, \\ \boldsymbol{a}_{s,n}^m &= \frac{\boldsymbol{a}_n^m - b_a}{s_a}, \end{aligned} \tag{4.8}$$

where the sub indices g and a denote the gyroscope and the accelerometer respectively. In the initial training, $\boldsymbol{\varphi}_{s,n}^m$ and $\boldsymbol{a}_{s,n}^m$ are used as targets.

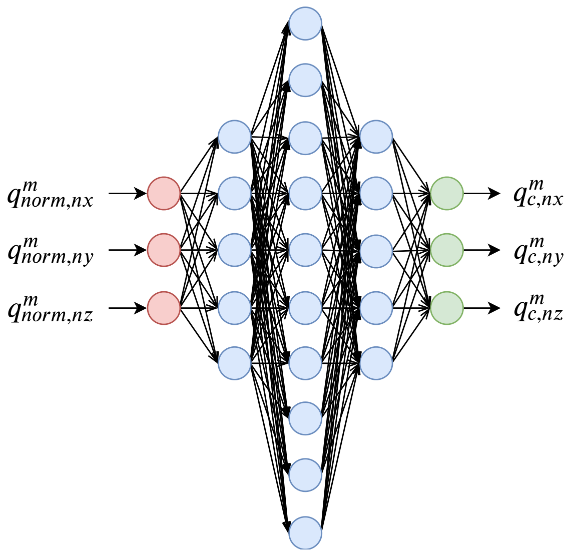


Figure 4.5: Structure of the NN_φ and the NN_a in the online adaptive calibration algorithm. The vector $\boldsymbol{q}$ represents $\boldsymbol{a}$ or $\boldsymbol{\varphi}$.

Both the NN_a and the NN_φ use the architecture shown in Figure 4.5. In the figure $\boldsymbol{q}$ is used to denote the considered quantity, i.e can be replaced with $\boldsymbol{a}$ or $\boldsymbol{\varphi}$. The network has three hidden layers with five neurons in the first and third hidden layer and ten neurons in the second hidden layer. The activation function used is the Leaky ReLU function with α set to 0.1. In the training process, the loss is calculated as the MSE($\boldsymbol{q}^m_{s,n}$,$\boldsymbol{q}^m_{c,n}$) and the weights of the network are updated using the optimizer Adam.

4.4 Adaptive Gyroscope Calibration

The adaptive calibration of the gyroscope is designed to adjust the gyroscope measurements to fit the accelerometer data. The part of the adaptive calibration algorithm that adjusts the sensor model for the gyroscope is illustrated in a block diagram in Figure 4.6, where the first block represents the initial training. After the initial training the algorithm loops between collecting a new sequence of data and using the data to update the calibration model. Each new sequence, m, of data consists of gyroscope measurements, $\boldsymbol{\varphi}^m_n$, and accelerometer measurements, $\boldsymbol{a}^m_n$.

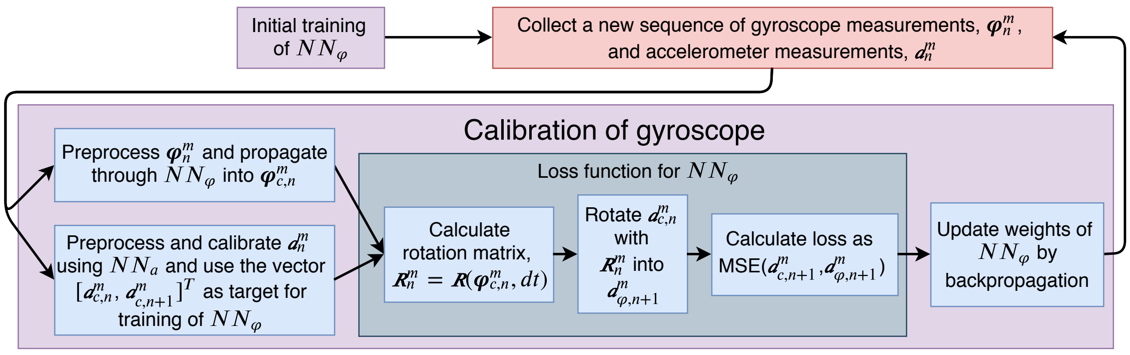

Initial training of NN_φ
Collect a new sequence of gyroscope measurements, $\boldsymbol{\varphi}^m_n$, and accelerometer measurements, $\boldsymbol{a}^m_n$
Calibration of gyroscope
Preprocess $\boldsymbol{\varphi}^m_n$ and propagate through NN_φ into $\boldsymbol{\varphi}^m_{c,n}$
Preprocess and calibrate $\boldsymbol{a}^m_n$ using NN_a and use the vector $[\boldsymbol{a}^m_{c,n}, \boldsymbol{a}^m_{c,n+1}]^T$ as target for training of NN_φ
Loss function for NN_φ
Calculate rotation matrix, $\boldsymbol{R}^m_n = \boldsymbol{R}(\boldsymbol{\varphi}^m_{c,n}, dt)$
Rotate $\boldsymbol{a}^m_{c,n}$ with $\boldsymbol{R}^m_n$ into $\boldsymbol{a}^m_{\varphi,n+1}$
Calculate loss as MSE($\boldsymbol{a}^m_{c,n+1}$,$\boldsymbol{a}^m_{\varphi,n+1}$)
Update weights of NN_φ by backpropagation

Figure 4.6: Structure of the adaptive gyroscope calibration where the accelerometer is assumed to be perfect.

The gyroscope measurement, $\boldsymbol{\varphi}^m_n$, is normalized using equation (3.7) into $\boldsymbol{\varphi}^m_{norm,n}$ and is given as input to the NN_φ during training. The corresponding accelerometer measurement, $\boldsymbol{a}^m_n$, and the next accelerometer measurement, $\boldsymbol{a}^m_{n+1}$, is calibrated using the NN_a into $\boldsymbol{a}^m_{c,n}$ and $\boldsymbol{a}^m_{c,n+1}$ respectively. The vector $[\boldsymbol{a}^m_{c,n}, \boldsymbol{a}^m_{c,n+1}]^T$ is given as target to the NN_φ. After propagation of $\boldsymbol{\varphi}^m_{norm,n}$ through the NN_φ, the calibrated angular rates, $\boldsymbol{\varphi}^m_{c,n}$, are obtained. From $\boldsymbol{\varphi}^m_{c,n}$ the rotation matrix, $\boldsymbol{R}^m_n = \boldsymbol{R}(\boldsymbol{\varphi}^m_{c,n}, dt)$, is obtained using equation (4.1)-(4.3). The rotation matrix is used to rotate $\boldsymbol{a}^m_{c,n}$ into $\boldsymbol{a}^m_{\varphi,n+1}$ as

$$\boldsymbol{a}^m_{\varphi,n+1} = \boldsymbol{R}^m_n \boldsymbol{a}^m_{c,n}. \tag{4.9}$$

The loss, $\mathcal{L}$, is calculated as the mean squared error

$$\mathcal{L} = \frac{1}{3} \sum_{k=\{x,y,z\}} (a^m_{c,(n+1)k} - a^m_{\varphi,(n+1)k})^2. \tag{4.10}$$

An example in $\mathbb{R}^2$ of how $\boldsymbol{a}^m_{\varphi,n+1}$ and $\boldsymbol{a}^m_{c,n+1}$ could be related is shown in Figure 4.7. The figure shows three different coordinate frames, $\{T\}$, in which the gravitational acceleration, $\boldsymbol{a}$, is measured. The subscript of T denotes to which measurement each coordinate frame belongs. The blue frame, $\{T^m_{\varphi,n+1}\}$, is the result from rotation of $\{T^m_{c,n}\}$ using $\boldsymbol{R}^m_n$, i.e the orientation of the sensor coordinate frame at sample $n+1$ according to the gyroscope. The dashed black frame, $\{T^m_{c,n+1}\}$ is the orientation of the sensor coordinate frame at sample $n+1$ according to the accelerometer.

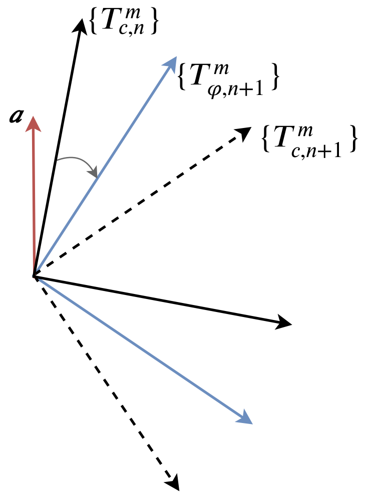


Figure 4.7: The gravitational acceleration, $\boldsymbol{a}$, measured in three coordinate frames, $\{T^m_{c,n}\}$, $\{T^m_{c,n+1}\}$ and $\{T^m_{\varphi,n+1}\}$.

In Figure 4.8, the coordinate frames have been separated and aligned with each other and the direction of the gravitational acceleration is illustrated for each separate frame. From the image, the loss can be seen as the MSE$(\boldsymbol{a}^m_{c,n+1}, \boldsymbol{a}^m_{\varphi,n+1})$ according to equation (4.10).

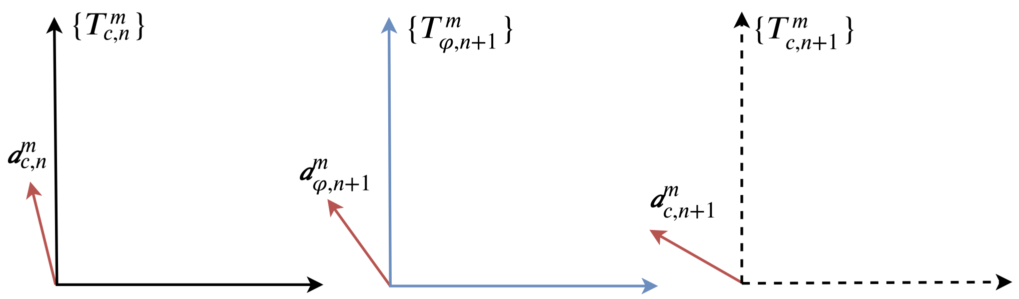


Figure 4.8: The three coordinate frames $\{T^m_{c,n}\}$, $\{T^m_{c,n+1}\}$ and $\{T^m_{\varphi,n+1}\}$ separated.

4.5 Adaptive Accelerometer Calibration

The adaptive calibration of the accelerometer is designed to adjust the accelerometer measurements to fit the gyroscope data. The part of the adaptive calibration algorithm that adjusts the sensor model for the accelerometer is illustrated in a block diagram in Figure 4.9, where the first block represents the initial training. After the initial training the algorithm loops between collecting a new sequence of data and using the data to update the NN_a.

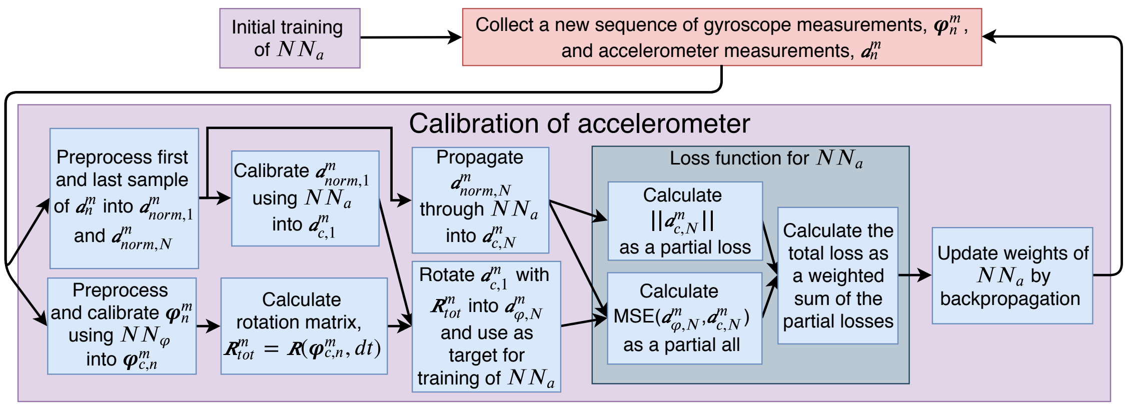

Initial training of NN_a
Collect a new sequence of gyroscope measurements, $\boldsymbol{\varphi}_n^m$, and accelerometer measurements, $\boldsymbol{a}_n^m$
Calibration of accelerometer
Preprocess first and last sample of $\boldsymbol{a}_n^m$ into $\boldsymbol{a}_{norm,1}^m$ and $\boldsymbol{a}_{norm,N}^m$
Calibrate $\boldsymbol{a}_{norm,1}^m$ using NN_a into $\boldsymbol{a}_{c,1}^m$
Propagate $\boldsymbol{a}_{norm,N}^m$ through NN_a into $\boldsymbol{a}_{c,N}^m$
Preprocess and calibrate $\boldsymbol{\varphi}_n^m$ using NN_φ into $\boldsymbol{\varphi}_{c,n}^m$
Calculate rotation matrix, $\boldsymbol{R}_{tot}^m = \boldsymbol{R}(\boldsymbol{\varphi}_{c,n}^m, dt)$
Rotate $\boldsymbol{a}_{c,1}^m$ with $\boldsymbol{R}_{tot}^m$ into $\boldsymbol{a}_{\varphi,N}^m$ and use as target for training of NN_a
Loss function for NN_a
Calculate $||\boldsymbol{a}_{c,N}^m||$ as a partial loss
Calculate MSE($\boldsymbol{a}_{\varphi,N}^m$,$\boldsymbol{a}_{c,N}^m$) as a partial all
Calculate the total loss as a weighted sum of the partial losses
Update weights of NN_a by backpropagation

Figure 4.9: Structure of the adaptive accelerometer calibration where the gyroscope is assumed to be perfect.

When training the NN_a to perform a calibration of the accelerometer the loss is similar as for the NN_φ but, instead of feeding the network with all measurements, only the first and last sample of every sequence of accelerometer measurements are considered. The first and last measurement represent stationary orientations.

The two stationary accelerometer measurements of a sequence, m, are called $\boldsymbol{a}_1^m$ and $\boldsymbol{a}_N^m$. Using NN_a, $\boldsymbol{a}_{norm,1}^m$ is calibrated and called $\boldsymbol{a}_{c,1}^m$. The associated sequence of calibrated gyroscope measurements, $\boldsymbol{\varphi}_{c,n}^m$, are used to obtain a sequence of infinitesimal angles $\boldsymbol{\theta}_n^m$ using equation (4.1). For each angle, $\boldsymbol{\theta}_n^m$, the rotation matrix, $\boldsymbol{R}_n^m$, is calculated according to equation (4.3). A rotation matrix, $\boldsymbol{R}_{tot}^m$, representing the accumulated rotations from the sequence, is obtained by multiplication of all $\boldsymbol{R}_n^m$ as

$$\boldsymbol{R}_{tot}^m = \prod_{n=1}^{N} \boldsymbol{R}_n^m. \tag{4.11}$$

The first calibrated stationary orientation, $\boldsymbol{a}_{c,1}^m$, is then rotated with $\boldsymbol{R}_{tot}^m$ into $\boldsymbol{a}_{\varphi,N}^m$ as

$$\boldsymbol{a}_{\varphi,N}^m = \boldsymbol{R}_{tot}\boldsymbol{a}_{c,1}^m. \tag{4.12}$$

During training of the NN_a, $\boldsymbol{a}_N^m$ is normalized using (3.7) into $\boldsymbol{a}_{norm,N}^m$ and given as input to the network NN_a. As target $\boldsymbol{a}_{\varphi,N}^m$ is given. The loss is then calculated partly from the mean squared error between the predicted input, $\boldsymbol{a}_{c,N}^m$, and the target, $\boldsymbol{a}_{\varphi,N}^m$, and partly from the magnitude of the predicted input vector, $\boldsymbol{a}_{c,N}^m$, as

$$\begin{aligned}
\mathcal{L}_1 &= \frac{1}{3}\sum_{k=\{x,y,z\}} (a_{\varphi,Nk}^m - a_{c,Nk}^m)^2, \\
\mathcal{L}_2 &= abs\left(1 - ||\boldsymbol{a}_{c,N}^m||\right), \\
\mathcal{L} &= w \cdot \mathcal{L}_1 + (1-w)\cdot \mathcal{L}_2,
\end{aligned} \tag{4.13}$$

where w is a term deciding on how the two partial losses will be weighted. Different weightings were tested but best performance was obtained using $w = 0.90$.

An example in $\mathbb{R}^2$ of how the predicted input, $\boldsymbol{a}^m_{c,N}$, and the target, $\boldsymbol{a}^m_{\varphi,N}$, can be related is shown in Figure 4.10. The figure shows three different coordinate frames in which the gravitational acceleration, $\boldsymbol{a}$, is measured. The blue frame $\{T^m_{\varphi,N}\}$ is the result from rotation of $\{T^m_{c,1}\}$ using $\boldsymbol{R}^m_{tot}$, i.e the orientation of the sensor coordinate frame at sample N according to the gyroscope. The dashed black frame $T^m_{c,N}$ is the orientation of the sensor coordinate frame at sample N according to the accelerometer.

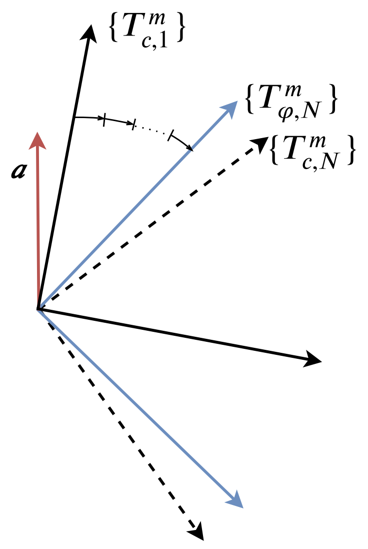


Figure 4.10: Three coordinate frames, $\{T^m_{c,1}\}$, $\{T^m_{c,N}\}$ and $\{T^m_{\varphi,N}\}$ in which the gravitational acceleration, $\boldsymbol{a}$, is measured.

4.6 Evaluation of Online Calibration Algorithm

For validation of the online calibration method, the simulated dataset, Dataset 2, was used. The algorithm was evaluated both during and after the calibration.

4.6.1 Result During Calibration

Every tenth sequence in the training set was translated according to the sensor specification and used for initial training of NN_φ and NN_a. All 500 sequences were then, one at a time, processed by the algorithm to update the calibration. After the initial training, the model was evaluated on the test set both for the accelerometer and the gyroscope. The result for the accelerometer is shown in Figure 4.11, where the red circles represent the ground truth, $\boldsymbol{a}^{test}_{gt,n}$, and the blue points represent the calibrated measurements, $\boldsymbol{a}^{test}_{c,n}$. In the figure, all of the samples are outside the red circles.

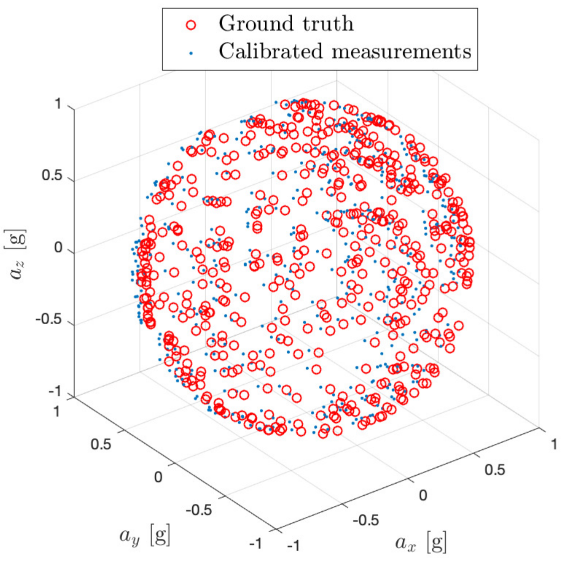


Figure 4.11: Plot of the ground truth, $\boldsymbol{a}_{gt,n}^{test}$, and the calibrated accelerometer measurements, $\boldsymbol{a}_{c,n}^{test}$, in the test set after initial training.

The result for the gyroscope is shown in Figure 4.12, where the blue line represents the ground truth, $\boldsymbol{\varphi}_{gt,n}^{test}$, and the red line represents the calibrated gyroscope measurements, $\boldsymbol{\varphi}_{c,n}^{test}$.

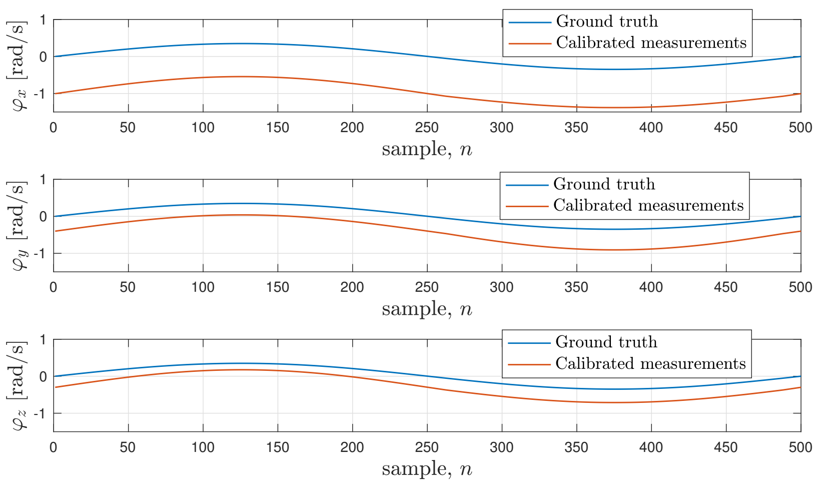


Figure 4.12: Plot of the ground truth, $\boldsymbol{\varphi}_{gt,n}^{test}$, and the calibrated gyroscope measurements, $\boldsymbol{\varphi}_{c,n}^{test}$, in the test set after initial training.

To show how the result changes during the training the model was also evaluated after each iteration of the calibration loop on the $M = 100$ validation sequences, each containing $N = 101$ samples. During each evaluation, i, the calibrated measurements were compared to ground truth and the errors, $e_{g,i}$ and $e_{a,i}$, were calculated according to

$$e_{g,i} = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\sqrt{(\varphi_{c,nx}^{m}-\varphi_{gt,nx}^{m})^2+(\varphi_{c,ny}^{m}-\varphi_{gt,ny}^{m})^2+(\varphi_{c,nz}^{m}-\varphi_{gt,nz}^{m})^2}$$

and (4.14)

$$e_{a,i} = \frac{1}{M}\sum_{m=1}^{M}\sqrt{(a_{c,Nx}^{m}-a_{gt,Nx}^{m})^2+(a_{c,Ny}^{m}-a_{gt,Ny}^{m})^2+(a_{c,Nz}^{m}-a_{gt,Nz}^{m})^2},$$

where $\boldsymbol{\varphi}_{c,n}^{m}$ and $\boldsymbol{a}_{c,N}^{m}$ are the calibrated measurements, $\boldsymbol{\varphi}_{gt,n}^{m}$ and $\boldsymbol{a}_{gt,N}^{m}$ are the ground truth.

Figure 4.13 presents the result from evaluation of the algorithm on the validation set during training. In the left plot of the figure $e_{g,i}$ is presented and in the right plot of the figure $e_{a,i}$ is presented. The x-axis represents evaluation, i, where $i = 0$ denotes the algorithm's state after the initial training and $i \in [1, 500]$ denotes the algorithm's state in the loop phase.

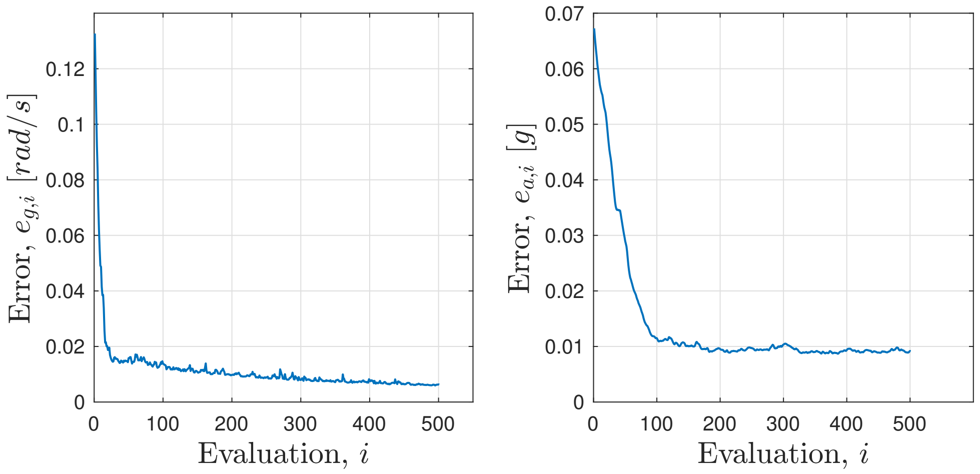


Figure 4.13: Error during training when evaluating both of the sensors on the validation set. The left plot represents the gyroscope and the right plot represents the accelerometer.

In Figure 4.14, $e_{g,i}$ and $e_{a,i}$ are compared to each other. The plots shows faster and more stable convergence of the accelerometer. The accelerometer converges to an error of about 10 mg while the gyroscope continues improving over all iterations and has an error of about 6 $mrad/s$ after the last evaluation.

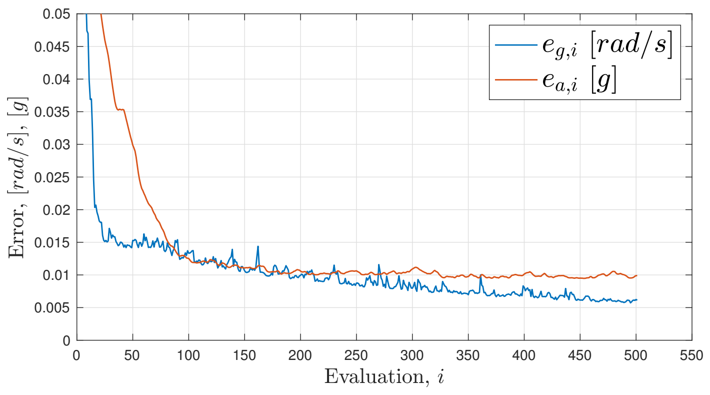


Figure 4.14: Comparison of the gyroscope and the accelerometer convergence when calibrating with 500 sequences.

4.6.2 Result After Calibration

After the calibration process the performance of the gyroscope and the accelerometer were evaluated on the test sets. In Figure 4.15 the red circles represent the ground truth, $\boldsymbol{a}_{gt,n}^{test}$, of the accelerometer and the blue points represent the calibrated measurements, $\boldsymbol{a}_{c,n}^{test}$. In the figure, apart from a few exceptions, most calibrated measurements fit the ground truth.

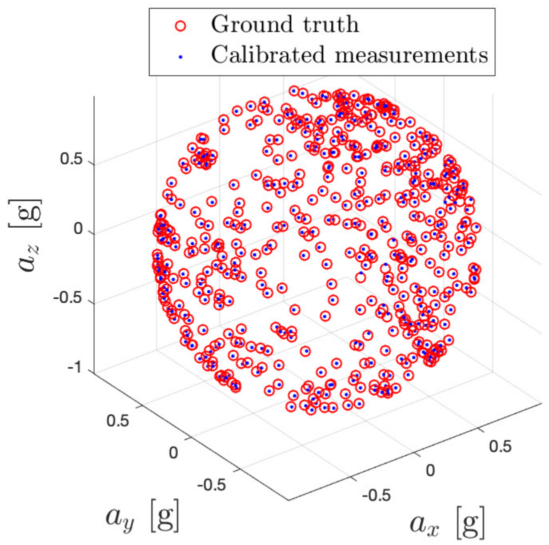


Figure 4.15: Plot of the ground truth, $\boldsymbol{a}_{gt,n}^{test}$, and the calibrated accelerometer measurements, $\boldsymbol{a}_{c,n}^{test}$, in the test set.

The errors, $e_{a,n}$, between $\boldsymbol{a}_{gt,n}^{test}$ and $\boldsymbol{a}_{c,n}^{test}$, were in each sample of the test set calculated according to equation (3.8). Figure 4.16 shows the distribution of $e_{a,n}$ and the mean error and error standard deviation are shown in Table 4.1. In the figure most residuals are smaller than 12 mg.

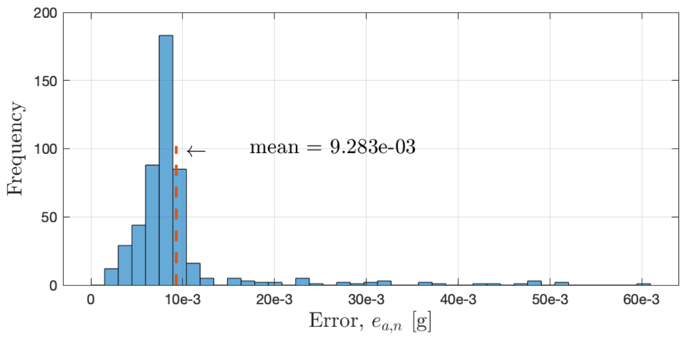


Figure 4.16: Distribution of $e_{a,n}$ when evaluating the accelerometer on the test set.

Table 4.1: Mean error and error standard deviation when evaluating the accelerometer on the test set.

Mean $[mg]$	Standard deviation $[mg]$
9.283	7.133

Evaluating the gyroscope on the test set gives the result in Figure 4.17, where the blue line represents the ground truth, $\boldsymbol{\varphi}_{gt,n}^{test}$, and the red line is the calibrated gyroscope measurements, $\boldsymbol{\varphi}_{c,n}^{test}$. The calibrated measurements of sample $80 - 160$, highlighted with a black box, has lower accuracy than the remaining measurements.

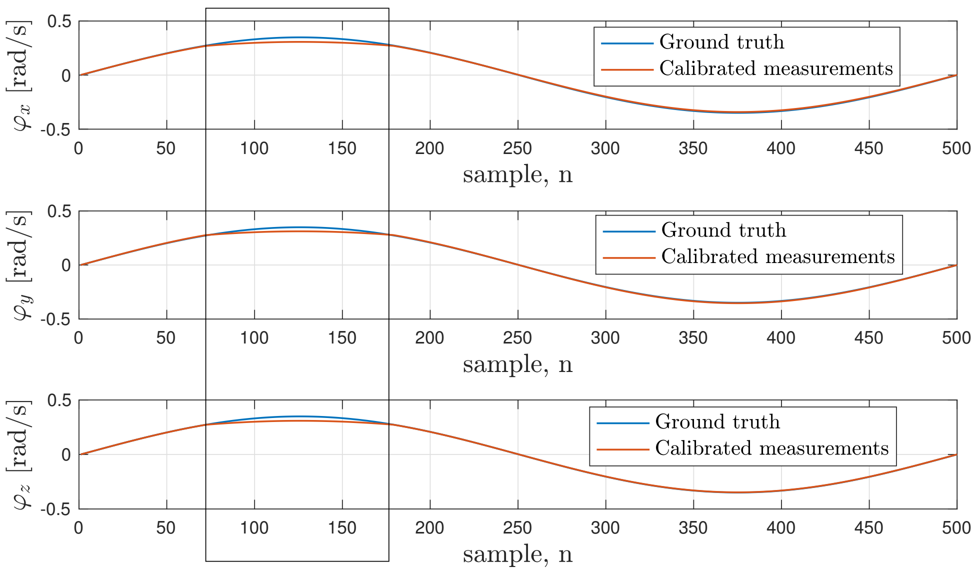


Figure 4.17: Plot of the ground truth, $\boldsymbol{\varphi}_{gt,n}^{test}$, and the calibrated gyroscope measurements, $\boldsymbol{\varphi}_{c,n}^{test}$, in the test set.

The errors, $e_{g,n}$, between $\boldsymbol{\varphi}_{gt,n}^{test}$ and $\boldsymbol{\varphi}_{c,n}^{test}$ were again, in each sample of the test set, calculated according to the method in equation (3.8), but with $\boldsymbol{a}_{gt,n}^{test}$ shifted to

$\boldsymbol{\varphi}_{gt,n}^{test}$ and $\boldsymbol{a}_{c,n}^{test}$ shifted to $\boldsymbol{\varphi}_{c,n}^{test}$. Figure 4.18 shows the distribution of the $e_{g,n}$ and the mean error and error standard deviation are shown in Table 4.2. The figure shows that most residuals are smaller than 10 $mrad/s$ but the mean is significantly larger.

Table 4.2: Mean error and error standard deviation when evaluating the gyroscope on the test set.

Mean $[mrad/s]$	Standard deviation $[mrad/s]$
13.62	19.55

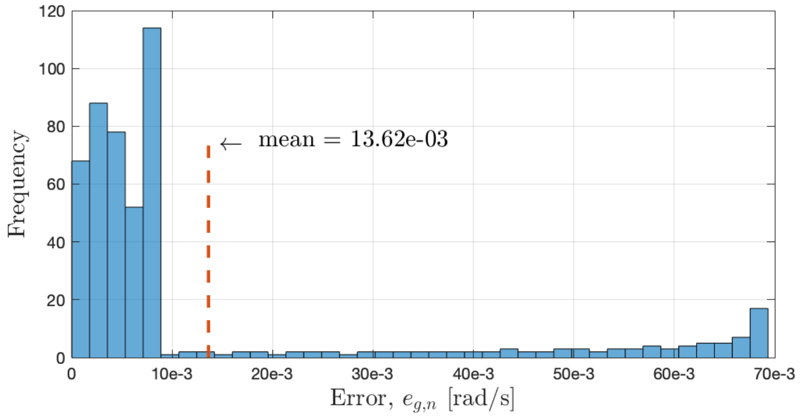


Figure 4.18: Distribution of $e_{g,n}$ when evaluating the gyroscope on the test set.

4.7 Discussion

In this section the introduced adaptive calibration algorithm is discussed. The discussion includes an analysis of the result from the evaluation of the algorithm, and possible underlying reasons for the outcome. Also the practical usefulness of the algorithm is considered.

4.7.1 Convergence of Adaptive Algorithm

When applied to the simulated data, the adaptive algorithm can perform a calibration of the accelerometer and the gyroscope, only using information retrieved from the IMU considered. The result displayed in Figure 4.14 shows that the calibration error for the accelerometer improves in a smoother way than the gyroscope calibration, which to a greater extent tend to fluctuate. One reason for this disparity between the characteristics of the convergence curves could be related to the loss functions. The loss function for the calibration of the accelerometer has one more constraint than the one for calibration of the gyroscope, namely that the norm of the output must be 1 g. It is also possible that a smoother convergence could be obtained by further tuning of the training parameters of the neural network.

For the gyroscope calibration, the error initially drops more rapidly. This could be derived to the fact that, for each generated sequence of measurements from a motion, there are 101 different gyroscope measurements available while the accelerometer calibration only utilizes two measurements from each motion. For the accelerometer to discover a sufficiently large fraction of the set of possible states, it will hence require more motions than the gyroscope.

4.7.2 Accuracy of Adaptive Algorithm

Although the results show that the calibration error produced by the current sensor model decreases during the training, the calibration does not reach an accuracy that is satisfying. Why the accuracy of the algorithm does not continue to improve but stagnates at a mean euclidean error that is relatively high can be caused by the choice of network and training parameters. It is also possible that the developed loss functions are designed in a way that makes it difficult or impossible for the neural network to produce a model with higher accuracy.

During the offline calibration with neural network sensor models, from chapter 3, the networks are trained on all available data points at the same time. During the adaptive algorithm, the networks are trained on a large amount of data but the network only have access to one sequence at a time. Since one sequence only represents a small subset of all training data, the design of the training process may induce overfitting.

As mentioned in the theoretical framework, overfitting makes the model perform very well on data that is represented in the training set but poor on unseen data. From the histograms in Figure 4.16 and Figure 4.18, it can be seen that the sensor model does not calibrate all measurements successfully. Especially for the gyroscope the spread of the residuals causes the mean error to increase significantly. In Figure 4.17, the disparity between calibrated measurements and true angular rate is noticeable larger between samples $80 - 160$ than for the remaining samples. For the accelerometer it can be seen in Figure 4.15 that most calibrated measurements, but not all, quite accurately fit to the true acceleration. Figure 4.11 and 4.12 show that after the initial training all calibrated measurements are located at a significant distance from ground truth. Comparing these figures with Figure 4.15 and 4.17 indicates that the adaptive calibration adjusts the sensor model such that the improvement of the calibration is not consistent for all measurements. The inconsistent behaviour is probably due to overfitting.

Evaluation of the adaptive calibration algorithm on the test set resulted in an average error of about 9 mg for the accelerometer. For offline calibration of the accelerometer the average error was about 1 mg. This comparison shows that the adaptive calibration algorithm does not perform as well when trained on ideal simulated data as the offline does when trained on real data.

4.7.3 Practical Aspects

The algorithm is designed with the assumption that the center of the sensor frame is also the center of rotation. In most applications where IMUs are used, they are mounted such that rotations that do not cause any linear acceleration, is either difficult or impossible to perform. To be considered for implementation in a real system, the algorithm needs to work without this assumption.

4.8 Further Development

The developed adaptive algorithm can be considered for further development in several aspects. There are changes that can be made in order to increase the practical usefulness of the algorithm and adjustments that could improve its accuracy.

4.8.1 Finer Tuning of Training Parameters

The accuracy of a neural network and the speed of the training process can be improved by accurate tuning of parameters such as learning rate, choice of optimizer and activation functions, and the structure of the network. In this project, parameters have been tuned as far as time has allowed. Further development of the algorithm could though include a more thorough tuning, perhaps extending parameters to be adaptive. For instance the learning rate could be modeled as a function of the current loss such that, as the loss decreases the adjustments of the model becomes finer.

4.8.2 Eliminate Assumptions

When the accelerometer is subjected to linear acceleration, it is not possible, by only looking at the measurements, to distinguish how much of the measured acceleration is due to gravitational force. To avoid this problem the constraint that the IMU only rotates around its own center during calibration was introduced. However, the constraint limits the algorithm's possibility to be useful in practice. A possible improvement is hence to extend the model such that fewer assumptions and simplifications are needed. In dead-reckoning, the gyroscope is used to estimate the orientation of the IMU such that the gravitational acceleration can be cancelled. It could be investigated if it is possible to integrate the calibration algorithm with a filter that performs pose and orientation estimation and hence take advantage of the information obtained from filtering to circumvent the assumptions.

4.8.3 Alternative Sensor Model

Based on the discussion regarding overfitting, the algorithm may benefit from applying another method for sensor model approximation than neural networks. For instance the sensor model could be a predefined polynomial function. The parameters could then be updated using an iterative optimizer, like gradient descent, but keeping the general concept of the algorithm and the calculations of the loss. This could be beneficial since it would be easier to observe, constrain and guide the iterative process of the algorithm. Choosing a sensor model that is known to be good at interpolating between samples would reduce the risk of overfitting.

5 Conclusions

In this thesis two approaches for sensor modeling have been investigated and evaluated. More specifically, it was investigated how useful neural networks are for sensor model approximation. It was found that there is a trade off between the accuracy and the amount of data used by the algorithm. Using 400 data points, the neural network performs a calibration that halves the errors compared to a linear calibration model.

The thesis has also included the development of an adaptive calibration algorithm. For the adaptive online calibration method, the approach has been to iterate between calibrating the accelerometer, using the gyroscope as reference, and calibrating the gyroscope, using the accelerometer as reference. The purpose was to make the IMU self-calibrated and the results indicate that the introduced concept can be used to approximate the sensor model without any external information. The developed algorithm is adaptive in the sense that it continuously take new measurements into account and can update the calibration accordingly. In this thesis, temperature has not been considered as a parameter in the sensor modeling. It can though be argued that, since the adaptive algorithm update the sensor model continuously, it will be able to compensate for changes in the sensor's characteristics due to temperature. This argument also holds for other changes in the sensor's nature or in the local environment.

The results from validation show that the implemented adaptive calibration does not perform as well as the offline calibration. A further reason that complicates implementation in a real system is that the algorithm is constrained to use data from an IMU that only rotates around its own center. There are though some future work that could improve the accuracy of the algorithm and increase its value in a practical point of view.

Analysis of the calibration error, after termination of the online algorithm, shows that the calibration is accurate for some measurements but the spread of the residuals is wide which degrade the algorithm's overall performance. The wide spread is probably produced due to overfitting. Overfitting is hard to prevent since, although a lot of data is used in total, the algorithm only processes one measurement at a time. The developed algorithm could hence benefit from a less complex sensor model which interpolates between samples in the data.

The final conclusion of this thesis is that the introduced approach for adaptive calibration can update the sensor model in the right direction. It though needs some further investigations to see whether a higher accuracy can be achieved and if the limitations on the algorithm can be neglected. The mentioned improvements would increase the possibility of practical implementation of the algorithm.

Bibliography

[1] M. Abdel-Hafez, “On the development of an inertial navigation error-budget system”, *Journal of the Franklin Institute*, vol. 348, pp. 24–44, Feb. 2011. DOI: `10.1016/j.jfranklin.2009.02.008`.

[2] M. Kok, J. D. Hol, and T. B. Schön, “Using inertial sensors for position and orientation estimation”, *CoRR*, vol. abs/1704.06053, 2017. arXiv: `1704.06053`. [Online]. Available: `http://arxiv.org/abs/1704.06053`.

[3] X. Niu, Y. Li, H. Zhang, Q. Wang, and Y. Ban, “Fast thermal calibration of low-grade inertial sensors and inertial measurement units”, in *Sensors*, 2013.

[4] K. Draganová, M. Lassak, P. Lipovsky, V. Kan, and T. Kliment, “Gradient methodology for 3-axis accelerometer static calibration”, *International Conference on Military Technologies (ICMT) 2015*, pp. 1–5, 2015.

[5] Y. Zhong, Y. Xu, N. He, and X. Yu, “A new drone accelerometer calibration method”, in *2018 37th Chinese Control Conference (CCC)*, Jul. 2018, pp. 9928–9933. DOI: `10.23919/ChiCC.2018.8482568`.

[6] Z. F. Syed, P. Aggarwal, C. Goodall, X. Niu, and N. El-Sheimy, “A new multi-position calibration method for MEMS inertial navigation systems”, *Measurement Science and Technology*, vol. 18, no. 7, pp. 1897–1907, May 2007. DOI: `10.1088/0957-0233/18/7/016`. [Online]. Available: `https://doi.org/10.1088%5C%2F0957-0233%5C%2F18%5C%2F7%5C%2F016`.

[7] A. Bernardini and S. D. Fina, “A neural network to approximate nonlinear functions”, in *[1991] Proceedings of the 34th Midwest Symposium on Circuits and Systems*, May 1991, 545–548 vol.1. DOI: `10.1109/MWSCAS.1991.252103`.

[8] D. Xu, Z. Yang, H. Zhao, and X. Zhou, “A temperature compensation method for mems accelerometer based on lm_bp neural network”, in *2016 IEEE SENSORS*, Oct. 2016, pp. 1–3. DOI: `10.1109/ICSENS.2016.7808702`.

[9] G. Araghi and R. J. Landry, “Temperature compensation model of mems inertial sensors based on neural network”, in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, Apr. 2018, pp. 301–309. DOI: `10.1109/PLANS.2018.8373395`.

[10] Q. Zhang, Z. Tan, and L. Guo, “Compensation of temperature drift of mems gyroscope using bp neural network”, in *2009 International Conference on Information Engineering and Computer Science*, Dec. 2009, pp. 1–4. DOI: `10.1109/ICIECS.2009.5365140`.

[11] C. Fan, Z. Jin, W. Tian, and F. Qian, “Temperature drift modelling of fibre optic gyroscopes based on a grey radial basis function neural network”, *Measurement Science and Technology*, vol. 15, no. 1, pp. 119–126, Nov. 2003. DOI:

10.1088/0957-0233/15/1/016. [Online]. Available: https://doi.org/10.1088%2F0957-0233%2F15%2F1%2F016.

[12] X. Li and Z. Li, "Vector-aided in-field calibration method for low-end mems gyros in attitude and heading reference systems", *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 11, pp. 2675–2681, Nov. 2014, ISSN: 0018-9456. DOI: 10.1109/TIM.2014.2313434.

[13] F. Bin, C. Wusheng, and D. Li, "An optimal calibration method for a mems inertial measurement unit", *International Journal of Advanced Robotic Systems*, vol. 11, p. 1, Feb. 2014. DOI: 10.5772/57516.

[14] L. Wang and F. Wang, "Intelligent calibration method of low cost mems inertial measurement unit for an fpga-based navigation system", *International Journal of Intelligent Engineering and Systems*, vol. 4, pp. 32–41, Jun. 2011. DOI: 10.22266/ijies2011.0630.04.

[15] J.-K. Shiau, C. L. Huang, and M.-Y. Chang, "Noise characteristics of mems gyro ' s null drift and temperature compensation", in *Journal of Applied Science and Engineering*, vol. 15, 2012, pp. 239–246.

[16] M. Wahde, *Biologically inspired optimization methods: An introduction.* WIT Press, Aug. 2008, ch. A.3, ISBN: 9781845643447.

[17] A. Tch, *The mostly complete chart of neural networks, explained*, Accessed: 2019-04-09, Towards Data Science, Aug. 2017. [Online]. Available: https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464.

[18] D. V. H Abdi and B. Edelman, *Neural networks, quantitative applications in the social sciences.* Thousand Oaks, California, USA: SAGE Publications, Inc, 1999, ch. 1-2, ISBN: 9781412985277.

[19] J. Brownlee, *A gentle introduction to the rectified linear unit (relu) for deep learning neural networks*, 2019. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[20] A. Sharma. (Mar. 2017). Understanding activation functions in neural networks. Accessed: 2019-02-05, [Online]. Available: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

[21] A. Kattan, R. Abdullah, and Z. W. Geem, *Artificial neural network training and software implementation techniques. [electronic resource].* Ser. Computer networks. Nova Science Publishers, 2011, ch. 4, ISBN: 9781611229905. [Online]. Available: http://proxy.lib.chalmers.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat06296a&AN=clc.b2050342&site=eds-live&scope=site.

[22] C. C. Aggarwal, *Neural networks and deep learning. [electronic resource] : A textbook.* Springer International Publishing, 2018, ch. 1.3, ISBN: 9783319944630. [Online]. Available: http://proxy.lib.chalmers.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat06296a&AN=clc.b2548388&site=eds-live&scope=site.

[23] P. J. Werbos, *The roots of backpropagation: From ordered derivatives to neural networks and political forecasting.* New York, NY, USA: Wiley-Interscience, 1994, ISBN: 0-471-59897-6.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT Press, 2016, `http://www.deeplearningbook.org`.

[25] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015, 2014. [Online]. Available: `http://arxiv.org/abs/1412.6980`.

[26] A. Bhande. (Mar. 2018). What is underfitting and overfitting in machine learning and how to deal with it. Accessed: 2019-05-17, [Online]. Available: `https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76`.

[27] D. Lay, *Linear algebra and its applications*, New International Edition. London, England, United Kingdom: Pearson Education M.U.A., Aug. 2013, ch. 6, pp. 388–394, ISBN: 9781292028316.

[28] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Data preprocessing for supervised learning", *International Journal of Computer Science*, vol. 1, pp. 111–117, Jan. 2006.

[29] S. Stančin and S. Tomažič, "Angle estimation of simultaneous orthogonal rotations from 3d gyroscope measurements", *Sensors*, vol. 11, no. 9, pp. 8536–8549, 2011, ISSN: 1424-8220. DOI: `10.3390/s110908536`. [Online]. Available: `http://www.mdpi.com/1424-8220/11/9/8536`.

[30] S. Stancin and S. Tomazic, "On the interpretation of 3d gyroscope measurements", *Journal of Sensors*, vol. 2018, pp. 1–8, Mar. 2018. DOI: `10.1155/2018/9684326`.