CHALMERS

Automatic calibration and virtual alignment of MEMS-sensor placed in vehicle for use in road condition determination system

Master's Thesis in Systems, Control, and Mechatronics

REBECKA LARSDOTTER, DAVID JALLER

Department of Signals & Systems
Chalmers University of Technology
Gothenburg, Sweden 2014

Master's Thesis 2013:1
EX002/2014

Abstract

This master's thesis is based on a wish for an addition to an existing software/hardware solution used to detect softness of gravel roads. The goal of the thesis has been to create an algorithm that, using real time sensor inputs, automatically detects a misalignment between a hardware containing MEMS-sensors and a vehicle in which the hardware is positioned. This algorithm would consequently enable unspecified mounting of such a hardware in a vehicle while keeping the input signals to other software as if the hardware where aligned with the vehicle.

To reach the goal of the project different methods on orientation estimation using sensor information were studied. The methods considered most suitable for the application were assembled into a complete model. Simulations were performed with data collected from tests with a log device mounted with a known misalignment and the output, being three angles estimates of this misalignment, were compared to the true angles of misalignment. The algorithm was tuned until the desired behavior was reached.

The result of the project is a model of the algorithm that in a relatively short time detects a three dimensional misalignment and rotates the input data from the sensors according to the three calculated angles. The main deficiencies of the algorithm is that it does not distinguish between the vehicle frame and the world frame and that it is not tuned to be independent of the driving environment in which it is used. The resulted algorithm is not specific to the intended area of usage but might instead be used for several applications with some adjustments.

Acknowledgements

We would like to thank our supervisors Jonas Fredriksson at Chalmers University of Technology and Magnus Andersson at Semcon for support, ideas and guidance throughout the work. We would also like to thank Semcon for providing the opportunity to participate in the BiFi project and our colleges at Semcon for help during the project.

The Authors, Gothenburg 20/1/2014

Contents

1

Introduction

The introduction aims to give the reader an understanding of the purpose of the project and how it is related to the already existing BiFi project. It also includes a detailed description of the problem and information about the conditions for the work.

1.1 Spring thaw and it's effect on gravel roads

According to [2], the phenomena that the water in the ground freezes during a limited period due to temperature drop is called seasonal ground frost. In Sweden and other areas, where the temperature difference during the year is big enough to cause a freeze and a melt period, this causes problems. Since the density of ice is lower than of water the change between these states can lead to deformations as well as changed properties of the ground which causes problems for roads located in these areas.

The most critical period of the year is the period when the frozen water in the ground melts. When the temperature increases water that was bounded in ice will be released. The increased amount of water might lead to saturation of the construction which will result in decreased shear solidity. The melting starts at the top layer and in the first stage of the melting process the lower parts remain frozen. During this stage the gravel roads become wet and can easily be deformed if heavy traffic is driven on these roads (see figure 1.1). It is at this stage important to prevent heavy vehicles to run here. To close a road from traffic can be expensive for the forest industry due to need of detours, restructuring of loads or pauses in logging. On the other hand, if the roads are open when the carrying capacity is too low they might suffer from major damage which can be very costly due to need of reconstruction. Hence, it is important to have the road closed when and only when it is needed. When the melting period occurs and for how long it is ongoing is dependent on several parameters like soil type, temperature changes, traffic, water availability and drainage. Up to this day predictions on this period are

Figure 1.1: Gravel road softened due to melting of spring thaw.[17]

made manually.

To make a accurate prediction based on all parameters is a complex problem that raised a wish for a method to perform real time detection of carrying capacity that together with a prediction will generate a more reliable result.

1.2 BiFi

The Gothenburg-based technology company Semcon has since 2010 together with Klimator and other interessants been involved in a project named BiFi, see [13], which stands for load bearing information through vehicle intelligence (Bärighetsinformation genom fordonsintelligens) aimed at automatically determine the current carrying capacity of gravel roads.

An algorithm has been developed that is based on the fact that the lateral vibrations of a vehicle will be dampened if the road that the vehicle is driving on is soft and that these vibrations can be properly measured using accelerometers. This damping is, due to centrifugal acceleration, particularly noticeable in curves and within a certain frequency band damping are expected to differ depending on the road condition.

For the purpose of measurements and calculations a specific hardware has been developed on which the algorithm run. The hardware is meant to be mounted in postal delivery vehicles which operates in areas where spring thaw is a concern. These vehicles will cover mostly the same routs every day, allowing for a stable source of road condition information.

In addition to accelerometers, the hardware contains equipment for processing data and transfer of the processed data and the condition of the road surface (softened or hard) is estimated and, together with the GPS coordinates, uploaded to a database. The uploaded data are then validated and interpreted against a road weather model based on, amongst other, weather forecasts, to provide information about which roads

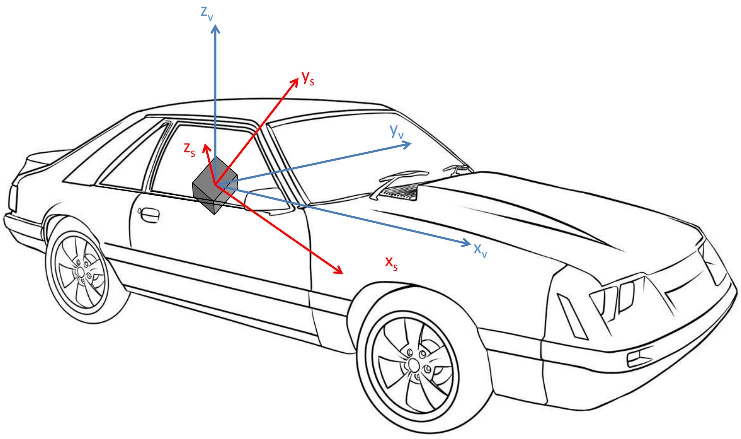


Figure 1.2: Misalignment between the coordinate system of the vehicle and the coordinate system of the sensors.

that tolerate being trafficked by heavy vehicles and which roads that should be closed for such traffic until conditions are improved.

1.3 Problem motivation

As the number of vehicles taking part in the BiFi project increases, the probability for one or more of the devices being mounted incorrectly does as well. Also, the mounting of the hardware will have to be carried out by people outside of the project, postal service personel, mechanics etc, making it harder to guarantee that every device is mounted correctly. An incorrect mount of the device results in that the coordinate system of the device will not concide with the coordinate system of the vehicle, see figure 1.2, which affects the reliability of the soft curve detection results.

As mentioned above, the soft curve detection algorithm uses measurements of acceleration experienced in the lateral direction, i.e the y-direction, of the vehicle (y^v). Here and in the rest of this paper the notation s indicates the sensor frame and v the vehicle frame. It has up to now been assumed that the accelerometer axis y^s is perfectly aligned with y^v. If that is not the case he result of the algorithm will be effected. If for example the device were rotated 90° around its z^s-axis so that $y^s = x^v$, then the input to the algorithm would instead be the longitudinal acceleration, which has different characteristics and which will not be affected by the centrifugal force during a curve. If the device were instead rotated 90° around its x^s-axis so that $y^s = z^v$, the input would be almost only the static and constant gravitational acceleration.

An angle between the sensor and vehicle frame, $0 < \alpha^{sv} < 90°$, around one of the axes x^s or y^s would result in a projection of a_x^v or a_z^v on the measured acceleration a_y^s, and if the angle is unknown the result of the algorithm will be unpredictable.

If the device were rotated around the x^s or z^s axes with an angel of exactly 180° it would not affect the BiFi algorithm since it is using variance to compare the damping, making it independent of signs on the axes. The same is true for any angle round the y^s axis due to orthogonality. Although, in order to make the algorithm general and possible to use in other applications all three angles are considered.

An algorithm that could find and compensate for the misalignment would effectively eliminate this cause of unreliability. Additionally, for the sensors to measure the same vibration as the vehicle is experiencing, the device has to be mounted firmly and there might be vehicles where the preferred mounting position gives the device a misalignment. With automatic virtual alignment this would not be a problem.

1.4 Detailed problem description

A misalignment by angles α_x^{sv}, α_y^{sv}, α_z^{sv} could be described with the rotation $R = R_x R_y R_z$. The rotation can be reverted by applying the inverse rotation $R^{-1} = R^T$ so that if the measured accelerations a_x^s, a_y^s, a_z^s sampled in the sensor frame are multiplied with R^{-1}, the resulting vector describes the accelerations a_x^v, a_y^v, a_z^v experienced in the vehicle frame (more on rotation matrices in section 3.1).

The existing BiFi software is thus to be updated so that before the soft curve detection algorithm can start running an algorithm is executed that finds the rotation matrix R. Then during normal operation the matrix R^{-1} will be applied to the raw data after every sample before it can be used by the curve detection algorithm, or by any other algorithm used. The challenge of this project is to find the algorithm for finding the rotation matrix R that describes the misalignment.

The soft curve detection algorithm is developed in such a way that the driver of the car does not have to do any adjustments in his/her driving pattern or even be aware that the system is mounted in the car. This should be true also for the updated software, so few assumptions can be made on how the vehicles is moving during the calibration.

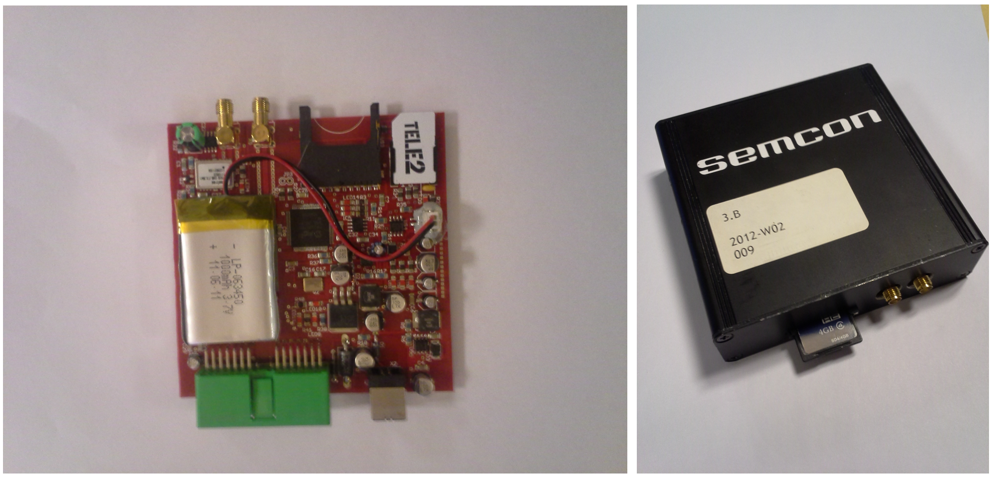
The calibration will be a so called self calibration, since the same sensors that are to be calibrated are used in the calibration process.

1.4.1 Description of the hardware

The project is limited to the existing BiFi hardware shown in figure 1.3. All components and connectors are fitted on a single card which is then protected by a aluminum chassis. Below are listed descriptions of the components included that are of importance to this project.

Digital Accelerometer ADXL345

The ADXL345 is a three axis accelerometer with embedded circuitry for sampling and AD conversion. It has a maximum measuring range of $\pm 16g$ and a resolution of 13 bits, yielding a maximum sensitivity of 3.9 mg per bit. It measures both static and dynamic



(a) (b)

Figure 1.3: The hardware used in the project , shown in a) whiteout its chassis and in b) with its chassis.

acceleration. In free fall the accelerometer measures zero and when being still on the ground it measures the normal acceleration g from the ground, i.e. $+g$ on the z axis.

Fastrax IT520 GPS Module

The IT520 contains receiver for GPS satellite signals and circuitry for calculating position, velocity and heading. It can output horizontal position with an accuracy of 3 meters and velocity with 0.1 m/s accuracy at a rate of 1 Hz.

L3G4200D Gyroscope

The gyroscope measures angular velocities around three axes. It has configurable resolution of down to 8.75 millidegrees per second, and a maximum sampling rate of 800 Hz. It also contains integrated configurable digital filters, and a temperature sensor.

Microchip PIC32MX795F512L Microprocessor

The controller features a 80 MHz 32 bit processor with Microchip's MIPS32 instruction set. The processor allows for two-cycle multiplication of 32 bit numbers. The controller contains 512 KB of program memory and 64 KB internal memory. Number of analog inputs are 16 with 10-bit AD conversion. There is also interface trough USB, CAN and JTAG.

Other equipment

Additionally the hardware is equipped with components for transmitting data via GSM network such as SIM-card module and GSM transceiver, as well as SD-memory slot for nonvolatile storage, much useful especially during software development. The card is powered from the cigarette lighter outlet in the vehicle, but the card also holds a battery allowing for shutdown processes to complete after the driver has turned off the ignition.

2

Related work

Several projects concerning the problem with calibration of a three dimensional set of accelerometers have been done. A frequently used technique to determine the orientation of the sensors is to use the gravitational acceleration in static state (i.e when the vehicle is not moving) together with a significant type of movement easy to classify (e.g forward acceleration or braking). In papers that cover these projects methods are presented where assumptions that simplifies the actual problem are common. For example it is common to assume that there are no slope of the ground or that the average of left turns and right turns will be even. In the projects that do not make this kind of assumptions the methods are often based on specific information that requires the driver to perform a set of actions.

In the paper "Research on Calibration method for the installation error of three-axis acceleration sensor" [19] Z.Hui et al. present a mathematical method to compensate for the installation error of a set of accelerometers. This mathematical method includes a solution to the problem of separation of the rotation of sensors and the rotation of the vehicle. In their paper they introduce, beyond the ordinary pitch (ϕ), roll (θ) and yaw (γ) that represent the orientation of the sensors with respect to the world, two additional angles that represent the orientation of the vehicle with respect to the world. One of the new angles is the angle between the slope of the ground that the vehicle is placed on (α) and the other is the difference between the direction of the vehicle and the direction in which the slope of the ground is maximal (β). As many before them they use the fact that only the gravitational acceleration is present in stationary state and as a result they get an undetermined set of equations. By describing the relation between the sensors and the vehicle as a plane equation and applying a least-squares fitting method they obtain an estimation of the plane parameters. Several measurements and corresponding estimates are then used as basis to find the plane that best describes this relation and it is possible to calculate an estimation of the angles. The method explained in their paper requires the driver to park the vehicle on the same plane several times which also means that the method requires involvement of the driver.

Javier Almazan et al. [7] addresses a problem similar to the one presented in this report. Their project is based on a smartphone inside a vehicle where the problem is to find the orientation of the phone relative to the vehicle using the phone's accelerometers and gyroscopes. They use the gyroscope readings filtered with a noise reducing Kalman filter as the pitch and roll. They then decouple the contribution of gravity on the lateral and longitudinal accelerometer readings and assumes that when the vehicle is accelerating forward on a straight line these readings should after decoupling show only contribution from the forward acceleration, thus making it possible to determine also the yaw angle. The resulting estimation of the yaw angle is highly fluctuating but after applying an Expectation-Maximization (EM) algorithm rather accurate estimates are obtained. Several simplifications are made throughout the project; Almazan et al. assumes that the road has no slope in either direction and that it is known that the vehicle is accelerating forwards without turning. Also the fact that gyroscopes measures angular velocity which needs to be integrated in order to obtain the current angle is not addressed, possibly since firmware in the smartphone performs the integration returning absolute angles.

V.N.Padmanabhan et al. has a similar project setup with a smartphone inside a car. The purpose of their project is to detect traffic conditions with sensors in a smartphone and in their paper "Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones" [10] they present a calibration algorithm based only on information from the accelerometers and GPS. Instead of using the fact that the only acceleration present when the car is stationary is the gravitational acceleration they use the median of the accelerations over a 10 second time window. They claim that the median of the measured accelerations will be stable since bumps and other spikes in acceleration will be followed by a corresponding spike in the opposite direction. Hence, the resultant of the median values will correspond to the direction of gravitation and they align the z axis of the coordinate system of the accelerometers with the direction of the gravitational acceleration. After alignment the yaw angle will not be affected by the gravitation and to detect the yaw angle a distinct acceleration with known orientation orthogonal to the z-axis is needed. Instead of using forward acceleration they use deceleration with the motivation that braking tends to be harder and hence easier to detect. The restrictions for using a braking sequence for alignment is that the deceleration is strong enough and that there are no sharp changes in the path. This information is provided by the GPS.

In "Correcting Smartphone Orientation for Accelerometer-Based Analysis" [8] Tundo et al. calibrates a three axis accelerometer with unknown orientation, though without concerning the yaw angle. In their paper they present a more sophisticated way of transforming between two coordinate systems; instead of representing rotation with three Euler angles they use a representation of one rotation around one axis, namely the axis orthogonal to the plane spanned by two vectors. One of these vectors is the resultant of the measurements and the other is an axis in the original coordinate system with known magnitude. The angle of rotation is the angle between these two vectors and rotating the coordinate system around the calculated axis with this angle will represent the transformation from the coordinate system of the measurements to the coordinate system with known magnitudes.

J.D.Begin et al. [1] and Intelligent Mechatronic Systems Inc. [9] has both created and patented algorithms that automatically calibrates three-dimensional acceleration data. The structure of their solutions to the problem is similar in the way that they both use the effect of gravity when the vehicle is still and the effect of forward acceleration when the vehicle is moving for calibration. In both patents the vehicle is assumed to be on a flat surface. In [1] data is sampled and stored when the vehicle is determined to be in zero movement state and in forward acceleration state. Both sets of data are then used to solve the systems of equations consisting of the complete rotation matrix, the accelerometer readings for the different states and the expected readings if the accelerometers were aligned. The developed method for solving the equations, named "Recursive Orientation Update", is not fully explained in the patent registration, but the concept is to recursively calculate the error between measured and transformed accelerometer values, expected values when in the zero movement state or the forward acceleration state, and then update the orientation matrix. It is stated in the report that if the velocity of the vehicle is low enough there will be no lateral acceleration even if the vehicle is turning. In their algorithm there is therefore no need for the vehicle to be moving in a straight line in the forward acceleration state but instead the restrictions will be on the velocity, determined from GPS data, to be low enough, and for the sum of acceleration in all three directions to exceed a threshold.

In [9] the calibration is done in two steps; vertical- and horizontal alignment. The first step is to find the Euler angles roll and pitch and to generate the transformation matrices for vertical alignment. This is done by comparing accelerometer data from when the vehicle is in zero motion state with the expected gravity acceleration vector. In the second step, when the vehicle is determined to be accelerating forward in a straight line, acceleration data is sampled and multiplied with the transformation matrices previously found. They use the fact that the effect of gravity should be seen on neither of the horisontal axes x and y after transformation since flat surface is assumed, so all contributions for the accelerations seen are due to movement of the vehicle. To assure that the vehicle is in fact driving in a straight line in the forward acceleration state a restriction is stated saying that the angle between the resultant from the x- and y-axes, and the y-axis must remain constant within a threshold. The data will still be varying within the interval of the threshold why a rather large set of data is collected and a statistical method called Principal Component Analysis is applied to find the direction of heading of the vehicle relative to the measurement device, thus finding also the third transformation matrix for horisontal alignment.

Sara Stansin [15] et al. presents in their report "Angle Estimation of Simultaneous Orthogonal Rotations from 3D Gyroscope Measurements" a technique for deciding the tilt angle of a rotating object. They show how three simultaneous rotations around orthogonal axes, as measured by a gyroscope, instead can be represented by a rotation around one axis which components are the normalization of the magnitudes of the three simultaneous rotations. An experiment is performed using a known axis of rotation and a 3D gyroscope tilted in respect to that axis. The gyroscope was placed on the rotating disk of a gramophone and the gyro measurements of the three axes while the disk was spinning with constant speed was sampled and averaged. The average was used to calculate the axis of rotation in the gyroscope frame, which could then be used to determine how much the gyro was tilted.

3

Theory

THIS CHAPTER is meant for the reader who needs some background knowledge in order to fully understand every concept used in the algorithm. A suggestion is to first skip this chapter and then return to specific sections when needed to understand the following chapters.

3.1 Rotations

A rotation is a change of orientation of a coordinate system where the position of origo remains unchanged. The same rotation can be described in multiple ways with different conventions on angles and axes.

The two main representations for rotations are combinations of rotation matrices and the axis-angle representation. The axis-angle representation is more concise with only one angle needed to represent any rotation while rotation matrix representation has the advantage that rotations can be easily separated into sub rotations.

3.1.1 Rotation matrices

According to [18], a rotation with an angle α around one of the three axes (x,y,z) in a Cartesian coordinate system is described by

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \tag{3.1}$$

$$R_y(\alpha) = \begin{bmatrix} cos(\alpha) & 0 & -sin(\alpha) \\ 0 & 1 & 0 \\ sin(\alpha) & 0 & cos(\alpha) \end{bmatrix} \tag{3.2}$$

$$R_z(\alpha) = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Any rotation can be described by a sequence of one or more of these rotations, using one of several possible conventions. Combinations of rotations are not commutative, the resulting orientation after rotating first around the x-axis and then around the y-axis will not be the same as if rotating with the same angles but in the opposite order.

$$R_x(\alpha)R_y(\alpha) \neq R_y(\alpha)R_x(\alpha) \tag{3.4}$$

Rotation matrices have other specific properties that are important to consider; a negative rotation is obtained by multiplication with the inverse of the rotation matrix and the inverse of a rotation matrix is the same as the transpose of the same matrix.

$$R(-\alpha) = R(\alpha)^{-1} = R(\alpha)^T \tag{3.5}$$

The inverse of a combination of rotations can therefore be represented with the inverse of each rotation but in the opposite order.

$$R_{xy}(\alpha_x,\alpha_y)^{-1} = \left(R_x(\alpha_x)R_y(\alpha_y)\right)^{-1} = R_y(\alpha_y)^{-1}R_x(\alpha_x)^{-1} \tag{3.6}$$

A vector in a coordinate system can be described either as a row vector or a column vector and hence be multiplied with the rotation matrix with pre- or post multiplication. To apply a rotation that corresponds to a rotation around the x-axis followed by a rotation around the y axis the order of rotations are switched according to

$$R_x(\alpha)R_y(\alpha)\begin{bmatrix} x \\ y \end{bmatrix} = [x\ y]R_y(\alpha)R_x(\alpha) \tag{3.7}$$

Conventions

All changes in orientation can be achieved by different combinations of rotations around three axes and there are several conventions on how to describe the change. Depending on the application, choosing the most appropriate convention can help facilitate calculations. In general though, it is of greater importance to be consistent with the convention rather than which one to use.

According to [3], any orientation can be reached either by a sequence of rotations around all three axes, or by rotation around one of the axes, then around one of the remaining two, followed by rotation around the first axis again. In the former case the three angles to rotate are called proper Euler angles or just Euler angles, and in the later case the angles are referred to as Tait-Brian angles or Nautical angles. Since rotations are not commutative the order of the rotation sequence has to be decided. There are six possible sequences of Euler angles; z-x-z, x-y-x, y-z-y, z-y-z, x-z-x and y-x-y, and as many for Tait-Bryan angles; x-y-z, x-z-y, y-x-z, y-z-x, z-x-y and z-y-x.

According to [16], the combinations of rotations, independent on choice of combination convention, can describe rotations either around the axes of the original frame that remains fixed trough the rotation - Extrinsic rotation, or around the axes of the rotating frame - Intrinsic rotation. Extrinsic rotation is rather easy to grasp while intrinsic needs a little further explanation. If for example a Tait-Bryan rotation is intrinsic the sequence x-y-z will be written x-y'-z", where y represents the y axis of the rotating frame after rotation around the x-axis and z" is the z axis of the rotating frame after the first two rotations. e.g if a cartesian coordinate system is rotated 90° around the x-axis such that the y-axis is aligned with the original z-axis and a rotation is applied around the x-axis this rotation will be around the new axis (the original z-axis). The difference between an intrinsic and an extrinsic rotation is illustrated in figure 3.1 respectively figure 3.2.

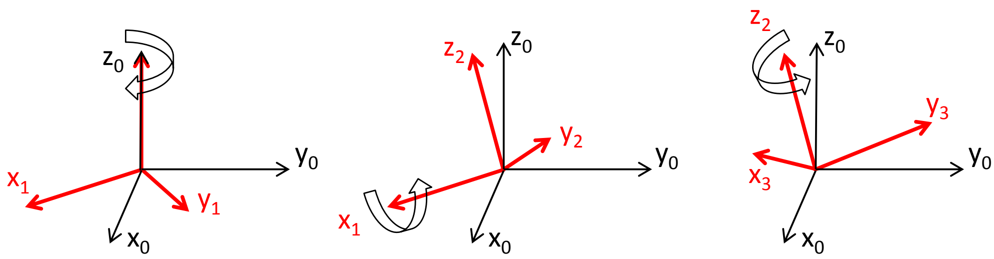


Figure 3.1: Intrinsic rotations (varying rotation axes).

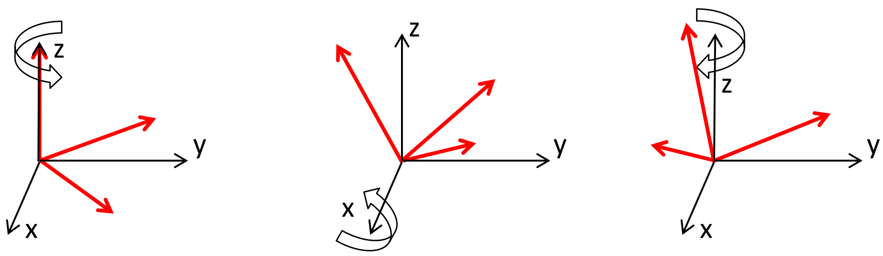


Figure 3.2: Extrinsic rotations (fixed rotation axes).

3.1.2 Axis-angle representation

As mentioned in section 3.1 the most common way of describing rotation is with the rotation matrices. However, in the axis-angle representation a rotation is described by just one angle of rotation, ϕ, instead of three (α_x, α_y and α_z) and the vector around which to rotate, $\hat{\mathbf{e}}$.

$$(axis{,}angle) = \left(\begin{bmatrix} \hat{e}_x \\ \hat{e}_y \\ \hat{e}_z \end{bmatrix}, \phi \right) \tag{3.8}$$

Since $\hat{\mathbf{e}}$ is a unit vector the same rotation can be described more compactly with the rotation vector Φ.

$$\Phi = \hat{\mathbf{e}}\phi \tag{3.9}$$

$\hat{\mathbf{e}}$ is the vector that will not be affected by transformation with rotation matrix R , i.e.

$$\hat{\mathbf{e}} = R\hat{\mathbf{e}} \tag{3.10}$$

It is thus also true that $\hat{\mathbf{e}}$ is an eigenvector to R corresponding to the eigenvalue 1. The angle of rotation ϕ is the angle between any vector $\mathbf{v}$ that is perpendicular to the rotation axis and the same vector after transformation, $R\mathbf{v}$, as can be read in [11].

Simultanious rotations

A rotation matrix $R(\alpha_x,\alpha_y,\alpha_z)$ represents a sequence of rotations around the three axes in some order depending on the convention. The same rotation could also be represented by simultaneous rotations around the three axes which easily can be described according to the axis-angle representation. Important to note is that the magnitudes of rotation when they are simultaneous would the not be the same as if they were in a sequence. The measurements of 3D gyroscope is angular rates of simultaneous rotation around three orthogonal axes.

If three angles $\varphi_x, \varphi_y, \varphi_z$ of simultaneous rotation are known the rotation vector is easy to calculate.

$$\hat{\mathbf{e}} = \frac{\boldsymbol{\varphi}}{||\boldsymbol{\varphi}||} = \frac{1}{\sqrt{\varphi_x^2 + \varphi_y^2 + \varphi_z^2}} \begin{bmatrix} \varphi_x \\ \varphi_y \\ \varphi_z \end{bmatrix} \tag{3.11}$$

and

$$\phi = \sqrt{\varphi_x^2 + \varphi_y^2 + \varphi_z^2} \tag{3.12}$$

The three Euler angular velocities $\omega_x = \frac{d\varphi_x}{dt}$, $\omega_y = \frac{d\varphi_y}{dt}$,$\omega_z = \frac{d\varphi_z}{dt}$ can be represented by a rotation velocity vector which is orthogonal to the rotation plane and which magnitude is equal to the angular velocity in that plane

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3.13}$$

It can, according to [15], be shown that as long the angular velocity around all three axes remain constant or as long as their proportions does, the axis of rotation will remain the same.

$$\Phi = \frac{\boldsymbol{\omega}}{T} \tag{3.14}$$

Thus

$$\hat{\mathbf{e}} = \frac{\boldsymbol{\omega}/T}{||\boldsymbol{\omega}/T||} = \frac{\boldsymbol{\omega}}{||\boldsymbol{\omega}||} \tag{3.15}$$

3.2 Four quadrant inverse tangent

Four quadrant inverse tangent or more commonly *atan*2 is a variation of the trigonometric function *arctangent* used in many computer languages. As explained in [6], in contrast to the normal arctangent atan2 takes two arguments; the x and the y component of the vector to which the angle is to be determined. This allows for the signs of the components to affect the resulting angle, which is represented in all four quadrants with the range $[-\pi \ \pi]$. The precise definition is the following:

$$atan2(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{x} > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{y} \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{y} < 0, x < 0 \\ +\frac{\pi}{2} & \text{y} > 0, x = 0 \\ -\frac{\pi}{2} & \text{y} < 0, x = 0 \\ \text{undefined} & \text{y} = 0, x = 0 \end{cases} \tag{3.16}$$

3.3 Principal Component Analysis

Principal Component Analysis (PCA) is a method for detecting patterns in data that can be used for dimensionality reduction. The method is based on the fact that the eigensystems (eigenvectors with corresponding eigenvalues) of the covariance matrix describes the variance of the projection of data in different directions and that the direction with maximal variance best describes the pattern of the data. In other words the first principal component is the line that minimizes the mean square error of the data points while passing through the multidimensional mean, as described in [4].

3.3.1 Covariance

Covariance between two variables is a measure on how the change of the variables are related and is calculated according to

$$cov(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{\mathbf{x}})(y_i - \bar{\mathbf{y}})}{(n-1)} = E[\mathbf{xy}] - E[\mathbf{x}]E[\mathbf{y}] \tag{3.17}$$

In the two dimensional case a high positive covariance means that the change of the two variables are highly related. On the contrary, a high negative covariance means that the change are highly related but inversed, e.g if one of of the variable is strictly increasing the other is strictly decreasing.

$$var(\mathbf{x}) = \frac{\sum_{i=1}^{n}(x_i - \bar{\mathbf{x}})^2}{n-1} \tag{3.18}$$

The covariance matrix is a description of variance in multiple dimensions, formulated as

$$C = \begin{bmatrix} cov(\mathbf{x},\mathbf{x}) & cov(\mathbf{x},\mathbf{y}) \\ cov(\mathbf{y},\mathbf{x}) & cov(\mathbf{y},\mathbf{y}) \end{bmatrix} \tag{3.19}$$

where the diagonal consist of the variances of the different variables and the other elements is the covariance between the variables. The covariance matrix is a symmetric matrix since $cov(\mathbf{x},\mathbf{y}) = cov(\mathbf{y},\mathbf{x})$ and is also positive-semidefinite.

3.3.2 Eigensystems

An eigensystem of a matrix A is a system composed of an eigenvector v and a corresponding eigenvalue λ that fulfills

$$A\mathbf{v} = \lambda\mathbf{v} \tag{3.20}$$

The equation above describes that an eigenvector $\mathbf{v}$ to a matrix A is a vector that can be multiplied with the matrix yielding a scaled version of the vector itself as result. Every eigenvector has a corresponding eigenvalue λ which describes this scaling; the size describes the change of magnitude and the sign describes whether the direction is the same as the eigenvector (positive) or the opposite (negative). As explained in [5], for a 2×2 matrix the eigenvalues of a matrix is calculated as the roots to the equation

$$det(A - \lambda I) = 0 \tag{3.21}$$

and with the eigenvalues known the corresponding eigenvectors can be calculated with the following system of equations:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \lambda v_1 \\ \lambda v_2 \end{bmatrix} \tag{3.22}$$

Only a square matrix ($n \times n$) can have eigensystems and if it does, the amount of systems is the same as the number of rows (n) of the matrix. Other properties of importance is that eigenvectors are always perpendicular and that the eigenvalues of a positive-semidefinite matrix is strictly positive. Consequently, the covariance matrix describing the variance of two parameters have two eigensystems with perpendicular eigenvectors.

3.3.3 Variance of projections

If the data is projected on a unit vector according to

$$\mathbf{p} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}^T \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{3.23}$$

the variance of the projection is given by

$$var(\mathbf{p}) = \frac{\sum_{i=1}^{n}(x_i - \bar{\mathbf{x}})u_x(y_i - \bar{\mathbf{y}})u_y}{n-1} = \mathbf{u}^T C \mathbf{u} \tag{3.24}$$

where C is the covariance matrix of the original data. To maintain the information stored in the data after projection the unit vector to project the data on should be the vector that the describes the direction in which the variance of the projections is greatest. This can be seen in figure 3.3 where it is possible to see that the variance along x and y is significantly smaller than along $\mathbf{u}$. Since the covariance matrix is symmetric the maximal value of $var(\mathbf{p})$ is obtained when $\mathbf{u}$ is the eigenvector of C with the highest corresponding eigenvalue, as explained in [12].

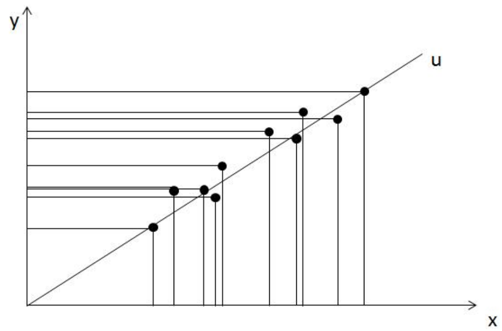


Figure 3.3: Two dimensional data estimated as the vector with maximal variance.

According to [14], the eigenvector with the highest corresponding eigenvalue is called the first principal component. Since the two eigenvectors are perpendicular and the eigenvector with the largest corresponding eigenvalue describes the direction in which the variance of the variables is largest consequently the other eigenvector describes the direction in which the variance of the variables is smallest.

4

The algorithm

In this chapter the ideas adapted and developed for the alignment algorithm are explained. The algorithm contains several steps and the chapter is divided accordingly. The first two sections describes the avalable input signals and the rotation conventions used.

4.1 Analysis of available signals

The signals available for the calibration are accelerations in three dimensions (a_x^s, a_y^s and a_z^s) from the accelerometer, angular rates (ω_x^s, ω_y^s and ω_z^s) from the gyroscope and velocity (v), heading and longitudinal and latitudinal coordinates from the GPS module. The heading and coordinates are not very useful since they are represented in the world frame rather than in the vehicle frame. Figure 4.1 shows accelerations and velocity as they are provided from the sensors during a test drive in urban environment with the hardware aligned with the vehicle. Since the two coordinate systems $[x^v\ y^v\ z^v]$ and $[x^s\ y^s\ z^s]$ coincide, the signal a_x^s corresponds to acceleration and braking. Its magnitude rarely exceeded 2 m/s^2. Large excitation on a_y^s corresponds to when the vehicle is taking a curve and its magnitude depend on the velocity and angular velocity according to

$$a^c = v\omega_z \tag{4.1}$$

The magnitude of this signal is larger than the one from acceleration and breaking. It can be noticed that the velocity has a lower update rate than the accelerometers, and that the acceleration signals are rather noisy. The noise on a_x^s and a_y^s arrives mostly from vibrations from the motor (the noise has same characteristics when the vehicle is stopped with the engine on as when driving) while the noise on a_z^s comes also from the roughness of the road.

Figure 4.2 shows the same situation as in figure 4.1, but after applying a FIR low pass filter on the acceleration signals. The filter used has a cutoff frequency of 0.2 ω^{ny}

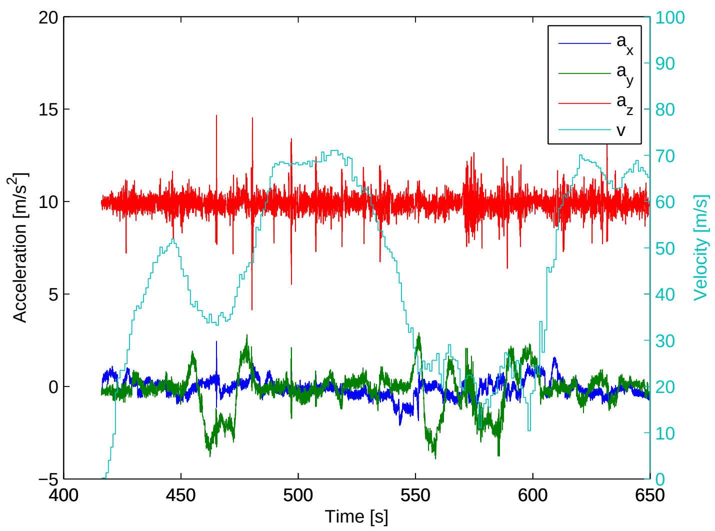


Figure 4.1: Velocity and unfiltered accelerations during three minutes of unspecified driving with the hardware aligned with the vehicle.

and the number of coefficients used for the filter was 20. In order to eliminate even more of the undesired fluctuations a filter with lower cutoff frequency is required. However, to design such a filter, the number of filter coefficients has to be increased. The number of filter coefficients used effectively increases the computational complexity and already at 20 coefficients the filtering uses a large part of the time available between sampling instances. By lowering the cutoff frequency the risk to filter out also useful information from the signals increases.

Figure 4.3 shows angular rates measured during the same test drive as in figure 4.1 and 4.2. The high peaks on ω_z corresponds to when the vehicle is turning.

In the example presented in figures 4.1 - 4.3 the hardware was mounted so that the sensor frame was aligned with the vehicle; the measured signals in the sensor frame equals the signals in the vehicle frame ($\mathbf{a}^s = \mathbf{a}^v$ and $\boldsymbol{\omega}^s = \boldsymbol{\omega}^v$). When instead the hardware is mounted with a misalignment the characteristics in the different directions will be present on all three axes. Figure 4.4 shows a test drive with the hardware mounted with a misalignment.

4.2 Choice of conventions

As discussed in the theory chapter it is of great importance to be consistent with which convention to use when rotating data. Through the project the rotations are described and calculated with the Tait-Bryan angles z-y-x. Using Tait-Bryan angles makes it possible to divide the rotation into rotation around the three axes. All rotations are also described as intrinsic rotations, thus every rotation of a frame is defined around the axes

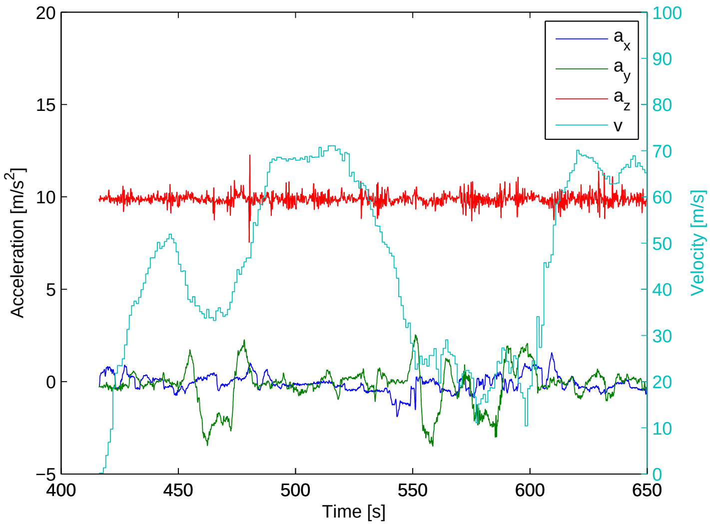


Figure 4.2: Velocity and filtered accelerations during three minutes of unspecified driving with the hardware aligned with the vehicle.

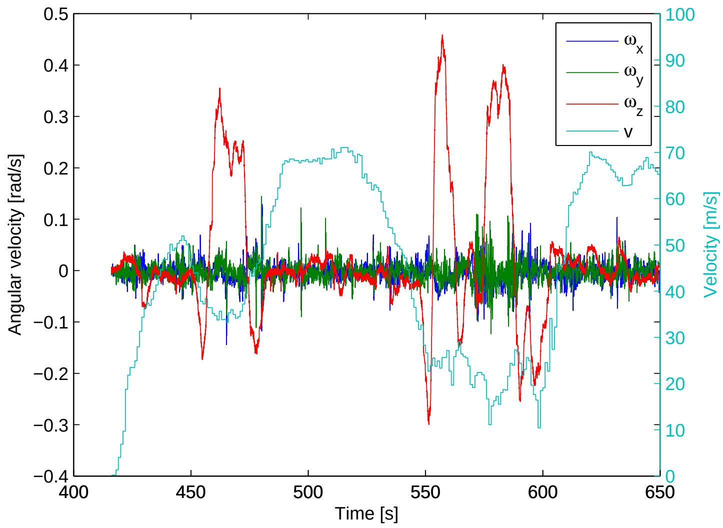


Figure 4.3: Angular velocities during three minutes of unspecified driving with the hardware aligned with the vehicle.

of the rotating frame it self (the sensor frame) rather than around the axes of a fixed reference frame (the vehicle frame or the world frame).

The misalignment of the sensor is considered as an undesired rotation of the coordinate system of the sensor described by

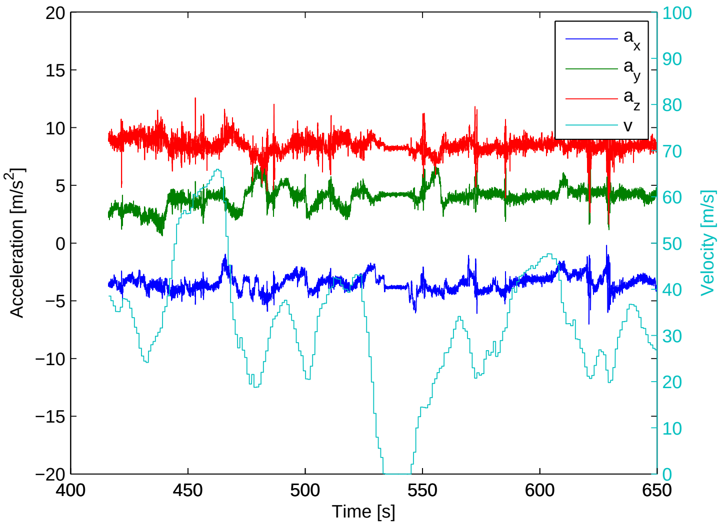


Figure 4.4: Velocity and accelerations during three minutes of unspecified driving with the hardware mounted with misalignment around all three axes.

$$\begin{bmatrix} x^s \\ y^s \\ z^s \end{bmatrix} = R_x(\alpha_z)R_y(\alpha_y)R_z(\alpha_z) \begin{bmatrix} x^v \\ y^v \\ z^v \end{bmatrix} \tag{4.2}$$

Since the matrices are multiplied using left multiplication, this means that the device, and therefore also the sensor frame, has been rotated first around its x-axis then its y-axis and lastly its z-axis. This convention is of great advantage if one wants to be able to first compensate for the rotation around the x- and y-axes (see section 4.3) . By looking at the equations described in section 3.1.1, one can see that the misalignment described by the equation above can be reverted with

$$\begin{bmatrix} x^v \\ y^v \\ z^v \end{bmatrix} = \left(R_z^{-1}(\alpha_x) \left(R_y^{-1}(\alpha_y) \left(R_x^{-1}(\alpha_x) \begin{bmatrix} x^s \\ y^s \\ z^s \end{bmatrix} \right) \right) \right) \tag{4.3}$$

This is the rotation that will be used to align the sensor frame with the vehicle frame and that will be the output from the algorithm. The extra parentheses in the equation are there to emphasize in which order the three rotations are allowed to be performed using this convention.

The choice of convention for representing angles used throughout this report is with the range $-\pi$ to π and with positive sign corresponding to counterclockwise angles.

4.3 The two step method

The two step method for solving this type of calibration problem suggested in [1] and [9] has been adapted, separating the problem into vertical and horisontal alignment. Vertical alignment means aligning the z-axes of the sensors and the vehicle, i.e to find the rotation matrix $R_{xy} = R_x(\alpha_x)R_y(\alpha_y)$. The angle α_z has no influence on the vertical misalignment. By rotating the accelerometer and angular velocity data with R_{xy}^{-1} before it is used in the horizontal alignment, the z-axis of the sensor, z^s, is aligned with the z-axis of the vehicle, z^v, and a_x^s and a_y^s are in the plane spanned by x^v and y^v according to figure 4.6. This is made possible because of the choice of convention for rotation, z-y-x, as explained in the previous section. When the horisontal alignment is completed also the $R_z^{-1}(\alpha_z)$ matrix that aligns the x-axes is determined.

Figure 4.5 describes the structure of the algorithm where the two darker boxes corresponds to these two main steps of the algorithm.

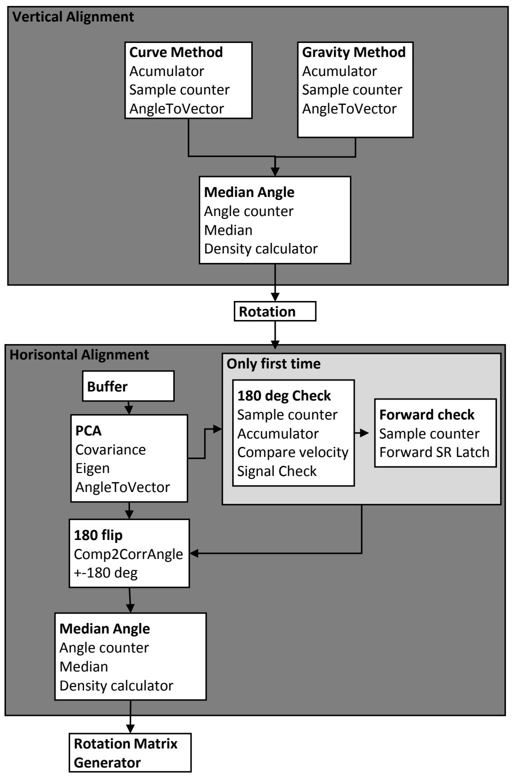


Figure 4.5: Schematic illustration of the various parts of the algorithm and in which order they are executed. The various part will be explained in more detail later in the report.

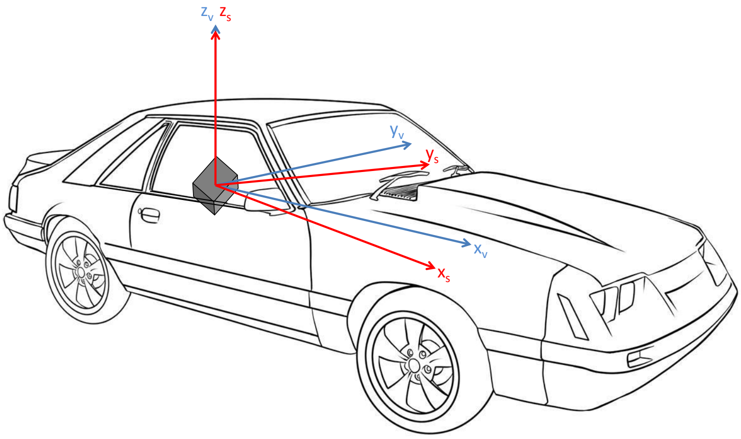


Figure 4.6: Misalignment between the coordinate system of the vehicle and the coordinate system of the sensors when rotation matrix R_{xy}^{-1} is applied.

4.4 Vertical alignment

When aligning the z-axis of the sensors with the one of the vehicle, the angles α_x and α_y are found. The angle α_z is orthogonal with the z-axis so it does not affect the vertical misalignment nor can it be determined in the process of vertical alignment.

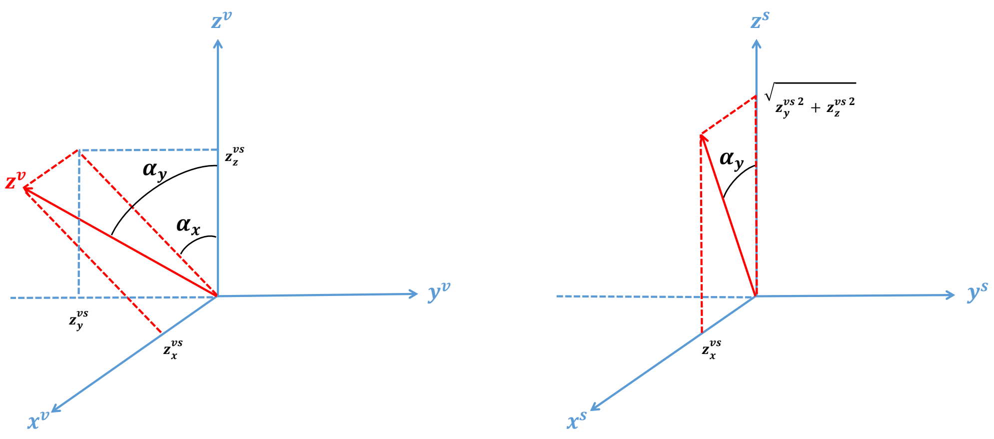
Desirable for this process is a situation where the resultant of the accelerometer or the gyroscope measurements points in a known direction in the vehicle frame. If for example readings on the rotated sensor frame is known to correspond to the vehicle frame axis z^v (let us denote this axis measured in the sensor frame z_v^s), then the misalignment angles α_x and α_y between the z_s axis of the sensor frame and z_v can be calculated using components of $\mathbf{z}^{vs}$.

This is illustrated in figure 4.7, where the sensor frame has been rotated with some angles α_x and α_y. These angles can be calculated using geometric relations.

When aligning, using the convention z-y-x, the rotation should be first around the x-axis, as described in the previous section. By looking first on the projection of the gravitation vector in the z-y-plane (see figure 4.7(a)) the angle α_x is obtained as

$$\alpha_x = atan2\left(\frac{z_y^{vs}}{z_z^{vs}}\right) \tag{4.4}$$

where $atan2$ is the four quadrant inverse tangent described in section 3.2. The angle α_y is the angle to turn around the y-axis after R_x^{-1} has already been applied so that the vector is in the x-z-plane. After rotating the vector around the x-axis, its z-component will be equal to the resultant of the y and z-components before the rotation while the x component stays the same, as is shown in figure 4.7(b). Thus



(a) (b)

Figure 4.7: Diagrams showing how the z-axes of the sensor and the vehicle frame are aligned by first rotating around the x^s and then the y^s axis. Although it is the axis z^{vs} that is fixed and the sensor frame that is rotated, for better readability, this illustration is drawn in the point of view of the sensor frame.

$$\alpha_x = atan2\left(\frac{z_x^{vs}}{\sqrt{z_y^{vs2} + z_z^{vs\,2}}}\right) \tag{4.5}$$

Two approaches for finding a situation when a vector can be measured that points in the vehicle z-direction has been investigated. The first one is to use the gravity vector when the car is standing still, a method presented in several works in the area of orientation calibration. The second one is to use angular velocities measured by the gyro when the vehicle is going around a bend.

The two techniques require different driving situations. This means that they can be used in parallel, without interfering with each other, to increase the number of situations that can be a basis for vertical alignment, effectively speeding up the calibration process.

4.4.1 Gravity method

There are two situations when the only acceleration experienced by an accelerometer inside a vehicle is the static acceleration from gravity; when the vehicle is moving with constant speed without turning or going over bumps (static state) or when it is standing still (zero motion state).

When the vehicle is in static state the resultant acceleration

$$a_{res}^s = \sqrt{a_x^{s\,2} + a_y^{s\,2} + a_z^{s\,2}} \tag{4.6}$$

should be close to g.

Even when the vehicle is moving straight ahead with constant speed, the signal a^s_{res} is strongly fluctuating. The fluctuations arises mostly from road roughness why its frequency depends both on the condition of the road and the velocity of the vehicle. There exists no limit on how low this frequency can be why the problem can not be solved by applying a low-pass filter.

Zero motion state (ZMS) can rather easily be detected using the velocity signal from the GPS module. Since the GPS signal is not absolutely accurate the velocity will never be exactly zero why a threshold is used. Figure 4.8 shows velocity and accelerations measured in a ZMS situation. The vertical lines in the figure shows the interval in which the velocity was below the threshold. We note that the accelerations within this interval is rather stable, only influenced by g and process noise, which is approximately Gaussian and does little effect to the average if the interval is long enough. This is ensured with a lower limit on the number of ZMS samples used.

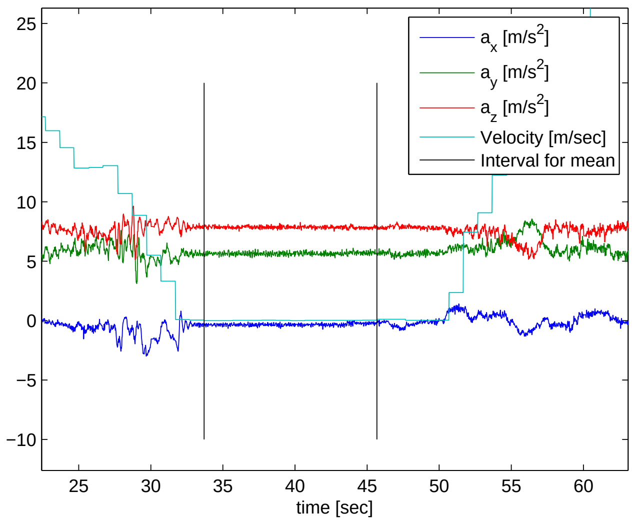


Figure 4.8: Accelerations and velocity plotted during a break - stop - acceleration sequence with sensor tilted.

In the example shown in figure 4.8, the sensor was given a small angle α_y and a larger angle α_x, resulting in a_z deviating from 9.81 and for a_x and a_y to deviate from 0.

The normalized vector from the averaged accelerations will give the z-axis of the vehicle in the sensor frame, $\mathbf{z}^{vs}$, according to

$$\mathbf{z}^{vs} = \frac{1}{N\sqrt{\sum^N {a^s_x}^2 + \sum^N {a^s_y}^2 + \sum^N {a^s_z}^2}} \begin{bmatrix} \sum^N a^s_x \\ \sum^N a^s_x \\ \sum^N a^s_x \end{bmatrix} \tag{4.7}$$

Since only the proportions are of interest when using tangents to obtain the misalignment angles, one can write

$$C\mathbf{z}^{vs} = \begin{bmatrix} \sum^N a_x^s \\ \sum^N a_x^s \\ \sum^N a_x^s \end{bmatrix} \tag{4.8}$$

where C is treated as an unknown scaling factor that is not of interest, effectively reducing the calculation complexity.

4.4.2 Curve method

When a vehicle is taking a curve it has a angular velocity around the center of the curve equal to

$$\omega_z = \frac{v}{r} \tag{4.9}$$

where v is the velocity of the vehicle and r is the radius of the curve. A 3D gyroscope placed in a vehicle, with its z-axis aligned with the z-axis of the vehicle, should when the car is taking a curve output an angular velocity on the z-axes much higher than in any other situations while the x- and y-axes should measure angular velocities close to zero. This was tested and confirmed and an example is shown in figure 4.9. This gives, according to (3.15), that the axis of rotation is close to $\hat{\mathbf{e}} = [0\,0\,1]^T$. Thus, when taking a curve the rotation axis of the vehicle frame is known.

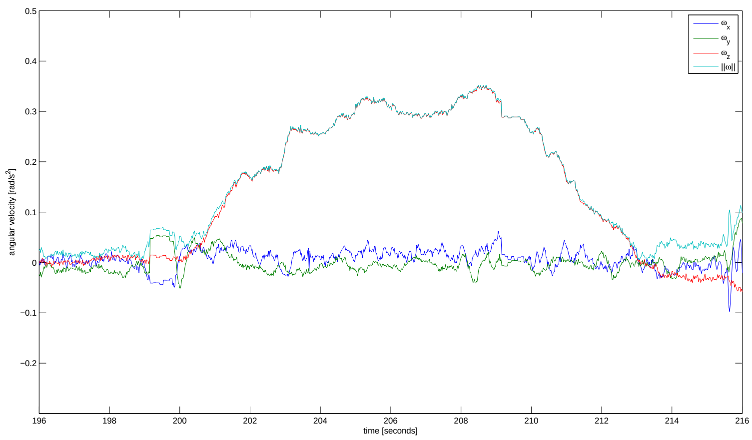


Figure 4.9: Unfiltered angular velocities and norm of angular velocities during a curve, measured with device aligned with the vehicle. Angular velocities on axes x and y are close to zero.

When the device is instead mounted so that the z-axis of the gyro is not aligned with the z-axis of the vehicle the measured angular velocity vector, $\boldsymbol{\omega}$, will still point in the direction of the z-axis of the vehicle but described in the sensor frame, $\mathbf{z}^{vs}$ (see figure 4.7(a). Thus the following equation holds

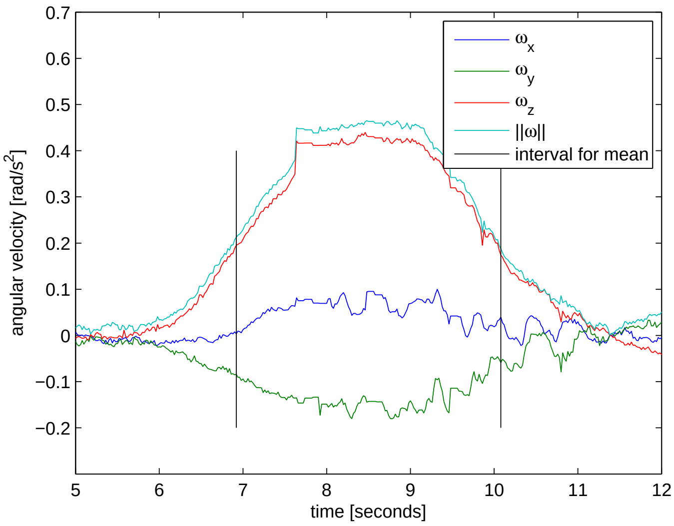


Figure 4.10: Unfiltered angular velocities during a curve measured with device misaligned to the vehicle. The relationship between the average of the three axes indicates how the sensor is leaned.

$$\mathbf{z}^{vs} = \frac{1}{\sqrt{\omega_x^{s2} + \omega_y^{s2} + \omega_z^{s2}}} \begin{bmatrix} \omega_x^s \\ \omega_y^s \\ \omega_z^s \end{bmatrix} \tag{4.10}$$

Two different approaches was investigated for using this relation in order to find the vector $\mathbf{z}^{vs}$; the average method and the LP-filter method.

Average method

When a vehicle is taking a curve its angular velocity will not remain constant but over the interval of the curve, the proportions of the averages should depend mostly on how the device is angled. If the sensor were aligned with the vehicle the average of angular velocity around axes $x^s = x^v$ and $y^s = y^v$ is near zero. Even if the car heels out of the curve due to centripetal acceleration, angular rate from going from no heel to heel will be canceled out in the average by the opposite angular rate when the car goes from heel to no heel.

Figure 4.10 shows measured angular velocities during a curve when the device was mounted with a misalignment of $\alpha_x \approx 162\,^\circ$ and $\alpha_y \approx -7\,^\circ$. Since the orientation of the device is completely unknown before calibration the restriction for detecting a curve is set on the magnitude of the angular velocity vector, $||\boldsymbol{\omega}||$, rather than on one of the components.

$$||\boldsymbol{\omega}|| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \geq ||\boldsymbol{\omega}||_{min} \tag{4.11}$$

In figure 4.10 angular velocities and the resultant $||\boldsymbol{\omega}||$ is plotted, as well as the start and stop of the interval in which the above restriction was met for an example threshold $||\boldsymbol{\omega}||_{min}$ of 0.2 rad/s. The vector $\mathbf{z}^{vs}$ can be calculated using

$$\mathbf{z}^{vs} = \frac{1}{N\sqrt{\sum^N {\omega_x^s}^2 + \sum^N {\omega_y^s}^2 + \sum^N {\omega_z^s}^2}} \begin{bmatrix} \sum^N \omega_x^s \\ \sum^N \omega_x^s \\ \sum^N \omega_x^s \end{bmatrix} \tag{4.12}$$

This vector can not be used directly to calculate the misalignment angles as in the gravity method since the vector might point either in the $+g$ or in the $-g$ direction. This will depend both on if the device is turned upside down or not, and if a left or right curve was taken since they will yield opposite sign on $\boldsymbol{\omega}$. These two cases can not be distinguished between. In order to know if the vector obtained is the desired $\mathbf{z}^{vs}$ the vertical misalignment angles are calculated as explained earlier in this section, whereby the matrix R_{xy}^{-1} is constructed and applied to a snapshot of the current accelerometer readings. If the z-component of the transformed acceleration is positive the correct vector $\mathbf{z}^{vs}$ has been found. In the case of negative, the vector needs to be multiplied with -1 before calculate the angles again. The z-component of the transformed acceleration vector should ideally be 9.81 or -9.81. Since it is just a snapshot the magnitude is highly dependent of both measurement and process noise, but its sign will not be changed by the noise.

LP- filter method

The second approach investigated was to use a FIR low pass filter to the angular velocity measurements. Figure 4.11 shows angular velocities in the same situation as in figure 4.10 but after applying the LP-filter. Instead of averaging the samples within the interval, the maximum of $||\boldsymbol{\omega}||$ is found and ω_x, ω_y, ω_z are used at that instance. As soon as $||\boldsymbol{\omega}|| > ||\boldsymbol{\omega}||_{min}$ the following samples are checked until $||\boldsymbol{\omega}||$ is no longer increasing. This might be a local maximum, but if the cutoff frequency of the filter is low enough this is not a big issue. In the example in figure 4.10 the cutoff frequency is $0.1\omega^{Ny}$. With $F_s = 50Hz$ this means 2.5 Hz.

When comparing this method to the average method, one should consider that the loss of information can be rather large when using such a filter, and also the processor power demanded. To achieve as good results as shown in figure 4.11 as many as 30 filter coefficients are needed, meaning that if time domain filtering is used, for every filtered sample, 30 multiplications and 30 additions are required. Frequency filtering means Fourier transforming and inverse Fourier transforming the data before and after filtering, which would be even more computationally heavy.

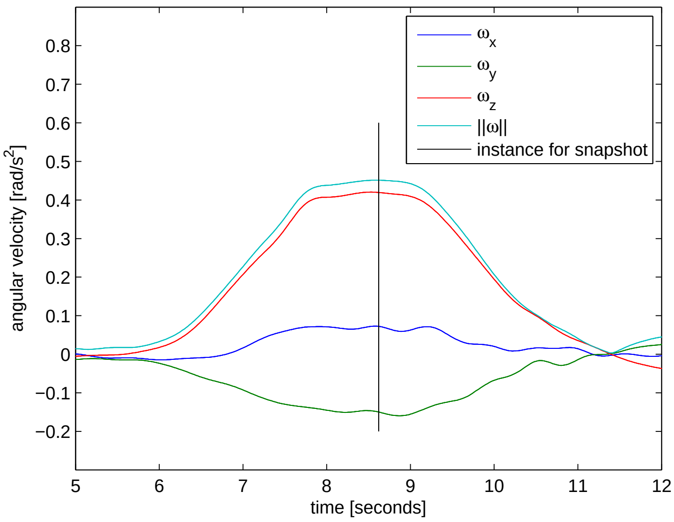


Figure 4.11: Low-pass-filtered angular velocities and norm of angular velocities measured when taking a sharp curve.

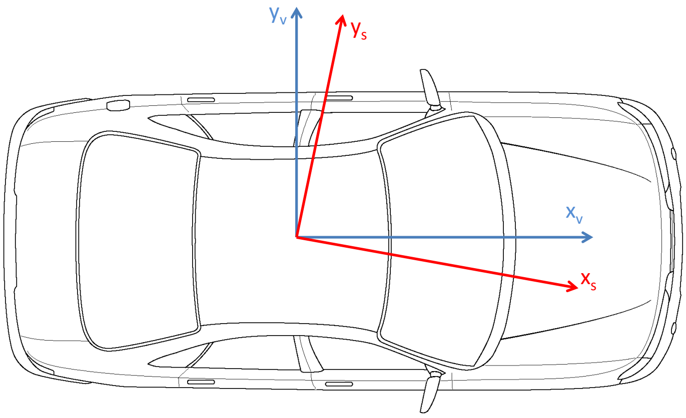


Figure 4.12: Two dimensional misalignment between the coordinate system of the vehicle and the coordinate system of the sensors.

4.5 Horisontal alignment

When the vertical alignment is completed the angles α_x and α_y are found and the remaining task is in two dimensions instead of three, namely to align the x- and y-axes of the sensor (x^s and y^s) with the x- and y-axes of the vehicle (x^v and y^v), see figure 4.12.

As for the vertical alignment a situation where the resultant of the accelerations or the gyro measurements points in a known direction of the vehicle frame is desired. There are two obvious situations where this is fulfilled; when the vehicle is taking a curve while in constant speed and the resultant of the measured accelerations points in the direction of the y^v (due to centripetal acceleration) or when the vehicle is accelerating/decelerating without turning and the resultant of the measured accelerations points in the direction of x^v. For the first case, to ensure that the speed is constant the GPS could be used. However, the sampling rate for the GPS is rather low (1 Hz) and there might be undesired accelerations in between the samples that are not registered. On the other hand, by also transforming the angular velocity vector $\boldsymbol{\omega}$ the rotation ω_z corresponds to the turning of the vehicle, it is easy to detect wether the vehicle is driving straight forward. When the vehicle is driving straight forward the resultant of the accelerations in the two dimensional space have approximately the same direction although the magnitude differs.

4.5.1 Expurgation of data

The horizontal alignment is based on data sampled when the vehicle is driving straight forward. In order to ensure that the samples used fulfills this criteria samples with a low value on the angle rate around the z-axis are selected as the basis for this alignment. When the accelerations in both directions is small the signal to noise ratio is low which means that a_x^s and a_y^s does not depend on how the sensor is aligned with respect to the vehicle. Since a negative and an positive acceleration will equal out the sum $a_x^s + a_y^s$, instead absolute values are used in the restriction

$$|a_x^s| + |a_y^s| \geq a_{tot}^{min} \tag{4.13}$$

In figure 4.13 the accelerations a_x^s and a_y^s from a test drive of 3000 samples when the device was vertically aligned with the vehicle and with a horisontal misalignment of 90° are plotted. Blue dots corresponds to data before expurgation while dots in green corresponds to the data when ω_z was below 0.005 rad/s. Shown in red are data when also threshold on $|a_x^s| + |a_y^s|$ of 1.5 m/s^2 was applied and in black is data from when also the velocity was below 20 km/h.

4.5.2 Principal Component Analysis

Even after expurgation the direction in which the resultants of accelerations point vary as can be seen in figure 4.13. In order to estimate the true direction the data is processed using the statistical method Principal Component Analysis (PCA) explained in section 3.3, adapted from [9]. When the number of samples collected is sufficient PCA is applied to this data set ($[a_x^s \ a_y^s]$). The procedure for PCA is to first remove the mean of all samples in the set from every sample in the set. The covariance matrix of a_x^s and a_y^s is then calculated according to (3.19) and the eigenvectors and eigenvalue of this matrix is determined using (3.21) and (3.22). The output from the method is the eigenvector $\boldsymbol{v}$ that represents the direction in which the variance is maximal (the line that minimizes the

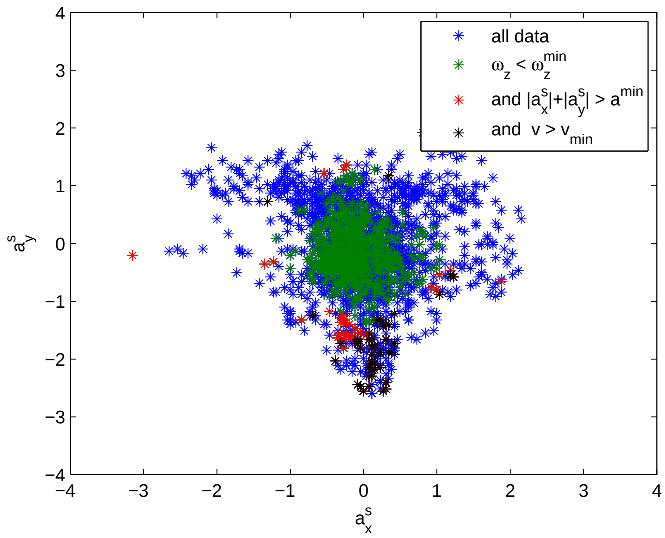


Figure 4.13: Example of how expurgation can improve a data set. Refer to the text for details.

squared distances to the data) and hence the direction of movement of the vehicle. The angle between the x^s axis and this direction can be calculated geometrically according to

$$\alpha_z = atan2(v_2,v_1) \tag{4.14}$$

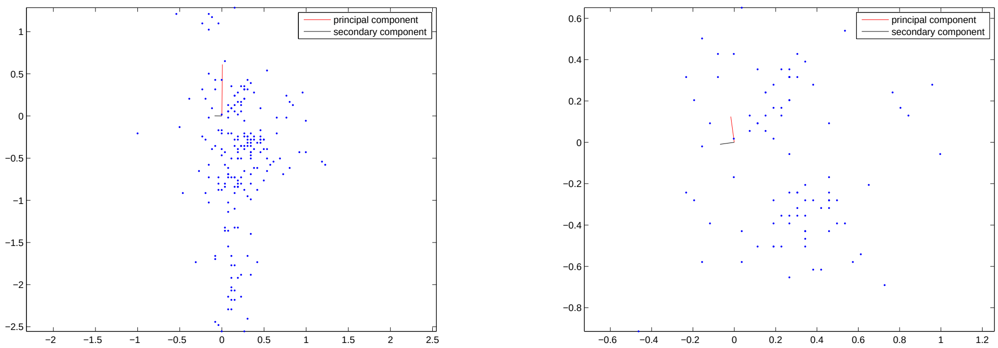
Eigenvalue ratio

The eigenvalues provide information on how large the variance is in the directions of the corresponding eigenvectors and the ratio between the highest and the lowest eigenvalue describes the distribution of data in the coordinate system to which the eigenvectors are base vectors. This is referred to as eigenvalue ratio (EVR) and a high ratio corresponds to an elongated distribution (high correlation) whereas a low value corresponds to a circular distribution (low correlation). In figure 4.14 the difference of a data set with high EVR respectively low EVR is visualized.

The EVR is used to control if the eigenvector $\boldsymbol{v}$ and therefor also α_z is reliable and if the EVR is too low the angle is discarded.

4.5.3 Integration method

The velocity in the forward direction of the vehicle (v_x^v) is corresponds to the velocity provided by GPS. Another investigated method for aligning x^s with x^v is to analyze the similarities between the velocity obtained from GPS and velocities in different directions in the sensor frame; when the coordinate systems of the vehicle and the device concide,



(a) Two dimensional data set with high EVR. (b) Two dimensional data set with low EVR.

Figure 4.14: Two dimensional data set with high respectively low EVR.

the velocity in the direction of the x-axis of the sensor frame v_x^s will be equal to the velocity obtained from GPS while v_y^s equals to zero. The velocity in the sensor frame v^s is ideally given by integration of the accelerations $\mathbf{a}^s$ and for some α_z the following equation holds.

$$v^{GPS} = v^v = R^{-1}(\alpha_z)v^s \tag{4.15}$$

However, a small offset in the acceleration quickly becomes a large error in velocity when integrating and both forward and backward Euler integration causes substantial drift. Hence, instead of the actual value the similarities of the different velocities are analyzed by covariances. The covariance between the calculated v_x^v and the velocity from GPS is calculated using equation (3.17). The angle α_z corresponding to the highest covariance is the angle of misalignment. In order to find this angle an iterative method is required which makes the method rather slow if a high accuracy is desired, especially since covariance is computationally heavy. Another problem with the method is that since the two coordinate systems are not necessarily aligned the only situation where the initial value for the integration is reliable is when the velocity is zero which may not occur frequently enough. Additionally, the update rate of the GPS velocity signal is low (1 Hz), making it unsuitable for comparison with processed acceleration data.

4.5.4 Direction check method

The PCA method does not make a difference between an angle and its opposite ($\pm 180°$); the principal component can point in either the forward or backwards direction of the vehicle. Hence, the output from the method varies between the true estimation of the angle and its opposite and the incorrect angles need to be flipped in order to be a valuable approximation. As basis for determination of forward direction an interval is selected where it is ensured that the vehicle is driving forward and that the velocity is strictly increasing or decreasing. The accelerations are rotated with the α_z calculated with PCA and the sign of the acceleration in x-direction is compared to the sign of the velocity change. If the sum of the accelerations is negative during this interval although the velocity is increasing the estimated angle is assumed to be 180° incorrect and its opposite is set as a reference. If the sign of the sum of the accelerations instead are the same as the sign of the velocity change the angle is assumed to be correct and the angle itself is used as a reference. When an interval that fulfills all requirements and an angle is found this angle serves as reference α_z^{ref} throughout the rest of the calibration process. The future outputs from PCA are compared to α_z^{ref} and if the difference between the current output and the reference is larger than 90° a flip is made. This flip is simply an addition or subtraction of 180°.

Ensuring forward movement

The velocity provided by GPS is strictly positive and hence does not give information about whether the vehicle is driving forward or backwards. However, this information is important in order to correctly detect the horisontal rotation (α_z) since if the alignment is done when the vehicle is driving backwards it will be 180° incorrect. The direction of movement is considered to be forward when the vehicle has been driven for a specific number of minutes or if the velocity has exceeded a certain value.

4.6 Median method

To compensate unrobustnes in vertical and horisontal alignment a statistical method using several calculated angles is used. Average of angles is not useful since if the misalignment angle is small, one poor result would affect the average severely. If instead the median of angles is used the magnitude of the angle to be calculated does not matter. Also a method for deciding when the angle is good enough is required. A variation of the probability density function is used; every time a new angle is added to the list of angles the median of all angles is calculated and a bin is created, some angles wide, centered at the median. The number of angles that are within the bin will be referred to as the density. The density can then be used to control when to stop the calibration and start running the BiFi algorithm.

Figure 4.15 shows how the median of the angles is getting closer to the expected value even though some calculated angles are less accurate. In this example the angle α_x is plotted together with the median of all previously calculated angles. Also the density is plotted, showing that the certainty of the median is steadily increasing. It is also possible to see in the figure that a greatly inaccurate value (as was experienced at times 1.8 and at 2.2) does not increase the density. In the figure is also plotted an example for the density threshold of 5 angles, meaning that in this case the α_x angle would be considered determined after 3 minutes.

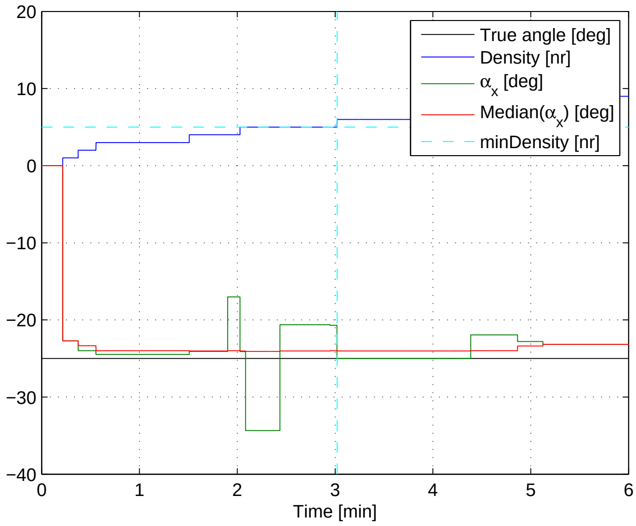


Figure 4.15: An example of current α_x, median filtered α_x and density of α_x.

5

Implementation

THE ALGORITHM was developed in the modeling and simulation software Simulink, with the advantage of good readability and ease of rapid simulations of each step during development. The device is programmed in C-code which was automatically generated from the the Simulink model. Throughout this chapter pseudo code is used to explain the various steps of the algorithm.

5.1 Two step method

As explained in section 4.3 the algorithm consists of two main parts; vertical and horisontal alignment. The pseudocode in algorithm 1 illustrates how the main calibration program uses these two functions. The variable `fistVerticalAlignmentDone`, which is set in `VerticalAlignment`, is used to control that `HorisontalAlignment` does not start until one verticalAlignment is completed, i.e., there exists at least one set of angles α_x and α_y to transform acceleration and gyro data into the 2-D-plane. When this is true, VerticalAlignment and HorisontalAlignment are run simultaneously until finished. See section 5.4 for explanation on how to decide if the alignment is finished.

Algorithm 1: Pseudocode showing the principals of the implementation of the two step method.

```
for Every sample do
    if not verticalAlignmentDone then
        step VerticalAlignment;
    end
    if firstVerticalAlignmentDone and (not horisontalAlignmentDone)
    then
        rotate acceleration vector with R(αx,αy);
        step HorisontalAlignment;
    end
    if verticalAlignmentDone and horisontalAlignmentDone then
        Create array with rotation coefficients [c11...c33] corresponding to
        R(αx,αy, αz) using equations (3.1) - (3.3);
        set calibrationtDone = TRUE;
    end
end
```

5.2 Vertical alignment

As explained in section 4.4 the two methods referred to as curve method and the gravity method are used to find the vertical misalignment. They are run in parallel and when a pair of new angles are available from either of these methods the new angles are sent as arguments to the median method, explained in section 4.6, to update the variables `densityX` and `densityY`, which are indicating how reliable the median of the angle is. When the density of an angle is high enough the median is no longer updated. The first time the median method is called, the flag `firstVerticalAlignmentDone` is set, which is outputted to the main program. Algorithm 2 shows how the function is implemented.

Algorithm 2: Pseudo code for the VerticalAlignment.

```
set firstVerticalAlignmentDone = FALSE;
while densityX < minDensity and densityY < minDensity do
    step CurveMethod and GravityMethod;
    if New angle available from ether method then
        set firstVerticalAlignmentDone = TRUE;
        if New angle available from curve method then
            α_x = α_x^curve;
            α_y = α_y^curve;
        else
            α_x = α_x^gravity;
            α_y = α_y^gravity;
        end
        if densityX < minDensity then
            run MedianMethod(α_x) to update densityX;
        end
        if densityY < minDensity then
            run MedianMethod(α_y) to update densityY;
        end
    end
end
```

5.2.1 Gravity method

The pseudo code for the gravity method is shown in algorithm 3. At each sample the velocity is read to check if the vehicle is in zero motion state (ZMS). While the velocity is below a threshold *maxStopVel*, the vehicle is determined to be in ZMS and the acceleration for each axis is accumulated for every sample and the number of samples, for which the vehicle is in ZMS, is counted. When the vehicle starts moving again the accumulated acceleration and the count is reset and if the count exceeded a threshold the length of the stop is considered to be sufficient and the angles α_x and α_y are calculated using atan2 as described in section 4.4.1.

Algorithm 3: Pseudocode explaining how the gravity method is implemented.

```
for Every sample do
    if GPS has lock then
        if velocity <maxStopVel then
            Add current acceleration to accumulated acceleration vector a^sum;
            Increase sample index by one;
        else
            Reset sample index and acumulated accelerations to 0;
            if Last sample was stop and sample index > minSecStop then
                set Calculate α_x and α_y by using using a_x^sum, a_y^sum and a_z^sum in
                equation (4.4) and (4.5);
            end
        end
    end
end
```

5.2.2 Curve method

The curve method, for which the implementation is described in algorithm 4, works similarly to the gravity method. Here samples of angular rate are accumulated when the norm of the angular velocity on the three axes is larger than a threshold *minOmegaTotCurve*. When the norm again goes below the threshold, if the curve was long enough, the vector Ω_{sum} of accumulated angular rates is used further and otherwise the curve is disregarded. In contrast to the gravity method, Ω_{sum}, which should be parallel to the z^{vs} vector as described in section 4.4.2, can not be used directly since it might point in the z^{vs} or $-z^{vs}$ direction. On lines 9-12 in algorithm 4 the angles are first calculated without changing sign of Ω_{sum}, and are tested by rotating the acceleration vector to see if the z-axis is pointing towards gravity or not, i.e if its value is positive or negative. If the value is negative the angles are corrected by flipping the vector Ω_{sum} and the angles are calculated again.

Algorithm 4: Pseudocode for the curve method.

for *Every sample* **do**
  Calculate the norm $||\Omega||$ of angular rate vector ;
  if $||\Omega||>$ minOmegaTotCurve **then**
    Add current angular rates to accumulated rates vector Ω^{sum};
    Increase sample index by one;
  else
    Reset sample index and accumulated angular rates to 0;
    if *Last sample was curve* **and** *sample index* $>$ minNrOfCurveSamples **then**
      Calculate α_x and α_y using ω_x^{sum}, ω_y^{sum} and ω_z^{sum} in equation (4.4) and (4.5);
      Rotate acceleration vector **a** with $\mathbf{R}(\alpha_\mathbf{x}, \alpha_\mathbf{y})$;
      if $a_z < 0$ **then**
        Multiply Ω_{sum} with -1;
        Again calculate α_x and α_y;
      end
    end
  end
end

5.3 Horisontal alignment

The horisontal alignment consists of the PCA method, the direction check method and the median method. As can be seen in algorithm 5 the direction check method is called only after that the first PCA is completed and only until a reference angle α_z^{ref} has been found. When the direction check is completed every new available α_z from the PCA method is, if the EVR indicates that the angle is to be trusted, compared to the reference angle and is corrected by 180° if needed. For a detailed explanation, see section

4.5.4. The corrected angle is then sent as an argument to the median method.

Algorithm 5: Pseudo code for the HorisontalAlignment.

```
while densityZ < minDensity do
    step PcaMethod;
    if (not directionCheckDone) then
        if firstPcaDone then
            step DirectionCheckMethod to find correct reference angle α_z^ref (see
            algorithm 7 for details);
        end
    else
        if new α_z available and eigenValueRatio > minEVR then
            if |α_z − α_z^ref| > π/2 then
                if α_z >= 0 then
                    α_z+ = π;
                else
                    α_z− = π;
                end
            end
            run MedianMethod(α_z) to update densityZ
        end
    end
end
```

5.3.1 PCA

The implemented PCA method applies the ideas presented in sections 4.5.1 - 4.5.2 and is presented in algorithm 6. For each sample the velocity, the accelerations a_x and a_y, and the angular rate ω_z (the later three has been transformed with $\mathbf{R}(\alpha_z, \alpha_y)$ to the 2D-plane) are compared with their threshold in order to exclude data from when the vehicle is not accelerating or decelerating without turning. If all restrictions are met the samples a_x and a_y are stored in arrays `pcaBufferX` and `pcaBufferY` of length *nrOfSamplesPca*. When the buffers are full the mean of all values in each buffer is removed from every value in the buffer. The covariance matrix for the two arrays are then calculated as well as the eigenvalues λ_i and the eigenvectors $\mathbf{v}_i$ of the covariance matrix. The eigenvector that has the highest eigenvalue is the first principal component, $\mathbf{v}^{fp}$, and is used to calculate the angle α_z. Also the eigenvalue ratio, which is the ratio between the first and the second pricipal component, is calculated here. When a new angle is calculated the flag `newAngleZ` is set high but it is reset on the next iteration.

The flag `firstHorisontalAlignmentDone` is set once and thereafter never reseted.

Algorithm 6: Pseudo code for the Principal Component Analysis method.

for *Every sample* **do**
    if minVelPCA $<$*velocity*$<$maxVelPCA **then**
        if $|a_x| + |a_y| >$ minAccSumPCA **then**
            if $\omega_z <$ maxYawPCA **then**
                `pcaBufferX` [`bufferIndex` ++] $= a_x$;
                `pcaBufferY` [`bufferIndex` ++] $= a_y$;
            end
        end
    end
    if `bufferIndex` $>=$nrOfSamplesPca **then**
        remove mean from buffered acceleration values;
        calculate covariance matrix $\mathbf{C}$ of `pcaBufferX` and `pcaBufferY`;
        calculate eigenvalues λ_i and eigenvectors $\mathbf{v}_i$ of $\mathbf{C}$;
        if $\lambda_1 > \lambda_2$ **then**
            $\mathbf{v}^{fp} = \mathbf{v}_1$;
            `eigenValueRatio` $= \lambda_1/\lambda_2$;
        else
            $\mathbf{v}^{fp} = \mathbf{v}_2$;
            `eigenValueRatio` $= \lambda_2/\lambda_1$;
        end
        calculate $\alpha_z =$ `atan2(` v_y^{fp}, v_x^{fp}`);`
        set `firstHorisontalAlignmentDone` = TRUE;
        set `newAngleZ` = TRUE;
    else
        set `newAngleZ` = FALSE;
    end
end

5.3.2 Direction check method

To check if the angle calculated with PCA is correct or if it is 180° wrong, the acceleration data is rotated and compared to the change in velocity. Since the velocity from the GPS module is only a magnitude, it will yield the same signal if accelerating backwards as when decelerating forwards, inconsistently with the acceleration signal. Therefor one must first be sure that the vehicle is driving forwards before comparing the velocity and acceleration. When the velocity exceeds a value *maxBackwardSpeed* or if the direction has been the same for more than *minNrOfSecSameDir* seconds the `isDrivingForward` flag is set. It is reset first when the velocity is close to zero since the vehicle might go from driving forwards to driving backwards. When the velocity is close to zero also the same direction sample counter is reset. When an interval of *minNrOfSamplesVelComp* is found, the current velocity is stored and compared to the first stored value in the interval. If the difference is large enough and if forward driving has been ensured, then the sign of the difference in velocity is compared to the sign of the average of all acceleration samples on the x-axis that has been stored during the interval. If the signs differ, 180° is added or subtracted from the input angle. The new angle, which is the output from the function, is called α_z^{ref} since it will be used as reference to correct all future angles instead of running this function for every new angle. If not all of the criteria where met at the end of an interval the sample count and the accumulated acceleration samples are reset for a new try.

Algorithm 7: Pseudo code showing in pricipal how the direction check method is implemented

```
for every sample do
    rotate acceleration a with R(α_z);
    set firstVel = velocity;
    if velocity>maxBackwardSpeed or sameDirSampleCount * T_s >
    minNrOfSecSameDir then
        set isDrivingForward = TRUE;
    else
        if velocity<minMovingSpeed then
            reset sameDirSampleCount = 0;
            reset isDrivingForward = FALSE;
        else
            sameDirSampleCount ++;
        end
    end
    if sampleCount > minNrOfSamplesVelComp then
        if |velocity − firstVel| ≥ minDiffVelComp then
            if isDrivingForward then
                if sign(a_x^sum) ≠ sign(|velocity − firstVel|) then
                    if α_z < 0 then
                        α_z^ref = α_z + π;
                    else
                        α_z^ref = α_z − π;
                    end
                else
                    α_z^ref = α_z
                end
            end
        end
        set firstVel = velocity;
        reset sampleCount = 0;
        reset a_x^sum = 0;
    else
        sampleCount ++;
        a_x^sum + = a_x;
    end
end
```

5.4 Median method

The median method introduced in section 4.6 is implemented according to algorithm 8. When a new angle is available it is added to an array of all earlier calculated angles and the median is calculated. To calculate the median will be more processor demanding for each additional angle, why a safety mechanism is applied in form of the limit `maxNrOfAngles`. The algorithm can still keep running if this value is exceeded but the new angles then overwrites the older ones.

The `density` variable used to decide when the calibration is finished is calculated according to lines 10-15 by counting the number of angles that are close enough to the median.

Algorithm 8: Pseudo code showing how the median method is implemented. The method is applied for all three angles.

```
7.1  while density < minDensity do
7.2      Do alignment algorithm;
7.3      if new angle available then
7.4          angleIndex ++;
7.5          if angleIndex < maxNrOfAngles then
7.6              angleIndex = 0;
7.7          end
7.8          angles [angleIndex ] = available angle;
7.9          Calculate median of angles;
7.10         set density = 0;
7.11         for all angles do
7.12             if angle - median < maxMedianDeviation then
7.13                 density ++;
7.14             end
7.15         end
7.16     end
7.17 end
7.18 return bestAngle = median;
```

5.5 Integration to BiFi software

The algorithm is developed modular to fit in the larger system for determination of road condition developed by Semcon, described briefly in section 1.2. Some modifications to the main program was though made and the function of the updated main program is illustrated in algorithm 9. When a calibration has once been completed it does not need to run again when the device is powered up the next time. The SD card is therefore used for non volatile storage of calibration result and it is read after power up before entering the main loop. The results consists of the flag `calibrationDone` and an array with the nine rotation matrix coefficients.

The program is written without use of interrupts in order to ensure atomicity. During development it is instead made sure that no task takes longer than the sampling period. The accelerometer and gyro signals are read first in every period, then the calibration is stepped if it is not completed and otherwise the acceleration data is transformed using the coefficients stored on the SD-card and the Bifi algorithm is stepped. Finally the program waits until the sample period is over. The first time the flag `calibrationDone` is true, the coefficients and the flag is saved to the SD-card.

Algorithm 9: Pseudo code showing how the algorithm is integrated to the larger system.

```
Initialize BiFi and Calibration;
Check SD for previous calibration result;
while device is turned on do
    if time for sampling then
        get new sensor readings;
        if calibration is done then
            if calibration result has not been stored then
                write calibration result to SD
            end
            transform accelerometer readings to vehicle frame;
            step Bifi algorithm;
        else
            step Calibration algorithm;
        end
    end
end
```

6

Testing and tuning

6.1 Tunable parameters

The implemented algorithm includes several parameters possible to tune in order to improve the algorithm or to adjust it for a desired behavior. In general, the choice of parameter setting is a compromise between good accuracy and short time of completion. The parameters are presented in 6.1 together with their purpose and explanations of their influence on the algorithm. The parameters are divided into six groups: ”Signal processing”, ”Curve method”, ”Gravity method”, ”PCA”, ”Direction check method” and ”Evaluation” based on where in the algorithm they are used.

It is important to note that all parameters are highly correlated to each other. For example, low accuracy of the calculated angle α_z due to a low number of samples used for PCA (*nrOfSamplesPCA*) can be compensated for by setting a high value on the parameter *minEVR*.

Several of these parameters are lower limits meant to ensure that the time, number of samples or the value of the accelerations/velocities is sufficient to ensure that the set is suitable for usage in curve method, gravitation method or PCA. A high value of these parameters means higher certainty on each calculated angle while a lower value means that more angles can be calculated within the same timespan. The parameters that are upper limits works similarly but reversed; a low value means higher certainty and a high value that more angels are generated.

The parameters used in the direction check method are critical but harder to tune; for example, since forward motion is much more common than backward motion the parameter *maxBackwardSpeed* will in most cases only affect the time needed for the direction check and not the resulting angle. If, though, the vehicle happens to be moving backwards while the direction check is preformed, it is crucial that this is detected since otherwise the resulting angle will be 180° incorrect. Therefor the limit needs to be sufficiently high to exclude the possibility that the vehicle is driving backwards when α_z^{ref} is calculated.

The evaluation parameters are used to set the requirements that must be met on the result, i.e. to decide when the calibration is done. Hence, these parameters affect neither the certainty on each calculated angle nor the number of angles calculated within a certain time.

Parameter	**Description**
Signal Processing	
filter_coeffs [nr]	- Number of filter coefficients used for the acceleration preprocessing filter. - Improves the filtering, especially if a low cutoff frequency is chosen.
filter_cutofffreq [Hz]	- Cut-off frequency in filter for the acceleration preprocessing filter. - A low frequency removes more undesired noise from road roughness, but might also remove desired information.
Curve Method	
minNrOfCurveSamples [nr]	- Lower limit on number of samples with high angular rate for curve samples to be used for alignment. - A high value improves the accuracy of the estimated $\mathbf{z}^{vs}$.
minOmegaTotCurve [rad/s]	- Lower limit on total angular velocity required for the vehicle to be consider to be in a curve. - A higher value makes the method less sensitive to noise in the angular velocity signals.
Gravity Method	
minSecStop [sec]	- Lower limit on the number of seconds in stationary state to align the z-axis with the gravitational acceleration. - Ensures that the interval of the stop is long enough to yield a trustworthy mean value.
maxStopVel [km/h]	- Upper limit on the velocity to classify the vehicle to be in stationary state. - Prevents small errors in velocity to falsely suggest that the vehicle is in zero motion state.
PCA	

maxVelPCA [km/h]	- Upper limit on the velocity for samples included in PCA. - Eliminates noisy samples caused by high velocity.
minVelPCA [km/h]	- Lower limit on the velocity for samples included in PCA. - Eliminates samples where the vehicle is not moving and the measured accelerations are only noise, i.e where the signal to noise ratio is low.
maxYawPCA [rad/s]	- Upper limit on the angular velocity around the z-axis, for samples to be used in PCA. - Eliminates samples where the resultant acceleration depend also on the centripetal acceleration.
minAccSumPCA [m/s^2]	- Lower limit on the total acceleration for samples included in PCA. - Eliminates samples with low signal to noise ratio.
nrOfSamplesPCA [nr]	- Number of samples used for PCA. - A higher value should improve accuracy, but also increase the time needed before the angle can be calculated.
minEVR [ ]	- Lower limit on EVR for including angels generated from PCA in median. - Prevent poor z-angles angles to corrupt the result.
Direction Check Method	
minDiffVelComp [km/h]	- Lower limit on velocity increase/decrease in detection of forward/backwards direction. - Decreases the risk of misleading result due to the lower update rate of the velocity signal.
minNrOfSamplesVelComp [nr]	- Lower limit on number of samples used to detect forward direction. - Decreases the risk of a misleading mean value of the accelerations during the interval.
minNrOfSamplesSameDir [nr]	- Lower limit on number of samples without stop to ensure forward movement. - A high value increases the certainty when deciding that the vehicle is moving forwards.
minMovingSpeed [km/h]	- Lower limit on velocity to ensure that the direction of movement is not changed. - Lower the risk of inaccuracy in the velocity signal to cause a changed in movement direction to be missed.

maxBackwardSpeed [km/h]	- Upper limit on backwards velocity to ensure direction of movement. - A high value increases the certainty when deciding that the vehicle is moving forwards.
Evaluation	
minDensity [nr]	- Lower limit on the number of angles needed in the median bin to consider the calibration to be finished. - Ensures that the median of the calculated angles is representative.
maxMedianDeviationX [rad]	- Defines the width of the median bin used to ensures that the median of the calculated angles is representative. - A low value means high accuracy is required.
maxMedianDeviationY [rad]	- Defines the width of the median bin used to ensures that the median of the calculated angles is representative. - A low value means high accuracy is required.
maxMedianDeviationZ [rad]	Defines the width of the median bin used to ensures that the median of the calculated angles is representative. - A low value means high accuracy is required.

Table 6.1: Table with parameters included in the algorithm and explanations on their purpose and effects.

6.2 Test drives

To choose a suitable setting for the parameters, tests were performed by mounting a hardware in a test vehicle (Volvo V70 diesel) and log data while driving. The hardware was programmed so that it only read the sensors and wrote the raw data to the SD-memory and was positioned with an appliance with a known misplacement as seen in figure 6.1 to enable comparison between the output and the expected result. The tests, used as a basis for analysis to choose the optimal parameter setting, were performed by unspecified driving in urban environment in central Gothenburg. Four different sets of log data were created, each set containing data corresponding to 39 minutes of driving. To validate the result in other environments tests with the same set up but while driving on gravel roads and on paved countryside roads between locations Gråbo and Sollebrunn north-east of Gothenburg were performed.

(a) (b)

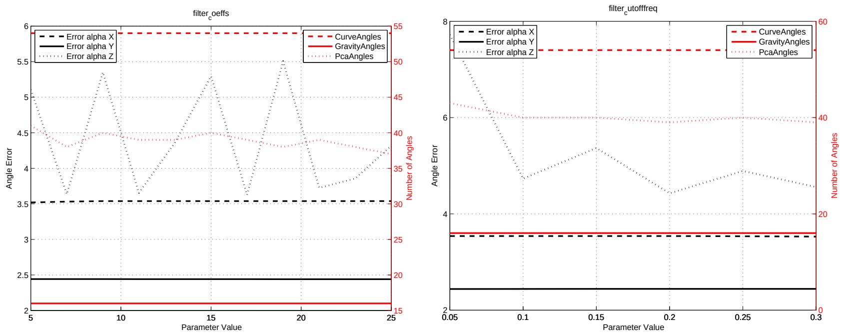
Figure 6.1: Test set up with misplaced device with known misalignment angles.

6.3 Analysis of parameter values

The collected data from the performed tests were used as input to a model of the algorithm in Simulink. To analyze the influence of the different values of the parameters the model was simulated with a standard set of parameters, presented in the first column in table 6.2. The behavior of the algorithm was then simulated with each parameter changed one at a time and the output angles were compared to the ones of the true misalignment. Also the number of angles generated from the algorithm were recorded for every value of each parameter. For every parameter setting, the average of the errors and of the number of angles from the four data sets was calculated and are presented in figures 6.2 - 6.5. These figures were used as a basis for the analysis of the parameters. For each parameter a value is chosen that yields an adequate small error on the angles while providing a sufficient number of angles. The evaluation parameters does not affect the certainty or the number of angles and are therefore not included in the figures. Since the parameters included in the direction check method only need to be set lower/higher than the point where they affect the result neither these are presented in the figures.

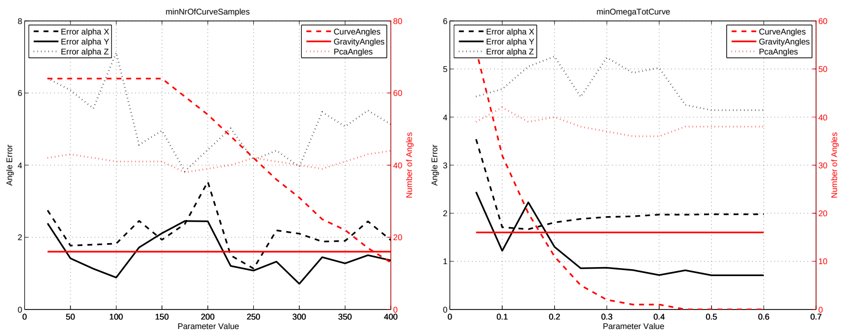
In figures 6.2 - 6.5 it is also possible to see that the parameters included in the gravity method and the curve method, which mainly affects α_x and α_y, also affects the result on α_z, but the case is not the same the other way around.

As previously mentioned the parameters are correlated. This means that to choose a value of a parameter based on the information contained in the figures exclusively does not necessarily generate a optimal result. However, the figures gives a hint on how to set



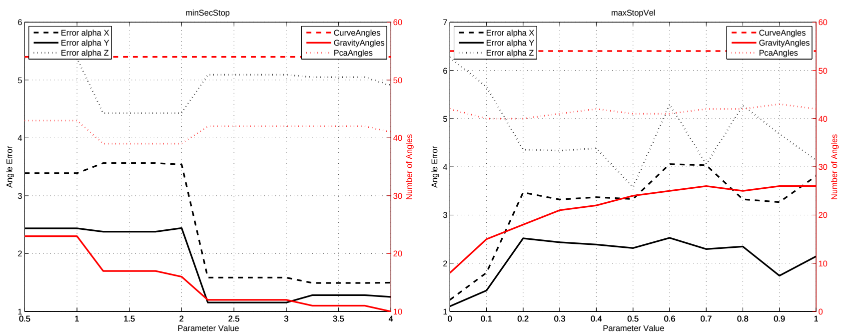
(a) (b)

Figure 6.2: Effect on angular certainty and number of angles due to changed values of filter parameters. All values are averages of the four tests.



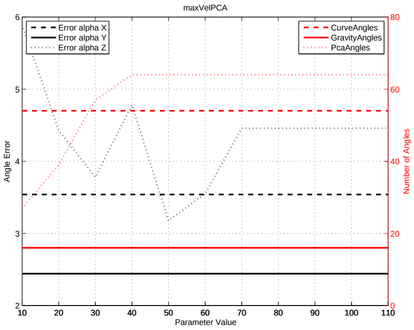
(a) (b)

Figure 6.3: Effect on angular certainty and number of angles due to changed values of parameters in curve method. All values are averages of the four tests.

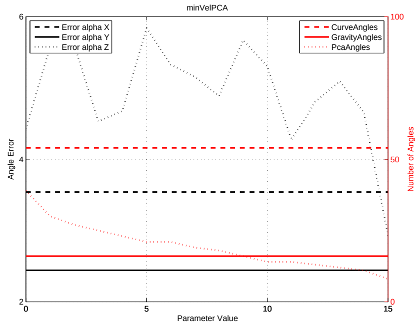


(a) (b)

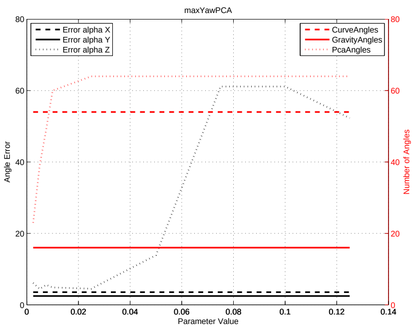
Figure 6.4: Effect on angular certainty and number of angles due to changed values of parameters in gravity method. All values are averages of the four tests.



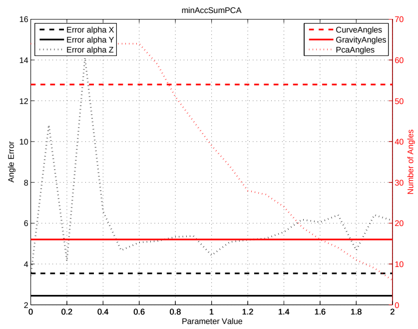
(a)



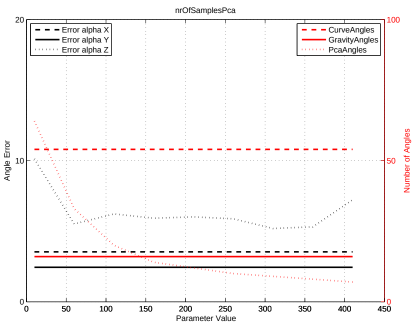
(b)



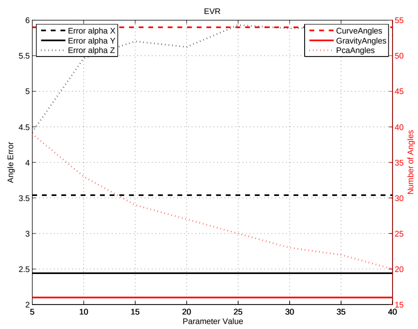
(c)



(d)



(e)



(f)

Figure 6.5: Effect on angular certainty and number of angles due to changed values of parameters in PCA. All values are averages of the four tests.

the parameters. The analysis of figures resulted in the parameter setting presented in the second column in table 6.2. When this setting was used in simulation it was found that further adjustments could be done. In the third column in the table the final choice of setting is presented.

Parameter	**Standard**	**Post analysis**	**Final**
minNrOfCurveSamples	X	X	X
minOmegaTotCurve	X	X	X
maxStopVel	X	X	X
minSecStop	X	X	X
minDensity	X	X	X
minVelPCA	X	X	X
maxVelPCA	X	X	X
maxYawPCA	X	X	X
minAccSumPCA	X	X	X
minNrOfSamplesVelComp	X	X	X
minDiffVelComp	X	X	X
minNrOfSecSameDir	X	X	X
minMovingSpeed	X	X	X
maxBackwardSpeed	X	X	X
nrOfSamplesPca	X	X	X
minEVR	X	X	X
filter_coeffs	X	X	X
filter_cutofffreq	X	X	X
maxMedianDeviationX	X	X	X
maxMedianDeviationY	X	X	X
maxMedianDeviationZ	X	X	X

Table 6.2: Parameter settings in different stages of the tuning process; the standard setting used for analysis, the setting chosen after analysis and the final setting.

7

Result

The output angles from the simulations run with the final parameter setting presented in table 6.2 can be seen in figure 7.1. In the figure the true misalignment angle on each axis is shown as a constant black line together with the converging median of the calculated angles, one line for each test drive. The vertical lines in the figure represents the times for when the algorithm is considered finished according to the evaluation parameters explained in section 6. It is from figure 7.1 possible to deduce that the results are significantly better for the alignment around the x and y axes than in around the z axis regarding both computation time and remaining error.

When the algorithm is simulated with data collected on gravel roads the results are less satisfying, as can be seen in figure 7.2. The vertical alignment works fine also in this environment but in four out of five tests, the horisontal alignment has not been completed within the 39 minutes. When the simulations are run with data from country road driving the results are even less satisfying. The horisontal alignment works even worse than on gravel roads and is not finished in any of the tests but for this data also the vertical alignment worked unsatisfactory. The results from simulations using data from countryside road driving can be seen in figure 7.3.

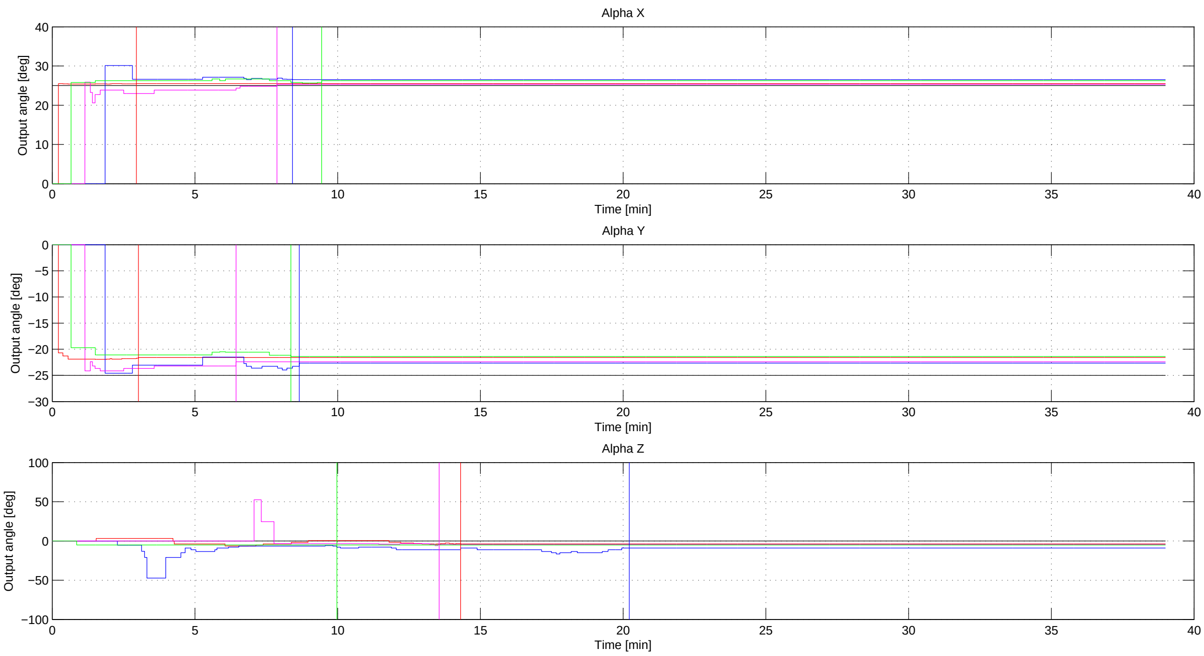


Figure 7.1: Output angles from simulation with data from urban driving.

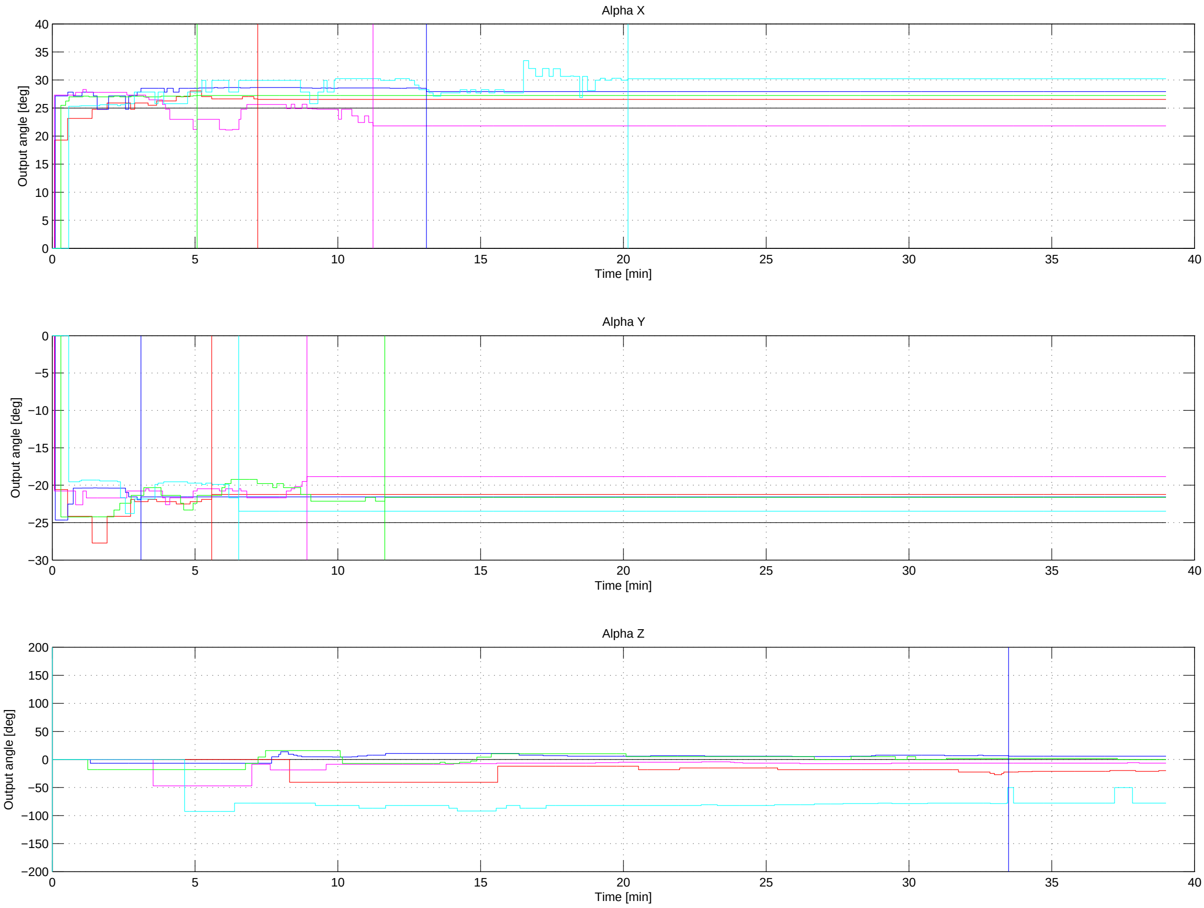


Figure 7.2: Output angles from simulation with data from driving on gravel roads.

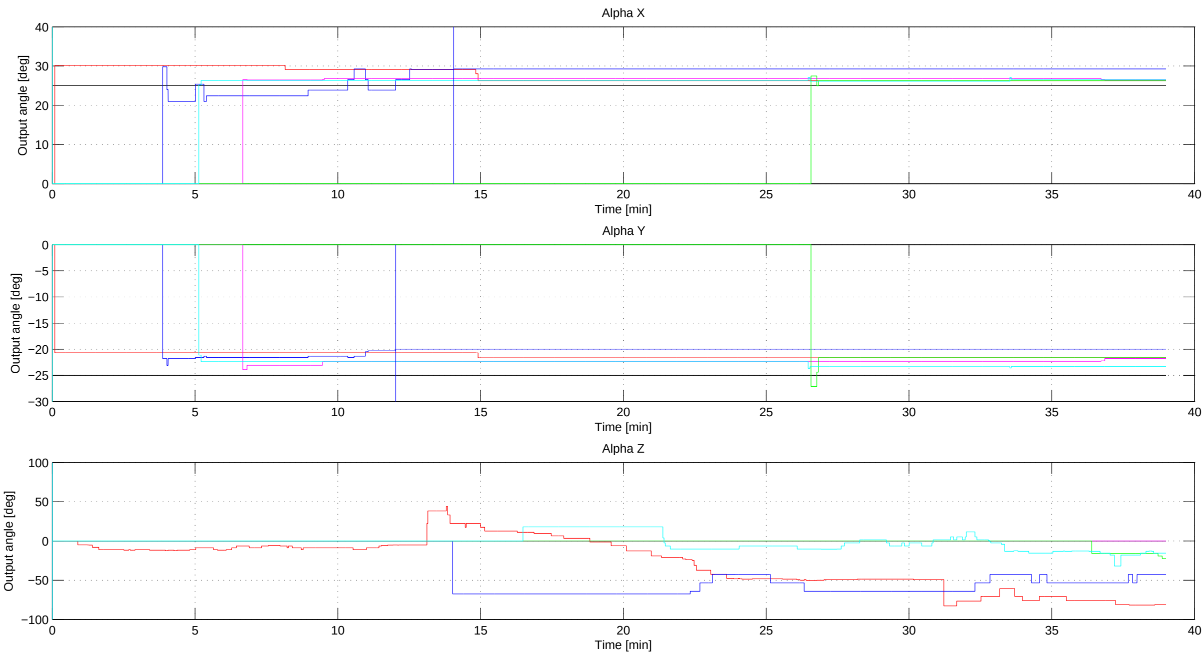


Figure 7.3: Output angles from simulation with data from driving on high roads.

8

Discussion and conclusions

In this chapter the results presented in section 7 are examined. The reliability of the results are discussed as well as the reason why the results from simulation with data collected in different environment differ. Further, factors that might have affected the results are presented and possible means for improvement are suggested. The chapter is concluded with an evaluation of the whole project and its outcomes.

8.1 Difference between results in different environments

As mentioned in section 7 the results from simulation with data collected on gravel roads and on country roads are significantly worse than simulations with data collected in urban environment.

The results from simulation with data collected on gravel roads (presented in figure 7.2) shows that few data sets that fulfills the requirements for usage in horisontal alignment are found and hence few angles α_z are calculated. This result is expected since on this kind of roads straight sections are not as common as in urban driving, and also when the road is straight, the driver might not be able to hold the steering wheel as still, due to the rougher road surface, why the angular rate ω_z might stay above its threshold. Another possible explanation for the fewer angles recorded is that rougher road means more fluctuations on the acceleration signals possibly yielding a data set with lower EVR. Angles with EVR below its threshold are not included in the median and hence not observable in the figures.

The results from simulation with data collected on country roads (presented in figure 7.3) shows that on this kind of roads also vertical alignment is difficult. This is a result of the fact that on country roads there might be few turns that yields a sufficient angular rate to be used in the curve method and few stops that enables performing the gravity method. The fact that also the horisontal alignment suffered difficulties in these tests might be a result of an increased level of noise on the gyro readings in higher velocities. Another possible explanation is that the velocity is kept rather stable, leaving out distinct accelerations/decelerations, which results in unreliable data sets representing sequences that can be used in PCA. A possible solution would be to increase the threshold on the total acceleration but that would instead result in longer time to finish calibration in this environment. As can bee seen in the lower pane of figure 7.3, in three of the five test drives, the first angle α_Z is not estimated until after a substantial time has passed (14, 17 and 37 minutes) and thereafter new angles are calculated more frequently. This suggests that the direction check method has been the bottleneck rather than the PCA method, and for the same reason that the velocity is kept steady for a long time, preventing comparison between average acceleration and velocity change.

In general, a main difference between the urban and the other two driving environments is that in the urban environment the vehicle is often completely stopped for several seconds or more at traffic lights, and acceleration from zero speed is thus also more common, providing data suitable for usage in the gravity method as well as for the PCA method and the direction check method. The tests on country roads and gravel roads where performed as 39 minutes of continuous driving sometimes without a single stop. Since the algorithm is aimed to be run in post delivery vehicles on rout, this driving pattern is not the most representative. If instead in the tests the vehicle would be frequently parked for a few seconds simulating mail delivery, the result might look more as the results from the urban environment test.

8.2 Differences between simulations and reality

A difference between reality and simulations is that when data is collected during the tests a log program is used that writes the data to a SD-card. This writing process takes time (t_w) and interrupts the sampling. To compensate for the resulting gap of data the sampling rate after writing is set to maximum until the amount of samples is up to date, going on for t_{ms}. The gap is then filled with the data sampled in retrospect which results in a square shaped data in the interval $t_w + t_{ms}$. The interval between the writings is in the test set to ten seconds but the actual writing time is unknown. This means that to what extent of the square data interval that is consisting of t_w respectively t_{ms} is unknown which makes it difficult to minimize the effect of this behavior. The described phenomena is visualized in figure 8.1, where the curve is the acceleration on one axis and the vertical lines represents the start and end of the interval $t_w + t_{ms}$. The square shape on the acceleration is found on accelerations on all axes as well as on the data from the gyro. This behavior influences the results from the simulations although exactly what the effects of this corruption yields is unknown.

8.2.1 GPS lock

For the data sets used in simulation, the logging was not allowed to start until the when the GPS module signals that it has a lock on the required amount of satellites, and in

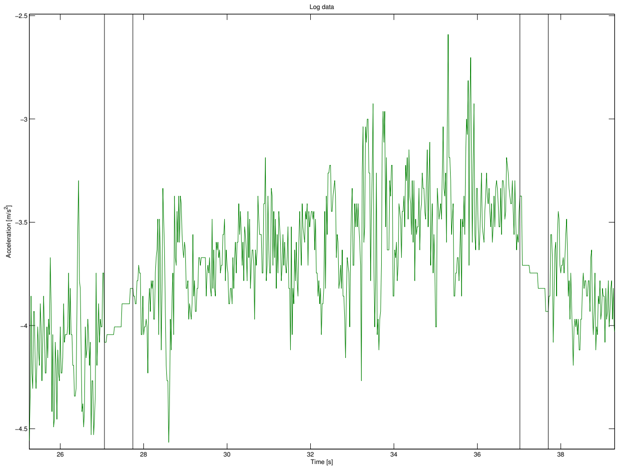


Figure 8.1: Square shaped data caused by compensation for sampling interruption due to writing data to SD-card.

table **??** and figures 7.1 - 7.3 the time 0 corresponds to when the GPS first got locked. A more informative approach would have been to store data also before the GPS has got a lock, making time 0 correspond to the instance when the device was powered up. This would be preferable also because not all of the methods in the algorithm uses GPS information. No proper statistics of the time needed for the GPS to get a lock has been established but it has been noted that in urban environments the time varies in the range of one to five minutes.

8.3 Result and adjustments

The algorithm is designed such that adjustments are easily done dependent on the desired behavior using the tunable parameters. At this point the parameters are tuned to get the best possible result from simulation run with data collected in urban environments. The BiFi program is used in postal cars which does not necessarily drive in urban environments during a sufficiently long period to complete the calibration, why a different parameter setting based on an analysis of driving in all kind of environments is motivated. The fact that the final parameter setting was chosen based on analysis of data collected exclusively in urban environment can also be the reason for the miscalculated reference angle that caused an incorrectly flipped angle α_z in simulation with data collected on gravel roads which implies that the robustness is deficient.

8.4 Compromise between computation time and accuracy

In addition to the parameters presented in section 6 it is possible to decide whether the PCA calculations should start from the moment the first set of x- and y angles are calculated or from when the hosrisontal alignment is considered completed. The main basis for this decision is the preferable ratio between time and accuracy; to start with PCA from start will make the algorithm faster but generate a less accurate result while the other alternative will make the algorithm slower but generate a more accurate result.

The intended usage of the algorithm is primarily within in the BiFi project where the algorithm would be run once and not again until the hardware is moved. If the time for the algorithm to be completed is more than 30 minutes the loss of collected data that day might be to severe and the algorithm might as well take the whole driving time that day. If, however, it is feasible to achieve a completion time lower than 30 minutes the philosophy is instead "the faster the better". Since, according to the results from simulation run with data collected in urban environments, it is feasible to reach a completion time within 30 minutes, as the parameters were chosen such that the algorithm is fast and less accurate. Hence, PCA starts from the start. If however the algorithm would be used for another application this setting might be relevant to change.

8.5 Problem with slope or banking of the road

As explained in section 1 the goal of this project was to compensate for the misalignment of the coordinate system of the vehicle and the coordinate system of the sensors. It is, as explained in section 4.4, desirable to have a situation where the resultant of the accelerometer or the gyroscope measurements points in a known direction in the vehicle frame. However, there is no such direction in the vehicle frame but instead the information is in the world frame. Hence, both the gravity method and the curve method actually finds the misalignment between the sensor frame and the world frame, for which the z-axis coincides with the z-axis of the vehicle frame only if the vehicle is standing or driving on plane surface. If the road traveled has a slope or a bank when ZMS or curve samples are collected, the algorithm will generate errors on α_y or α_x respectively. The error will correspond to the slope/bank angle of the road.

Since the median of several calculated angles is used, if as many uphill as downhill occasions and as many left as right bankings occur during the calibration, the final result will remain unaffected. However the algorithm is meant to be as usable in all environments, and for example if the road stretch to travel when a device has just been mounted in a vehicle is all uphill, then the result might be poor.

This problem can not be observed in any of the test results, possibly since none of the test environments features large changes in altitude, and further testing is needed to learn how problematic this issue is.

Also the angle α_z can bee affected by slopes of the road. If for example the vertical misalignment and the rotation matrix R_{xy} was calculated from when the vehicle was standing or driving on flat surface, but samples for PCA are collected when driving on a sloped road, parts of the undesired gravity acceleration can bee expected on the x-axis and/or y-axis measurements.

8.6 Small validation basis

The fact that the number of tests performed to analyze the result is low affects the reliability of the results. Since it is clear that, at this stage, the algorithm is highly dependent on where and how the vehicle has been driven further testing should be conducted in order to ensure that the system works as desired. A beta version of the BiFi algorithm (without the calibration algorithm) is at the time of writing implemented in a number of postal vehicles for validation. The same should preferably be done also with the updated program after it has been finally tuned to get a larger validation basis than can be achieved by the developers. A larger set of tests is also desirable in order to ensure that the fact that there is a misalignment also between the vehicle frame and the world frame does not generate an unacceptably large error.

8.7 Conclusions

The results from simulation indicates that the methods chosen is suitable for the purpose and that the algorithm itself works. However, as mentioned in section 8.6 a larger set of tests is required to accurately validate the performance of the algorithm and the problem of distinguishing between the coordinate system of the vehicle and the coordinate system of the world remains unresolved. As for the usage in the BiFi project the resulting algorithm would, with further validation and adjustments in the implementation, enable unspecified mounting of the hardware and hence improve the possibilities to extend the range of the project.

Bibliography

[1] Begin John David and Ka C. Cheok. Calibration of multi-axis accelerometer in vehicle navigation system. 2003. US6532419 B1.

[2] Berglund, Andreas. Tjäle - en litteraturstudie med särskilt fokus påtjällossning. LuleåTekniska Universitet, 2009.

[3] Bishop, Robert H. Mechatronic Systems, Sensors, and Actuators: Fundamentals and Modeling. Second edition. Taylor & Francis Group, LLC. 2007 (2008).

[4] Chelvi Kalai, T and Rangarajan, P, P. Principal Pattern Analysis: A Combined Approach for Dimensionality Reduction with Pattern Categorization. *International Journal of Computer Applications.* 41 no. 6 (2012): 35-41. Published by Foundation of Computer Science, New York, USA.

[5] Hinson, Anthony. Eigensolution of a 2x2 Matrix. 2011. http://www.ahinson.com/algorithms_general/Sections/Mathematics/Eigensolution2x2.pdf (2013-11-01).

[6] Intel. ATAN2.*Intel Fortran Compiler XE 13.1 User Reference Guides.* http://software.intel.com/sites/products/documentation/doclib/stdxe/2013/composerxe/compiler/for mac/GUID-60F7BC1C-DB2B-42C3-9D71-724127EABE28.htm (2013-12-10).

[7] Javier Almazan, Luis M. Bergasa, J. Javier Yebes, Rafael Barea and Roberto Arroyo. Full auto-calibration of a smartphone on board a vehicle using IMU and GPS embedded sensors. *Intelligent Vehicle Symposium (IV)*, June 2013. Gold Coast, QLD. doi: 10.1109/IVS.2013.6629658.

[8] Marco D. Tundo, Edward Lemaire and Natalie Baddour. Correcting Smartphone Orientation for Accelerometer-Based Analysis. *Medical Measurements and Applications Proceedings (MeMeA), IEEE International Symposium*, May 2013. Gatineau, QC. doi: 10.1109/MeMeA.2013.6549706.

[9] Otman A. Basir, Seyed Hamidreza Jamali, William Ben Miners and Jason Toonstra. Method of Correcting the Orientation of a Freely Installed Accelerometer in a Vehicle. 2013. US20130081442 A1.

[10] Prashanth Mohan, Venkata N. Padmanabhan and Ramachandran Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones. Microsoft Research India, 2008. http://research.microsoft.com/pubs/78568/nericell-sensys2008.pdf (2013-12-11).

[11] Bob and Richard Palais and Stephen Rodi. A Disorienting Look at Euler's Theorem on the Axis of a Rotation. *American Mathematical Monthly* 10 no.116 (2009): 892-209.

[12] Principal Component Analysis. Carnegie Mellon University. 2010. http://www.stat.cmu.edu/ cshalizi/490/pca/pca-handout.pdf (2013-09-08).

[13] Semcon, Klimator. Slutrapport BiFi del 2 - publik rapport. 2012. http://bifi.se/Documents/BiFi_Del2_Publik_Slutrapport.pdf (2014-01-11).

[14] Smith Lindsay. A tutorial on Principal Component Analysis. University of Otago, 2002. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf (2013-08-12).

[15] Stanšin Sara and Tomažič Sašo. Angle Estimation of Simultaneous Orthogonal Rotations from 3D Gyroscope Measurements. *Sensors* 11 no.9 (2011): 8536-8549. doi:10.3390/s110908536.

[16] Tamar Shinar. CSI30: Computer Graphics. Computer Science and engineering, UC Riverside. 2012. http://www.cs.ucr.edu/ shinar/courses/cs130-spring-2012/content/Lecture18.pdf (2013-10-28).

[17] Trafikverket. Tjälskador och tjällossning. 2012. http://portal.liikennevirasto.fi/sivu/www/s/underhall/tjallskador_tjallossning (2014-01-12).

[18] Weisstein, Eric W. Rotation Matrix. http://mathworld.wolfram.com/RotationMatrix.html (2013-10-22)

[19] Zang Hui, CHai Wei, Zeng Huan-Tao and Lou Qiang . Research on Calibration method for the installation error of three-axis acceleration sensor. *International Journal of Mechanical & Mechatronics Engineering.* 11 no. 4 (2011): 6-13. doi = 10.1016/S0006-3495(96)79538-3.