CHALMERS

Design and implementation of an attitude estimation system to control orthopedic components

Master's Thesis in Systems, Control and Mechatronics

TIMO VON MARCARD

Division of Signal Processing
Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2010
Master's Thesis EX032/2010

MASTER'S THESIS EX032/2010

Design and implementation of an attitude estimation system to control orthopedic components

Master's Thesis in Systems, Control and Mechatronics

TIMO VON MARCARD

Division of Signal Processing
Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2010

Design and implementation of an attitude estimation system to control orthopedic components
TIMO VON MARCARD

Master's Thesis EX032/2010
Division of Signal Processing
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

This document was typeset using LaTeX

Design and implementation of an attitude estimation system to control orthopedic components

Timo von Marcard

Division of Signal Processing
Department of Signals and Systems
Chalmers University of Technology

Abstract

This thesis presents the design and implementation of an attitude estimation system for controlling orthopedic and prosthetic devices. Recent developments have uncovered that human limb orientation relative to the ground is a convenient input signal and helps to improve functionality and comfort.
A sensor module provides self-contained measurements of angular rates, acceleration and magnetic field components. In order to yield accurate attitude estimates, an Extended Kalman Filter is designed to combine the measurements with a system model and to take advantage of the complementary sensor error characteristics. The system model is based on a quaternion attitude representation and derived from rigid body kinematics. Quaternions have the advantage of being computationally efficient and do not suffer from singularities.
The designed attitude estimation algorithm is verified with MATLAB in a hardware-in-the-loop environment and evaluated for floating-point and fixed-point number representations. In order to yield a portable, self-contained system with low power consumption, the algorithm is implemented on a 32-bit microcontroller. The computational limitations of microcontrollers have been considered throughout the filter design and tests have shown, that only the fixed-point version can be computed fast enough to provide accurate attitude estimates.
Static and dynamic accuracy of the system are determined for axes lying in the plane orthogonal to the gravity vector. The earth magnetic field occurred to be almost parallel to the gravity vector at the test location, which prevents the algorithm to provide reliable estimates of the angle about the vertical axis.

Keywords: AHRS, Inertial and magnetic sensors, Quaternions, EKF, TRIAD, Fixed-point implementation

Duderstadt, June 2010
Timo von Marcard

Acknowledgments

The present work was carried out at the electronic development department at Otto Bock Healthcare GmbH in Duderstadt, Germany. I would like to thank Otto Bock for offering such a fascinating and demanding master's thesis project to me.

I want to express my gratitude to Dipl.-Ing. (FH) Erik Albrecht-Laatsch and Dipl.-Ing. (FH) Michael Nolte for initiating and supervising the thesis project. I always appreciated the support during the complete time period of my work. I also want to thank all colleagues at the electronic development department for the warm welcome and the excellent working conditions.

My sincere thanks go to Professor Tomas McKelvey for supervising and examining the present work at Chalmers University of Technology. Thank you Bastian Nebenführ, for being my opponent during the thesis presentation and for providing several hints to improve and finalize this report.

Finally, I would like to express my appreciation to all people who contributed to my great and memorable time in Göteborg and at Chalmers. Obtaining the Master of Science degree in Sweden was an outstanding and inspiring experience.

Nomenclature

Abbreviations

A/D	Analogue/Digital
AHRS	Attitude Heading Reference System
DCM	Direction Cosine Matrix
DOF	Degree Of Freedom
DSP	Digital Signal Processor
EKF	Extended Kalman Filter
FPU	Floating Point Unit
GPS	Global Positioning System
HIL	Hardware-In-the-Loop
IDE	Integrated Development Environment
INS	Inertial Navigation System
MEMS	Micro-Electro-Mechanical System
NED	North East Down
PCB	Printed Circuit Board
QUEST	QUaternion ESTimaton algorithm
TRIAD	TRIAD algorithm (based on two triads)
UKF	Uncented Kalman Filter

Mathematical conventions

$\mathbf{x}$	column vector of dimension $n \times 1$
$\vec{x}$	column vector of dimension 3×1
$\hat{x}$	estimate of x
$\dot{x}$	time derivative of x
$\mathbf{x}^B$	vector $\mathbf{x}$ in body frame
$\mathbf{x}^N$	vector $\mathbf{x}$ in NED frame
$\mathbf{A}$	matrix of dimension $n \times m$
$\mathbf{A}^{-1}$	matrix inverse
$\mathbf{A}^T$	matrix transpose
$\mathbf{C}_B^N$	rotation matrix from body to global frame
$\mathbf{R}_x$	rotation matrix about axis x
$E(\cdot)$	expectational operator
$\hat{\mathbf{x}}^+$	a posteriori estimate of $\mathbf{x}$
$\hat{\mathbf{x}}^-$	a priori estimate of $\mathbf{x}$
$\mathbf{q}^*$	quaternion conjugate
$N(\mathbf{q})$	norm of $\mathbf{q}$

Symbols

$\mathbf{x}$	state vector
$\hat{\mathbf{x}}$	estimate of state vector
$\mathbf{u}$	vector of input signals
$\mathbf{y}$	vector of output signals
$\hat{\mathbf{y}}$	estimate of output signals
$\hat{\mathbf{e}}$	vector of estimation errors
$\mathbf{A}$	system matrix
$\mathbf{B}$	input matrix
$\mathbf{C}$	output matrix
$\mathbf{s}$	system noise vector
$\mathbf{v}$	measurement noise vector
$\mathbf{Q}$	system noise covariance matrix
$\mathbf{R}$	measurement noise covariance matrix
$\mathbf{P}$	state covariance matrix
$\mathbf{K}$	Kalman gain matrix
$\mathbf{f}$	(nonlinear) function of system dynamics
$\mathbf{g}$	(nonlinear) function of output equations
$\mathbf{F}$	Jacobian matrix of $\mathbf{f}$ w.r.t $\mathbf{x}$
$\mathbf{L}$	Jacobian matrix of $\mathbf{f}$ w.r.t $\mathbf{s}$
$\mathbf{H}$	Jacobian matrix of $\mathbf{g}$ w.r.t $\mathbf{x}$
$\mathbf{M}$	Jacobian matrix of $\mathbf{g}$ w.r.t $\mathbf{v}$
$\mathbf{q}$	quaternion
w, x, y, z	quaternion components of $\mathbf{q}$
ψ, θ, ϕ	Euler angles Yaw, Pitch, Roll
p, q, r	angular rates about body axes (x,y,z)
$\omega_x, \omega_y, \omega_z$	angular rates about NED-axes (x,y,z)
$\vec{g}$	gravity vector
$\vec{a}$	acceleration vector (measurement)
$\vec{B}$	earth magnetic field vector
$\vec{m}$	magnetic field vector (measurement)

List of Figures

List of Tables

Table of Contents

1. Introduction

1.1. Motivation

Standing, walking, sitting, grabbing, lifting, just to name a few, are all activities which are natural and normal to every healthy person. Accidents, diseases or congenital malformation may strongly impede these activities, if extremities are amputated or missing. Considering the physical consequences, a comparable reduction of life quality applies to persons with temporal or permanent limb dysfunctionalities. In order to enable affected persons to live their life independently and possibly without any restrictions, great effort has been put into development of prostheses and orthoses. Orthoses aim to support restricted limb functionalities, whereas prostheses completely replace missing extremities. Historically these products were pure mechanical constructions.
The revolutionary progress of measurement techniques and computer technology over the last decades has made it possible to implement sensors and actuators, all processed and controlled through microcontrollers. These complex, mechatronic innovations brought major improvements in functionality, flexibility, reliability and comfort to orthopedic products. Commonly utilized and required physical quantities include measurements of various joint angles, angular velocities and moments.
Recent developments showed the need to sense limb orientations relative to the ground to serve as an additional input signal. This Master Thesis Project "Design and implementation of an attitude estimation system to control orthopedic components" in cooperation with Otto Bock Healthcare GmbH deals with investigating and developing such a sensor system.

1.2. Problem statement

In order to enhance existing and upcoming orthopedic products, reliable and complete information about human limb orientations relative to the ground are required. Therefore a stand-alone attitude estimation module, equipped with inertial and magnetic sensors shall be developed. Desired limb attitude information will be accessible by rigidly attaching the module to the part of the body, which should be monitored.
The attitude estimation system has to be capable of a broad range of dynamical movements and must not depend on any external signal sources like GPS or optical trackers. This setup theoretically allows operation in any surrounding and can be implemented on a single, small and light weight board meeting the requirements for use in orthopedic products.

Commercial products, which may be used for the designated application are already available on the market. The unit price ranges at four figures, whereas costs for the individual hardware components are considerably smaller. For this reason it is desired to develop an own, separate solution.
A Printed Circuit Board (PCB) equipped with three orthogonal gyrometers, three orthogonal accelerometers and three orthogonal magnetometers serves as a basis for preliminary studies. National Instruments (NI) measurement cards in combination with Mathwork's MATLAB/Simulink are provided for measuring, modeling and verification of sensors and attitude estimation algorithms. Subsequently, the derived algorithm has to be implemented on a microcontroller and final results concerning estimation accuracy have to be determined and discussed.

1.3. Thesis organization

This thesis describes the derivation and verification of an algorithm to estimate attitude with inertial and magnetic field sensors suitable for real-time operation in a microcontroller. A state observer setup is created, combining a dynamic process model with sensor measurements by an Extended Kalman Filter. The thesis is organized as follows.
Chapter 2 reviews the theory and tools, which build the theoretical foundation of the thesis. The principle of state estimation, including state observers and Kalman Filters are presented. Required definitions for describing orientation in space are given and different attitude representations are reviewed. Inertial and magnetic field sensors principles are briefly introduced along with common and individual sensor errors. Besides, a short description of designated sensors and the utilized sensor module is included. A small section is dedicated to number representations in a digital computer or microcontroller.
Chapter 3 presents the mathematical modeling of the attitude estimation algorithm. Absolute attitude determination and attitude estimation approaches are reviewed. Modeling human motions is briefly discussed. A process model is derived and a Kalman Filter is designed, providing the desired estimation algorithm.
Chapter 4 is dedicated to implementation and verification of the algorithm designed in Chapter 3. The principle functionality of the algorithm is verified with MATLAB and tested for different configurations. Implementation of the attitude estimator in a microcontroller is described and evaluated. Additionally accuracy of the final attitude estimation system is determined.
Chapter 5 concludes the thesis by discussing the designed system and analyzing achieved results. Furthermore, suggestions for future work are given.

2. Theory

2.1. State estimation

In order to control or supervise a system in a desired way it is necessary to have knowledge about the actual internal states. State estimation subsumes the theory and tools to accurately determine these internal states from measurable input and output signals. This comes along with four major challenges: input and output dependencies have to be known, sensor measurements are corrupted by noise, disturbed by other signals and often the desired state can not be measured directly. In the latter case the state has to be computed through other, measurable signals.
A pure deterministic approach is to compute internal states directly from input and output sensor measurements. This requires an accurate system model, relating input and output signals to internal states and is mainly limited by the model accuracy and noise properties of sensors measuring input and output signals. A further requirement that applies to all state estimation approaches, is the states of interest have to be computable from system input and output variables, usually denoted as state observability.
In order to improve estimation accuracy, stochastic properties of the system model, disturbances and the sensors can be considered. This is very common, since measurements are always subject to noise processes, disturbances are rarely predictable and a system model can only approximate a real system.
The following sections introduces the principle of state observers and Kalman filtering. Presented information, properties and equations are taken from [Sim06] and [AW96].

2.1.1. State observer

In order to estimate internal states of a dynamical system a common concept is to implement a state observer. A state observer is a mathematical model of the real system, which provides internal state estimates based on measurable input and output signals. This approach requires that the real system can be modeled by a discrete time state space representation

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{2.1}$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k), \tag{2.2}$$

where $\mathbf{x}_k$ is the internal state vector and $\mathbf{u}_k$ are input signals at time k. The function $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ describes how the states at time $k+1$ depends on the previous state $\mathbf{x}_k$ and inputs $\mathbf{u}_k$. The system outputs $\mathbf{y}_k$ are related to the internal states by a function $\mathbf{g}(\mathbf{x}_k)$.

Equations (2.1) and (2.2) are assumed to model the real, physical system, where $\mathbf{x}_k$ is supposed to be the "true" state vector. The state observer is based on the same equations, but holding an estimate of $\mathbf{x}_k$ denoted as $\hat{\mathbf{x}}_k$. Based on the actual state estimate, the output signals can be predicted through Equation (2.2), denoted as $\hat{\mathbf{y}}_k$, respectively. Thus, the observer equations are

$$\hat{\mathbf{x}}_{k+1} = \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k) \tag{2.3}$$

$$\hat{\mathbf{y}}_k = \mathbf{g}(\hat{\mathbf{x}}_k). \tag{2.4}$$

An illustrative graphic describing the general process of state estimation with a state observer

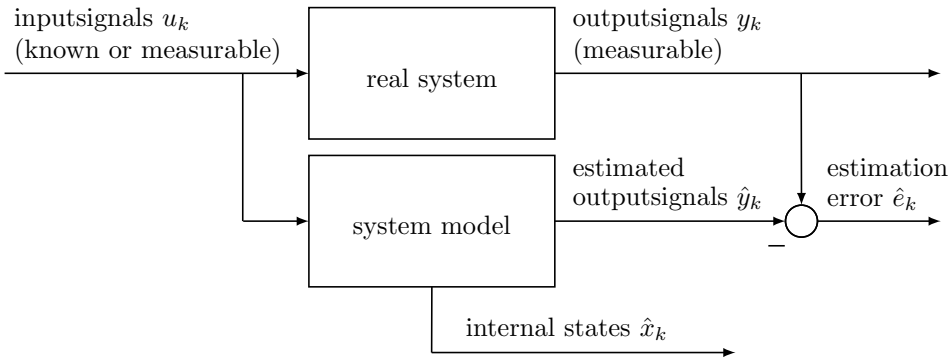

Fig. 2.1.: State observer in open-loop configuration

is depicted in Figure 2.1. The state observer operates in parallel to the real system and both yield the same input signal u_k. Assuming the mathematical model accurately describes the real system and the estimate $\hat{\mathbf{x}}_k$ is correct, both blocks produce the same output and the estimation error $\hat{\mathbf{e}}_k$, formed by the difference of $\mathbf{y}_k$ and $\hat{\mathbf{y}}_k$, equals zero. If $\hat{\mathbf{x}}_k$ is not correct, the real system output differs from the output estimate and $\hat{\mathbf{e}}_k$ becomes non-zero. Thus, the estimation error gives a benchmark how accurate the actual state estimates are. In order to improve the state estimate, $\hat{\mathbf{e}}_k$ could be used for correction, which is depicted in Figure 2.2. This state observer setup with feedback, commonly referred to as Luenberger observer, corrects the state $\hat{\mathbf{x}}_k$ by a function $\mathbf{k}(\hat{\mathbf{e}}_k)$, which has to be chosen in such a way that the estimation error will be as small as possible. A common approach that minimizes the variance of the estimation error is the Kalman Filter, which is presented in the next section.

2.1.2. Standard Kalman Filter

The standard Kalman Filter is a widely used filtering and estimation technique, which combines a linear system model with statistical methods to accurately estimate the system's state variables. It was named after Rudolf E. Kalman, who played a key role during its development around 1960.

The Kalman Filter is a recursive filter, which only stores the estimated state of the previous time step and current measurements to compute the actual state estimate. This property makes

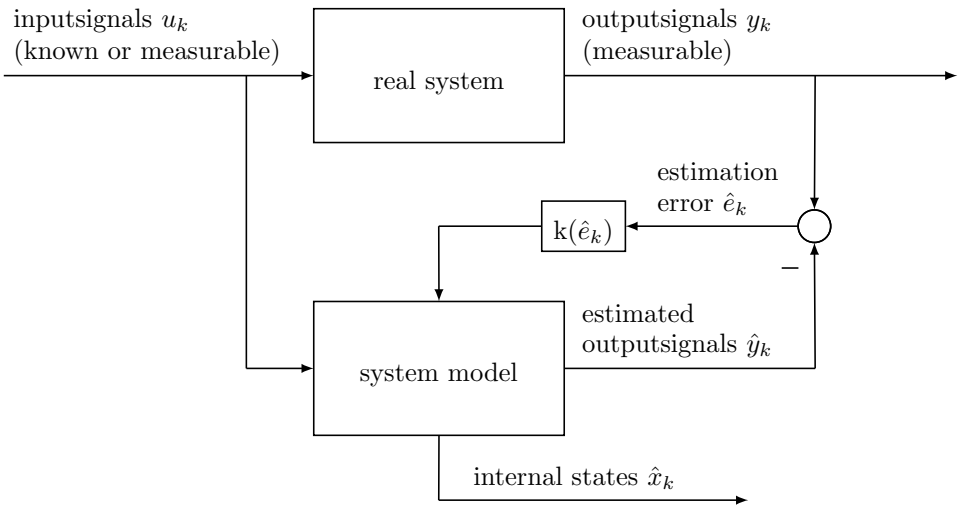

Fig. 2.2.: State observer with feedback

it very useful for real-time applications. Besides, it is the optimal state estimator, regarding minimizing the error variance between true and estimated system states, if the modeled noise processes are Gaussian. For alternative white noise processes it is still the optimal linear state estimator.
A standard Kalman filter requires a linear system model in state space form according to

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1} + \mathbf{s}_{k-1} \tag{2.5}$$

$$\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{v}_k \tag{2.6}$$

$$E(\mathbf{s}_k\mathbf{s}_j^T) = \mathbf{Q}_k\delta_{k-j} \tag{2.7}$$

$$E(\mathbf{v}_k\mathbf{v}_j^T) = \mathbf{R}_k\delta_{k-j} \tag{2.8}$$

$$E(\mathbf{s}_k\mathbf{v}_j^T) = \mathbf{0}, \tag{2.9}$$

where

- $\mathbf{x}_k$: state vector with dimension n×1
- $\mathbf{A}_{k-1}$: system matrix with dimension n×n
- $\mathbf{B}_{k-1}$: input matrix with dimension l×n
- $\mathbf{u}_{k-1}$: input vector with dimension l×1
- $\mathbf{s}_{k-1}$: system noise vector with dimension n×1
- $\mathbf{y}_k$: output vector with dimension m×1
- $\mathbf{C}_k$: output matrix with dimension m×n
- $\mathbf{v}_k$: measurement noise vector with dimension m×1

- $E(\cdot)$: expectational operator
- δ_{k-j}: Kronecker delta function

$\{\mathbf{s}_{k-1}\}$ and $\{\mathbf{v}_k\}$ are noise processes modeling the uncertainties of the system equation and measurements. They are required to be white, zero-mean, uncorrelated with known covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$. Since the filter is usually implemented on a computer or microcontroller, only discrete time system models and filter equations are considered.
The Kalman Filter is a recursive filter, periodically predicting the state variables based on the system equations and subsequently correcting them by considering the sensor measurements. These steps are commonly denoted as time-update and measurement-update respectively. The state correction is based on the state vectors covariance matrix denoted as $\mathbf{P}_k$.
The filter is initialized by

$$\hat{\mathbf{x}}_0^+ = E(\mathbf{x}_0) \tag{2.10}$$
$$\mathbf{P}_0^+ = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)^T] \ . \tag{2.11}$$

Subsequently, the Kalman filter equations are computed periodically at each sampling interval and are given as follows.

1. Perform a time-update, which includes computation of a priori state and covariance estimates based on the system equations:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{B}_{k-1}\mathbf{u}_{k-1} \tag{2.12}$$
$$\mathbf{P}_k^- = \mathbf{A}_{k-1}\mathbf{P}_{k-1}^+\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1} \ . \tag{2.13}$$

2. Perform a measurement-update, which corrects the state estimates based on measurements yielding a posteriori state and covariance estimates:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{C}_k^T(\mathbf{C}_k\mathbf{P}_k^-\mathbf{C}_k^T + \mathbf{R}_k)^{-1} \tag{2.14}$$
$$\hat{\mathbf{x}}_k^+ = \hat{x}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}_k\hat{\mathbf{x}}_k^-) \tag{2.15}$$
$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \ . \tag{2.16}$$

$\mathbf{K}_k$ is the Kalman gain, which amplifies the estimation error and corrects the states in an optimal way. Several other ways of writing the Kalman filter equations exist, which are mathematically equivalent. The Joseph stabilized version of the covariance expression in Equation (2.16) is chosen, because it guarantees that $\mathbf{P}_k^+$ is symmetric positive definite, as long as $\mathbf{P}_k^-$ has this property.

2.1.3. Extended Kalman Filter

In order to extend Kalman Filters to nonlinear system models, the Extended Kalman Filter (EKF) was developed. The EKF is based on a linearized system model, evaluated at the current

state estimate. Assuming a non-linear system given by

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{s}_{k-1}) \tag{2.17}$$

$$\mathbf{y}_k = \mathbf{g}_k(\mathbf{x}_k, \mathbf{v}_k) \tag{2.18}$$

$$\mathbf{s}_k \sim (\mathbf{0}, \mathbf{Q}_k) \tag{2.19}$$

$$\mathbf{v}_k \sim (\mathbf{0}, \mathbf{R}_k) \tag{2.20}$$

and initializing the filter identically to Equations (2.10) and (2.11), the EKF equations are given as follows.

1. Time-update:

$$\hat{\mathbf{x}}_k^- = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1}, \mathbf{0}) \tag{2.21}$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^T + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^T \ , \tag{2.22}$$

   where $\mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1}^+}$ and $\mathbf{L}_{k-1} = \left.\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{w}}\right|_{\hat{\mathbf{x}}_{k-1}^+}$ are Jacobian matrices, evaluated at the current state estimate.

2. Measurement-update:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^T)^{-1} \tag{2.23}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k[\mathbf{y}_k - \mathbf{h}_k(\hat{\mathbf{x}}_k^-, 0)] \tag{2.24}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^T\mathbf{K}_k^T \ , \tag{2.25}$$

   where $\mathbf{H}_{k-1} = \left.\frac{\partial \mathbf{h}_{k-1}}{\partial \mathbf{x}}\right|_{\hat{x}_k^-}$ and $\mathbf{M}_{k-1} = \left.\frac{\partial \mathbf{h}_{k-1}}{\partial \mathbf{v}}\right|_{\hat{x}_k^-}$ are Jacobian matrices, evaluated at the current state estimate.

The advantages of the EKF are the commonly simple derivations of Jacobian matrices and a deterministic computational effort. A disadvantage comes along with the linearization, which is only a first order approximation of the non-linear equations. This introduces linearization errors, which can be reduced by using higher order approaches at the cost of higher complexity and computational expense.

2.1.4. Alternative state estimators

Several other approaches exist for state estimation of linear and non-linear systems. The H_∞-filter was specifically designed for robustness capable of handling large modeling errors and noise uncertainties. It is a filter, which minimizes the worst-case estimation error. For non-linear systems, the Uncented Kalman Filter (UKF) is an alternative to the extended Kalman filter. The UKF avoids linearization of the system equations and approximates the non-linear transformation of the variable's distribution instead. This is done by computing the system equations for a set of so called sigma points and estimate the new mean and covariance from the

results. The UKF has the advantage of avoiding linearization but requires more computations, due to the number of sigma points. More examples for alternative state estimators are methods of information filtering, Fuzzy methods, particle filters and neuronal networks, which are not further described [vR06].

2.2. Attitude representation

This section determines necessary conventions of coordinate systems required for the present work. Additionally, three representations for describing orientation in space are presented: Direction Cosine Matrix (DCM), Euler angles and quaternions. Accompanying definitions, properties and formula for these representations are derived from [Ber07], [VvVB04] and [Vic01].

2.2.1. Coordinate Systems and Definitions

The spatial orientation generally describes how a coordinate system is aligned with respect to another (reference) coordinate system. In order to describe the orientation of an object it is necessary to define a body-fixed coordinate system. It is fixed to the objects geometry and moves with the object as it changes position or rotates. For the present work the body-fixed coordinate system is defined for the sensor module as depicted in Figure 2.3. The x-axis points in the direction of the long side of the sensor module, y-axis in the direction of the short side to the right and z-axis completes the right hand axis set.

The reference coordinate system is usually defined as a non-moving, non-rotating coordinate

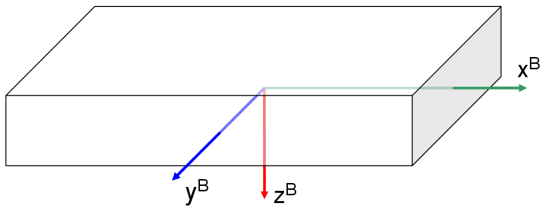

Fig. 2.3.: Definition of body frame within the sensor module

system. A common choice is a tangential, earth-fixed coordinate system with NED-convention shown in Figure 2.4 [Far08]. NED stands for North-East-Down, which means that the global x-axis points to north direction, y-axis to east direction and z-axis points down to the center of earth. Vectors and axes are labeled with a superscript B in the body-fixed frame and with a superscript N in reference, global or NED frame, respectively.

According to Euler's rotation theorem, the orientation of a coordinate system to another can be described by a rotation about a single axis, if one common point is fixed (e.g. the coordinate origin) [Wei03]. In three-dimensional euclidean space this requires at least three parameters. Often more parameters are used, which comes along with different advantages in certain applications. Three orientation representations with a different number of parameters are described in the following subsections, namely direction cosine matrix, Euler angles and quaternions.

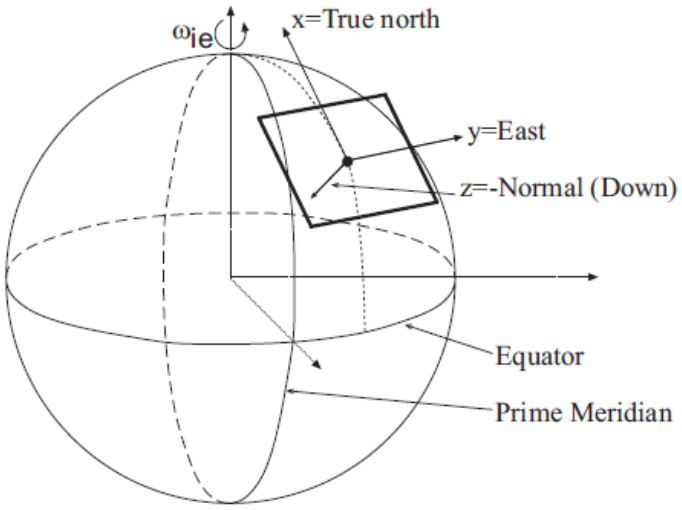

Fig. 2.4.: Definition of tangential NED frame [Far08]

2.2.2. Direction cosine matrix

The Direction Cosine Matrix (DCM) represents attitude or orientation of two coordinate systems by relating their accompanying basis vectors. Generally, basis vectors can be interpreted as vectors of unit length pointing in the direction of x, y and z-axis of a Cartesian coordinate system. A vector $\vec{a}$ within such a coordinate system can be written as:

$$\vec{a} = a_1\vec{e}_1 + a_2\vec{e}_2 + a_3\vec{e}_3 = a_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + a_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} , \tag{2.26}$$

where $\vec{e}_n$ are the basis vectors and a_n the basis components respectively. The vector is also said to be expressed in e, where e stands for the summation of the basis vectors $\vec{e}_n$. In order to express the same vector in a different coordinate system the following equation has to be fulfilled

$$\vec{a} = a_1\vec{e}_1 + a_2\vec{e}_2 + a_3\vec{e}_3 = b_1\vec{n}_1 + b_2\vec{n}_2 + b_3\vec{n}_3 , \tag{2.27}$$

where $\vec{n}_k$ are the basis vectors of the new coordinate system and b_k the basis components respectively. The latter relationship can be expressed in matrix form according to

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} , \tag{2.28}$$

where each matrix element c_{nk} is the cosine of the angle between e_n and n_k, also referred as direction cosine. Therefore, the matrix formed by the c_{nk} elements is commonly named "Direction Cosine Matrix". A DCM transforms a vector from e to n, thus it is also considered a

"rotation matrix from e to n". Even though the derivation is straight forward, it is not easy to imagine attitude from a DCM, thus it is not very intuitive.

Advantages of a direction cosine matrix for attitude representation:

- No singularities

Disadvantages of a direction cosine matrix for attitude representation:

- Not intuitive
- 9 parameters for attitude representation (not independent)

2.2.3. Euler angles

Euler angles describe the orientation of two coordinate systems by three successive rotations around separate coordinate axes. As the name suggests, this form of orientation representation has been developed by Leonhard Euler. There exist 12 different possibilities to rearrange the order of rotations, if only "right-handed" axes are considered. Each order defines a different Euler angle convention.
Because of its intuitive clarity the ZYX-convention is presented and used throughout the present work. For this convention Euler angles are defined by the following sequence of rotations to rotate a vector from NED frame to body frame:

1. positive rotation about the z-axis by an angle ψ
2. positive rotation about the new y-axis by an angle θ
3. positive rotation about the new x-axis by an angle ϕ

The angles ψ, θ and ϕ are also known as Yaw, Pitch and Roll respectively. These names are reserved for this convention and have the following ranges

$$\psi = \pm\pi \qquad \theta = \pm\frac{\pi}{2} \qquad \phi = \pm\pi \ . \tag{2.29}$$

A rotation around the x-axis about an angle ϕ can be written as a rotation matrix according to

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} . \tag{2.30}$$

Similarly for θ and ψ:

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.31}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{2.32}$$

The sequence of rotations described above can be summarized to a single rotation matrix

$$\mathbf{C}_N^B = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ -\cos(\phi)\sin(\psi) + \sin(\phi)\sin(\theta)\cos(\psi) & \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\cos(\theta) \\ \sin(\phi)\sin(\psi) + \cos(\phi)\sin(\theta)\cos(\psi) & -\sin(\phi)\cos(\psi) + \cos(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{2.33}$$

In order to rotate a vector from NED frame to body frame the following computation has to be performed:

$$\vec{x}^B = \mathbf{C}_N^B\vec{x}^N . \tag{2.34}$$

It should be mentioned that matrix multiplication is not commutative, so the order of multiplication has to be considered. Since C_N^B is a rotation matrix its inverse is equal to its transpose. Thus, a rotation from body coordinates to NED coordinates can be described by

$$\vec{x}^N = (\mathbf{C}_N^B)^{-1}\vec{x}^B = (\mathbf{C}_N^B)^T\vec{x}^B = \mathbf{R}_z^T\mathbf{R}_y^T\mathbf{R}_x^T\vec{x}^B = \mathbf{C}_B^N\vec{x}^B . \tag{2.35}$$

Advantages of Euler angles for attitude representation:

- Very intuitive
- Minimum number of 3 parameters used for attitude representation

Disadvantages of Euler angles for attitude representation:

- Singularity at Pitch angle $\theta = \pm\frac{\pi}{2}$; for this case Roll and Yaw axes are parallel and attitude has no unique Euler angle representation, since only the sum of ϕ and ψ is relevant.

2.2.4. Quaternions

An alternative method to describe orientation are quaternions. A quaternion is an extension of the complex numbers, consisting of a real scalar and a complex vector. The vector part contains three complex elements denoted i,j and k. They can be interpreted as basis vectors of a three-dimensional Cartesian coordinate system. Several quaternion definitions exist:

$$\mathbf{q} = w + xi + yj + zk = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = (w, \vec{v}), \tag{2.36}$$

where the imaginary parts have the following properties:

$$ii = i^2 = -1 \tag{2.37}$$
$$jj = j^2 = -1 \tag{2.38}$$
$$kk = k^2 = -1 \tag{2.39}$$

and

$$ij = k = -ji \tag{2.40}$$
$$jk = i = -kj \tag{2.41}$$
$$ki = j = -ik. \tag{2.42}$$

Intuitively, a quaternion represents orientation of two coordinate frames by defining a rotation axis with the complex vector part and the scalar part denotes the angle to rotate around this axis. According to Euler's theorem this is sufficient to describe any orientation in three-dimensional space.
Quaternion operations are almost identical to common vector operations. This is valid for component wise operations like equality, addition and multiplication with a scalar. A special case is quaternion multiplication, which is defined for two quaternions

$$\mathbf{q}_1 = w_1 + x_1 i + y_1 j + z_1 k \qquad \text{and} \qquad \mathbf{q}_2 = w_2 + x_2 i + y_2 j + z_2 k$$

as

$$\begin{aligned} \mathbf{q}_1 * \mathbf{q}_2 = {} & (w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) + (x_1 w_2 + w_1 x_2 - z_1 y_2 + y_1 z_2) i + \\ & + (y_1 w_2 + z_1 x_2 + w_1 y_2 - x_1 z_2) j + (z_1 w_2 - y_1 x_2 + x_1 y_2 + w_1 z_2) k \ . \end{aligned} \tag{2.43}$$

The conjugate of a quaternion $\mathbf{q} = w + xi + yj + zk = (w, \vec{v})$ is defined as

$$\mathbf{q}^* = (w - xi - yj - zk) = (w, -\vec{v}) \tag{2.44}$$

and the norm or magnitude N($\mathbf{q}$) is

$$N(\mathbf{q}) = \sqrt{\mathbf{q} * \mathbf{q}^*} = \sqrt{w^2 + x^2 + y^2 + z^2} \ . \tag{2.45}$$

A quaternion is normalized by

$$\mathbf{q}_{normalized} = \frac{\mathbf{q}}{N(\mathbf{q})} \ . \tag{2.46}$$

A normalized quaternion is also denoted as a unit quaternion, since it has norm of unity. Quaternion inverse is defined as

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{N^2(\mathbf{q})} = \frac{\mathbf{q}^*}{\mathbf{q}^*\mathbf{q}} \ , \tag{2.47}$$

which simplifies for a unit quaternion to

$$\mathbf{q}^{-1} = \mathbf{q}^* \ . \tag{2.48}$$

In order to perform transformations between coordinate frames with quaternions, they are required to be of unit length. A rotation of an angle θ about an axis defined by unit vector $\vec{v}$ can be represented as a quaternion of the form

$$\mathbf{q} = (\cos(\frac{\theta}{2}), \vec{v}\sin(\frac{\theta}{2})) \ . \tag{2.49}$$

Thus a rotation about negative θ is equal to the quaternion conjugate of q. Representing a vector $\vec{p} = [p_x \ p_y \ p_z]^T$ in different coordinate systems can be computed by rotating $\vec{p}$ about a quaternion $\mathbf{q} = [w \ x \ y \ z]$, where $\mathbf{q}$ describes the orientation of the coordinate systems. Therefore $\vec{p}$ has to be written in quaternion form according to

$$\mathbf{p}_q = \begin{bmatrix} 0 & p_x & p_y & p_z \end{bmatrix} \ .$$

Rotation of $\mathbf{p}_q$ by a quaternion $\mathbf{q}$ is defined as:

$$\mathbf{p}_{rotated} = \mathbf{q} * \mathbf{p}_q * \mathbf{q}^{-1} \ . \tag{2.50}$$

Multiplying $\mathbf{p}_q$ with $\mathbf{q}$ according to quaternion multiplication rules, reordering the result and considering just the vector part, a rotation can be written in matrix form:

$$\begin{aligned} &\vec{p}_{rotated} = \mathrm{vec}(\mathbf{q} * \mathbf{p}_q * \mathbf{q}^{-1}) = \\ &= \frac{1}{N(\mathbf{q})} \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & -2wz + 2xy & 2wy + 2xz \\ 2wz + 2xy & w^2 - x^2 + y^2 - z^2 & -2xw + 2yz \\ -2wy + 2xz & 2wx + 2yz & w^2 - x^2 - y^2 + z^2 \end{bmatrix} \vec{p} = \mathbf{M}(q)\vec{p} \ . \end{aligned} \tag{2.51}$$

Advantages of quaternions for attitude representation:

- No singularities
- Low number of 4 parameters used for attitude representation

Disadvantages of quaternions for attitude representation:

- Not intuitive

2.3. Sensors

This section presents a description of sensors and the sensor module used in the present work. Three different sensor types are used, a 3-axis gyroscope, a 3-axis accelerometer and a 3-axis magnetometer. They are chosen because of their autonomous operation, not requiring any additional artificial source. The sensors are built in Micro-Electrical-Mechanical Systems (MEMS) technology, which make it possible to produce them in a small package size, low power consumption, cheap and with high availability. Further, MEMS technology develops very fast and new releases are still accompanied with major improvements in precision and longtime stability.

2.3.1. Error modeling

In general, sensors measure a certain physical quantity and output a value that depends on its strength and direction. The output value is commonly an electric voltage, that has to be processed in order to yield a value that is in the same domain as the quantity of interest. Preprocessing usually include bias removal and scaling. Bias is any nonzero sensor output, if the input is zero. Scaling factors are used to scale the output to the correct amplification and correct for misalignments of nominal and real sensor axes.
A simple error model, including bias, scaling and misalignment parameters for a 3-axis sensor type is derived in [GWA01] according to

$$\vec{z}_{out} = s_{nominal} \left(\mathbf{I}_{3\times3} + \mathbf{M}_{sa} \right) \vec{z}_{input} + \vec{b}_z \ , \tag{2.52}$$

where $\vec{z}_{out}$ and $\vec{z}_{input}$ are the three sensor output and input values, respectively. The input vector is scaled by a nominal scale factor $s_{nominal}$ and $\mathbf{M}_{sa}$ contains individual scale factor deviations and input axis misalignments. The additive bias values are denoted as $\vec{b}_z$. The individual elements m_{ij} of $\mathbf{M}_{sa}$ are illustrated in Figure 2.5, where the larger arrows represent the nominal input axes and the smaller arrows represent scaling deviations ($i = j$) and misalignments ($i \neq j$). The measurable sensor outputs result from adding scaled input values and sensor biases. Equation (2.52) relates sensor output and input in error-generating form, which shows how the input value is corrupted by the errors. In an application, it is desired to calculate the input values from sensor output values. This can be done by solving Equation (2.52) for the sensor

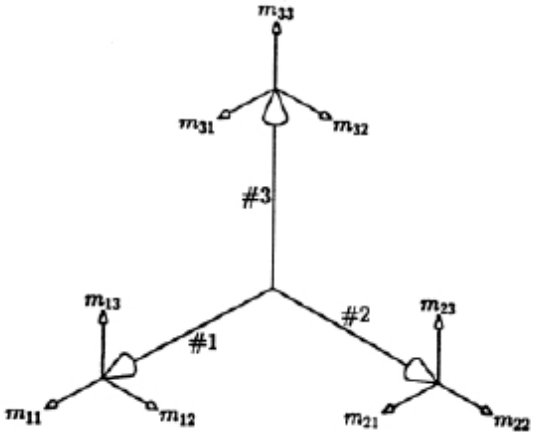

Fig. 2.5.: Elements of scaling and misalignment matrix [GWA01]

input values:

$$\begin{aligned}\vec{z}_{input} &= \frac{1}{s_{nominal}} \left(\mathbf{I}_{3\times3} + \mathbf{M}_{sa}\right)^{-1} \left(\vec{z}_{out} - \vec{b}_z\right) \\ &= \frac{1}{s_{nominal}} \left(\mathbf{I}_{3\times3} - \mathbf{M}_{sa} + \mathbf{M}_{sa}^2 - \mathbf{M}_{sa}^3 + ...\right) \left(\vec{z}_{out} - \vec{b}_z\right) \\ &\approx \frac{1}{s_{nominal}} \left(\mathbf{I}_{3\times3} - \mathbf{M}_{sa}\right) \left(\vec{z}_{out} - \vec{b}_z\right) \ . \end{aligned} \tag{2.53}$$

Defining

$$\mathbf{M} = \frac{1}{s_{nominal}} \left(\mathbf{I}_{3\times3} + \mathbf{M}_{sa}\right)^{-1} \tag{2.54}$$

gives the following short form of notation of the most basic sensor errors:

$$\vec{z}_{input} = \mathbf{M} \left(\vec{z}_{out} - \vec{b}_z\right) \ . \tag{2.55}$$

Matrix $\mathbf{M}$ and bias vector $\vec{b}_z$ have to be determined through a sensor calibration.

2.3.2. Accelerometers

A linear accelerometer is a device measuring acceleration in a single or in multiple spatial axes. The common principle of various existing sensor types is to sense the force on a proof mass and determine acceleration by Newton's second law of motion $F = ma$.
Accelerometers can be roughly classified into mechanical and solid state devices [Woo07]. Mechanical accelerometers measure displacement of a mass suspended by springs. Solid state devices consist of a cantilever beam, attached to a free moving proof mass at one end and rigidly attached to the case on the other end. Acceleration displaces the proof mass and the resulting force can be determined by measuring the beam's strain. MEMS accelerometers are based on the same principles as mechanical and solid state devices.

For the present work the MEMS inertial sensor LIS344ALH by ST is used, which is a very compact 3-axis accelerometer with a selectable full-scale of ±2g or ±6g. The LIS344ALH contains a proof mass in form of suspended silicon structures. Displacement of the proof mass is measured with capacitive half-bridges, producing a voltage proportional to the applied acceleration. Table 2.1 presents a selection of mechanical and electrical specifications taken from the accompanying datasheet for a supply voltage Vdd=3.3V. In order to prepare the analogue sensor output voltages for sampling and A/D-Conversion a lowpass-filter is applied to each output. According to the datasheets a first order lowpass-filter with 145 Hz cut off frequency was chosen.

Parameter	Test condition	Min.	Typ.	Max.	Unit
Sensitivity	Full-scale ±2g	0.66 -5%	0.66	0.66 +5%	V/g
	Full-scale ±6g	0.22 -10%	0.22	0.22 +10%	V/g
Zero-g-level	Full-scale ±2g	1.65 -5%	1.65	1.65 +5%	V
Bandwidth				1.8	kHz
Acc. noise density			50		$\mu g/\sqrt{Hz}$
Supply voltage		2.4	3.3	3.6	V
Supply current	normal mode		680	850	μA
Dimensions	LGA-16L Package		4x4x1.5		mm

Table 2.1.: Selection of mechanical and electrical specifications of accelerometer LIS344ALH by ST [ST08]

2.3.3. Gyroscopes

Angular rate sensors, commonly denoted as gyroscopes, exist in various types. They are roughly classified into mechanical, optical and MEMS gyroscopes [Woo07]. Optical devices are based on the principle of Sagnac effect, whereas mechanical and MEMS derivates operate on the principle of conservation of angular momentum. For the system of interest only MEMS gyroscopes are considered. The major disadvantage of MEMS gyroscope compared to mechanical and optical counterparts is their limited accuracy. According to [GWA01] this is due to various error sources including:

- bias variability from turn-on to turn-on, caused by changing gyroscope and electronics properties
- bias drift, due to thermo-mechanical white noise, which is usually modeled as a random-walk process
- temperature effects
- calibration errors

However, accuracy of MEMS gyroscopes increased noticeably over the last years, so further improvements are expected.
For the present work a two-axis (LPR550AL) and a single-axis (LY550ALH) MEMS gyroscope

by ST are used to measure angular rates in all spatial dimensions. A single-cased three-axis sensor was not available at the moment of time. The LPR550AL senses angular rates in Roll and Pitch, whereas the LY550ALH measures angular rate for Yaw. Both sensors are based on Coriolis principle and produce an analogue voltage related to the applied angular rate. Table 2.2 presents a selection of mechanical and electrical specifications taken from accompanying datasheets for a supply voltage of 3V. In order to prepare the analogue output voltages for sampling and A/D-Conversion a lowpass-filter is applied to each output. According to the datasheets a first order lowpass-filter with 100 Hz cut off frequency is applied.

Parameter	Test condition	Min.	Typ.	Max.	Unit
Measurement range	4x OUT (amplified) OUT (not amplified)		± 500 ± 2000		°/s °/s
Sensitivity	4x OUT (amplified) OUT (not amplified)		2 0.5		mV/°/s mV/°/s
Zero rate level			1.23		V
Bandwidth			140		Hz
Rate noise density			0.059		°/s/ $\sqrt{Hz}$
Supply voltage		2.7	3	3.6	V
Supply current	LY550ALH LPR550AL		4.8 6.8		mA mA
Dimensions	LGA-16 Package		5x5x1.5		mm

Table 2.2.: Selection of mechanical and electrical specifications of gyroscopes LPR550AL and LY550ALH by ST [ST09b], [ST09a]

2.3.4. Magnetometers

A magnetometer is a device that measures magnetic field direction and strength. Various devices and measurement principles exist, capable of measuring magnetic fields in a range of 10^{-15}T to 10T. Common MEMS magnetometers are based on the hall effect or resistors that change resistivity if a magnetic field is applied.

Apart from sensor errors due to bias, scale and misalignment deviations, magnetic field measurements are distorted by ferromagnetic elements in the vicinity of the sensor, labeled hard iron and soft iron distortions. Hard iron distortions are caused by materials, which exhibit a constant, additive magnetic field, whereas soft iron distortions are the result of materials, influencing or distorting but not necessarily generating the field [Kon09]. These errors have to be considered during calibration and make this process considerably more complicated.

The magnetometer HMC1053 by Honeywell utilized for the present work is a very compact 3-axis magnetometer including offset and set/reset straps. The HMC1053 basically contains a magneto-resistive Wheatstone bridge for every spatial direction, converting an incident magnetic field to a differential voltage output. Table 2.3 presents a selection of mechanical and electrical specifications of the sensing bridge elements taken from the accompanying datasheet. The HMC1053 provides differential voltage outputs, which have to be amplified and low-pass filtered

Parameter	Test condition	Min.	Typ.	Max.	Unit
Supply voltage		1.8	3.0	20	V
Resistance	Bridge current = 10mA	800	1000	1500	Ω
Field range		-6		+6	gauss
Sensitivity	Set/Reset Current = 0.5A	0.8	1.0	1.2	mV/V/gauss
Bandwidth	Magnetic signal		5		MHz
Noise density	@ 1kHz, Supply Volt. = 5V		50		nV/ $\sqrt{Hz}$
Dimensions	16-Pin LCC Package		7.4x7.4x2.8		mm

Table 2.3.: Selection of mechanical and electrical specifications of magnetometer HMC1053 by Honeywell [Hon06]

before sampling and A/D-conversion. The amplification is necessary due to the weak magnetic field strength on the earth's surface, which is shown in the following calculation.
The magnetic earth field ranges from less than 0.3 gauss to over 0.6 gauss [Can08]. A sensor supply voltage of 3.0V would give a sensor sensitivity of approximately 0.33mV/gauss. Then a local magnetic earth field strength of 0.5 gauss (50μT) and a perfectly aligned sensor axis would produce a differential voltage of 0.165mV. Only a fraction of that voltage is available if the vector is not perfectly aligned with the sensitive axis. In comparison, a 12 bit A/D-converter with 5V reference voltage has a resolution of approximately 1.22mV. Thus, measurements of the earth magnetic field are smaller than the A/D-converter resolution.
In order to achieve a sufficient resolution the output signal is amplified with a factor of approximately 371 by a differential amplifier INA333 by Texas Instruments (TI). The low-pass filter cut off frequency is set to 100 Hz to avoid anti-aliasing during A/D-conversion.

2.3.5. Sensor board configuration

This section provides information about the sensor board that has been used throughout the thesis. The sensor PCB has a two layer structure and dimensions 65 × 35 × 8 mm. It is equipped with three orthogonal gyroscopes, accelerometers, magnetometers, components for analogue signal processing and power regulation. Figure 2.6 shows the sensor board. In order to obtain very accurate sensor measurements, amplified gyroscope outputs and a full-scale of $\pm 2g$ accelerometer sensitivity is used.

2.4. Computer arithmetics

In a digital computer numbers are represented by strings of zeros and ones and the hardware can only perform a simple and primitive set of Boolean operations [Okl92]. Thus, to perform complex operations on "real-world" numbers in decimal format, it is necessary to find an adequate binary representation and to provide arithmetics based on boolean operations. The following subsections present two different approaches for number representation and briefly describes accompanying arithmetics, condensed from [Okl92] and [VvVB04].

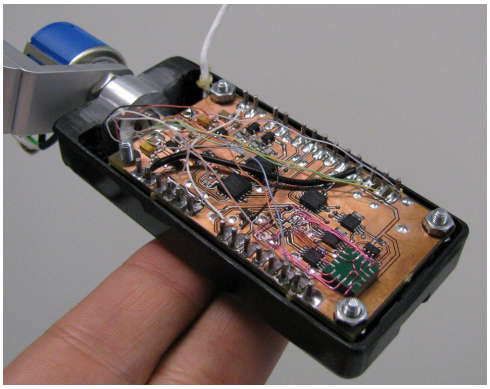

Fig. 2.6.: Sensor module PCB

2.4.1. Floating-point numbers

In computer arithmetics, a real number x can be approximated by a pair of integers (s,e) in exponential form according to

$$x \approx s \cdot 2^e \ , \tag{2.56}$$

where s is the significant and e is the exponential. If e is not known in advance and varies during runtime the number is denoted as a floating-point number, because the radix point of x changes or floats according to e. In order to store the floating-point number in binary form, a specific number of bits is assigned to the binary representation of s and another string of bits is used to store the binary representation of e.
Floating-point numbers are standardized in IEEE 754. This standard includes the definition of different formats, which assign the number of bits for significant and exponent. A single precision floating-point number is 32 bits long with a 24 bit significant. Double precision numbers assign 64 bits and have a 53 bit significant.
In order to express floating-point arithmetics with boolean bit operations the significants and exponents have to be treated separately. This includes scaling, rounding and normalization of interim results. To accelerate these operations often Floating-Point Units (FPU) are used. These units are special processors, which are able to perform a mathematical operation during one processor clock cycle. If no FPU is used, the operations have to be emulated with software, which increases computation time considerably.

2.4.2. Fixed-point numbers

In contrast to floating-point numbers fixed-point representations have a fixed position of the radix point, meaning the exponent e is fixed and has to be know in advance. A certain number of bits is assigned to represent the integer in front of the radix point and the remaining bits represent the fractional part.
Fixed-point arithmetics include only basic integer operations and shifting of bits, briefly de-

scribed in [ARM96]. Thus, mathematical operations can be processed very fast. Due to the fixed position of the radix point only a small range of numbers can be represented compared to floating-point representations. Choosing an appropriate scaling for fixed-point numbers, which avoids overflows but remains accurate enough is the major challenge using this number representation. Depending on the processor structure, usually 8, 16, 32 or 64 bits are assigned for a fixed-point number.

3. Mathematical modeling

3.1. Introduction

A system that maintains an accurate estimate of an objects orientation is commonly referred to as an Attitude Heading Reference System (AHRS). Originally these systems were utilized for navigation of aircrafts, spaceshuttles, submarines, ships and missiles. Recent advances in production technologies and miniaturization brought up MEMS sensors, which enabled the development of miniature, light weight and less expensive AHRS. That circumstance enhanced the range of possible applications including human motion tracking as well as motion sensitive controls for mobile devices or game consoles.
Accoring to [Woo07], early AHRS are based on a stabilized platform using gimbals, which allows the platform to rotate in all three axes. The platform is equipped with gyroscopes detecting any occurring rotations. Torque motors receive the gyroscope outputs and rotate the gimbals to cancel out such rotations and keep the platform aligned with the global frame. The orientation is then simply obtained by reading the angles of adjacent gimbals. This principle is also used for Inertial Navigation Systems (INS), where the platform also includes accelerometers to track position and velocity. This setup is depicted in Figure 3.1.

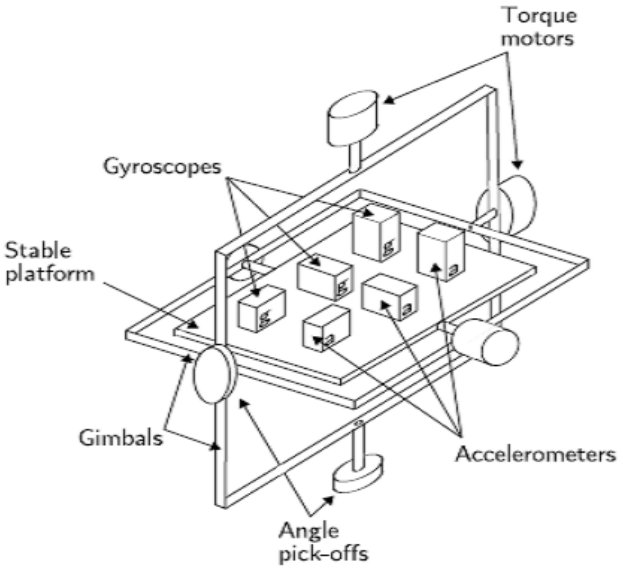

Fig. 3.1.: Stable platform principle [Woo07]

Another approach is to rigidly mount sensors onto the device and track the sensor outputs in body frame rather than global frame. This strapdown principle requires to process the sensor outputs in order to yield attitude information in global frame. Common sensor types used in strapdown configurations are gyroscopes, accelerometers, magnetometers and optical horizon or star trackers. Often a combination of different sensor types is used to either enable 3-DOF attitude determination or to increase accuracy. Additionally GPS or optical tracking systems are sometimes used to decrease errors in attitude estimation, having the disadvantage of being dependent on artificially generated, external signals.
According to [Woo07], strapdown systems have the advantage of being physically smaller and mechanically less complex compared to platform systems. Disadvantageous is a higher computational demand, but computational cost has decreased tremendously, which make them the preferred type of AHRS nowadays.
For the present work it is desired to design a small, low-power and self-contained AHRS in strapdown configuration. The following sections describe the mathematical modeling and derivation of the attitude estimation algorithm. Section 3.2 describes, which attitude information can be extracted from a single sensor type and Section 3.3 presents how attitude can be estimated by a combination of multiple sensors. Afterwards, an explicit attitude estimation algorithm is derived in Sections 3.4-3.6, which includes a simple human motion model, a process model and the Kalman Filter design, respectively. The presented algorithm is almost identical to an approach, presented by Yun et al. in [YB06] and [YBM08]. Advantages of the chosen algorithm as well as alternative approaches are presented throughout the sections and finally discussed in Section 5.

3.2. Absolute attitude determination

Several ways exist to determine attitude from inertial or magnetic field measurements arranged in strapdown principle. This section presents how attitude may be determined from a single sensor type and discusses accompanying constraints and limitations. These descriptions build the foundation for all upcoming approaches, which combine two or three of them.

Attitude determination with gyroscopes
A simple and intuitive way to determine attitude is to use a 3-axis gyroscope, which measures angular rates in all three spatial dimensions. By simply integrating the sensor outputs it is possible to determine attitude if initial conditions are known. The major disadvantages of this approach are the need to know initial conditions and particularly the inevitable integration of angular rates. Any error in the gyroscope outputs produce an attitude estimation error, that grows proportional to time, which is illustrated by the following calculation.
Assuming a system is at rest, a gyroscope attached to it is supposed to produce an output of $\omega = 0°/s$. But due to a small bias error it shows $\omega = 0.1°/s$. Integrating the gyroscope output produces an angle that grows proportional to time according to $\alpha = \omega t$. Thus, after $t = 60s$ the angle calculated from the gyroscope output is 6° and after 10 minutes it has grown to 60°.

Furthermore, the error accumulates for gyroscope measurements in multiple spatial dimensions. As described in Section 2.3.3, gyroscopes have various error types some of them time-varying so they can not be eliminated permanently. The effect of integrating errors reduces, if the system undergoes dynamical rotations and the actual angular rate is big compared to the errors.
It can be concluded that attitude determination with gyroscopes is only accurate for short time spans after initialization and during times of high dynamical motion. Additionally gyroscope measurements are relatively robust against linear accelerations.

Attitude determination by vector observations
An alternative method to determine attitude is to use vector observations of a physical quantity in body and global frame. Several quantities might be used, but only gravity vector $\vec{g}$ and earth magnetic field vector $\vec{B}$ are considered at this point.
Acceleration measurements of components of $\vec{g}^B$ in body frame are related to components of $\vec{g}^N$ in global frame by an objects attitude represented as a rotation matrix $\mathbf{R}_N^B$ according to

$$\vec{g}^B = \mathbf{R}_N^B \vec{g}^N \ . \tag{3.1}$$

If $\vec{g}^N$ is known and $\vec{g}^B$ is measured by an accelerometer it remains to find $\mathbf{R}_N^B$, relating both vector observations. This comes along with two major difficulties. Accelerometers measure force on a proof mass, which is a combination of a force caused by gravity and a specific force due to dynamic motions:

$$f_{measure} = f_{specific} + f_{gravity} \ . \tag{3.2}$$

Thus, attitude determination by gravity observation is only accurate if the specific force is considerably small, which equals a system almost at rest. The second limitation is a full 3-DOF attitude representation takes at least three independent parameters, whereas vector observations provide only two [Hal03]. Even though vectors have three parameters, one parameter is needed to fulfill a magnitude constraint, since gravity has a well defined value. This can also be illustrated by observing that Yaw cannot be determined from gravity measurements, since a rotation about Yaw is not observable in sensor outputs.
Attitude determination with magnetometers share the same properties as with accelerometers. As long as no external magnetic fields distort the earth magnetic field it is possible to estimate attitude for two axis. Again, a rotation about the magnetic field vector is not observable in sensor outputs.

3.3. Attitude estimation

This sections presents different approaches to estimate attitude from inertial and magnetic field sensor data. They can be roughly classified into static and dynamic attitude estimation algorithms [Hal03]. Static estimation algorithms subsume algorithms to compute attitude from

vector observations at a single time instance. Dynamic attitude estimators commonly base on a state estimation setup, merging the sensor data by a dynamic system model.

3.3.1. Static attitude estimation

Static attitude estimation algorithms are originated in spacecraft attitude systems, where measurements of magnetic fields, sun position and star constellations are available as vector observations. Recall from Section 3.2 that it takes three independent parameters to determine attitude and a single vector observation provides two parameters. In order to find a third independent parameter it is necessary to combine at least two vector observations, inevitably producing an overdetermined problem. Thus attitude determination with vector observations always turn to an estimation problem, commonly referred to as Whaba's problem.
In 1965, Grace Whaba formulated an optimality criterion that seeks to find an orthonormal matrix A which minimizes the cost function

$$J(\mathbf{A}) = \frac{1}{2}\sum_i a_i |\mathbf{b}_i - \mathbf{A}\mathbf{r}_i|^2 \ , \tag{3.3}$$

where $\mathbf{b}_i$ is a set of unit vectors observed in body frame, $\mathbf{r}_i$ are the corresponding vectors in a reference frame and a_i is a set of positive weights [Wha66]. Several algorithms exist to solve Whaba's problem and just a few are mentioned and briefly described in the following. A more detailed and complete description of algorithms can be found in [Shu06], [Wer78] and [Mar02].
A simple, deterministic but suboptimal approach using two vector observations is the TRIAD algorithm, well described in [Hal03]. The TRIAD algorithm basically forms two orthonormal frames (triads) by using one vector observation as the first basis vector and creating the successive axis by vector orthogonality principles. It remains to calculate the direction-cosine matrix, which relates the body frame triad to the reference frame triad. This involves only simple, algebraic operations. The TRIAD algorithm is considered suboptimal because it requires that one vector observation is exact, representing the first basis vector. The second vector observation just adds information to yield the third unknown parameter to determine attitude.
Sensor measurements naturally contain errors so it is usually desired to use all available information in measurements. In addition, it might be advantageous to use more than two vector observations to further decrease the estimation error. This approach is pursued with statistical methods, aiming to find estimators which minimize Whaba's cost function, defined in Equation (3.3).
According to [Shu06] several analytical, optimal least-squares solutions were published immediately after the problem was stated. They are all based on a rewritten problem formulation, simplifying the minimization. The computations, including matrix decompositions and inversions, were not suitable for spacecraft applications due to their computational demands. Davenport reformulated the method by using a quaternion attitude representation and showed that the minimization can be solved by finding the largest eigenvalue of a 4x4 matrix, constructed from vector observations and weighting factors. This method, commonly named q-method, reduces

the computational demand but is still numerically intensive. Shuster and Oh derived the QUEST algorithm (QUaternion-ESTimator), which is an approximation of the q-method, providing a more efficient way to solve the eigenproblem. A problem with QUEST is that the algorithm contains singularities, which can be avoided by extra calculations.
Alternatively, the minimization of Whaba's cost function can be performed iteratively, e.g. by Newton's method. Iterative approaches require an initial attitude matrix and iteratively adapt it to minimize the cost function to a predefined lower bound. The major disadvantages of iterative methods are: convergence may not be guaranteed and computation time until convergence varies. These are properties, which are not desirable for real-time applications.
Static attitude determination algorithms require sensor measurements only at a single time instance and some approaches are capable of considering noise properties of sensors. This could also include to detect disturbed measurements and adapt noise properties. However, since no information about the system dynamics is used these algorithms are very sensitive to disturbances.

3.3.2. Dynamic attitude estimation

Dynamic attitude estimation approaches rely on a process model considering motion dynamics and specific application properties to estimate attitude from noisy or disturbed sensor measurements. These specific properties could include individual motion restrictions or typical time constants. They are different for every application, thus only the fusion of dynamic motion models and sensor measurements are presented in the following.
Motion dynamic models are generally derived from rigid body kinematics, describing how attitude changes depending on current angular rates of the object. The following paragraphs describe the derivation of motion dynamics for an Euler angle and quaternion attitude representation, presented in [Dum99] and [Mar00] respectively.
Angular rates about x-, y- and z-axis are denoted as p, q, r in body frame and ω_x, ω_y, ω_z in earth frame, respectively. The Euler rates $\dot{\psi}$, $\dot{\theta}$ and $\dot{\phi}$ are related to angular velocities of the earth frame axes according to

$$\mathbf{E}_\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \mathbf{R}_z^T \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}_y^T \mathbf{R}_z^T \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} , \tag{3.4}$$

where the matrices $\mathbf{R}_i$ are defined in Section 2.2.3. It has to be considered that Euler rates are not identical to angular earth rates, because they are defined for a fixed order of rotations about separate axes. Angular body rates are obtained by rotating $\mathbf{E}_\omega$ into body frame

$$\mathbf{B}_\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z \mathbf{E}_\omega \tag{3.5}$$

and plugging (3.4) in the latter equation gives

$$\mathbf{B}_\omega = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \mathbf{R}_x\mathbf{R}_y \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}_x \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} =$$
$$= \begin{bmatrix} -\sin(\phi) \\ \sin(\theta)\cos(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix} \dot{\psi} + \begin{bmatrix} 0 \\ \cos(\phi) \\ -\sin(\phi) \end{bmatrix} \dot{\theta} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \dot{\phi} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} . \tag{3.6}$$

Solving for Euler rates by multiplying both sides of Equation (3.6) with the matrix inverse from left gives

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} . \tag{3.7}$$

In the latter equation it becomes obvious how the singularity at a Pitch angle of $\theta = \pm\frac{\pi}{2}$ cause problems. For this case certain matrix elements contain a division by zero.
For the derivation of quaternion derivatives in dependence of angular body rates it is helpful to define the following quaternions:

- $\mathbf{m}_{B(t)}^{E(t)}$: rotates a vector in body frame into earth frame at time t
- $\mathbf{m}_{B(t+\Delta t)}^{E(t+\Delta t)}$: rotates a vector in body frame into earth frame at time t+Δt
- $\mathbf{n}_{B(t+\Delta t)}^{B(t)}$: rotates a vector in body frame at time t+Δt into body frame at time t

Each of the listed quaternions represent a vector rotation, depicted in Figure 3.2.

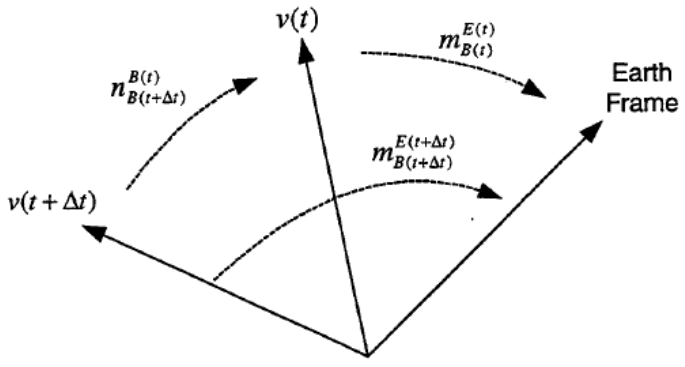

Fig. 3.2.: Determining quaternion derivatives [Mar00]

The differential of $\mathbf{m}_B^E$ at two successive time steps denoted as t+Δt and t is

$$d\mathbf{m}_B^E = \mathbf{m}_{B(t+\Delta t)}^{E(t+\Delta t)} - \mathbf{m}_{B(t)}^{E(t)} \ , \tag{3.8}$$

where

$$\mathbf{m}_{B(t+\Delta t)}^{E(t+\Delta t)} = \mathbf{m}_{B(t)}^{E(t)} \mathbf{n}_{B(t+\Delta t)}^{B(t)} \ . \tag{3.9}$$

Plugging Equation (3.9) into (3.8) gives

$$d\mathbf{m}_B^E = \mathbf{m}_{B(t)}^{E(t)} (\mathbf{n}_{B(t+\Delta t)}^{B(t)} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}) \ . \tag{3.10}$$

Assuming $\mathbf{n}_{B(t+\Delta t)}^{B(t)}$ represents a rotation of a small angle $d\alpha$ about a vector defined by u_x,u_y and u_z, the following approximation holds:

$$\mathbf{n}_{B(t+\Delta t)}^{B(t)} = \begin{bmatrix} \cos(\frac{d\alpha}{2}) \\ u_x \sin(\frac{d\alpha}{2}) \\ u_y \sin(\frac{d\alpha}{2}) \\ u_z \sin(\frac{d\alpha}{2}) \end{bmatrix} \approx \begin{bmatrix} 1 \\ u_x \frac{d\alpha}{2} \\ u_y \frac{d\alpha}{2} \\ u_z \frac{d\alpha}{2} \end{bmatrix} \ . \tag{3.11}$$

Combining Equation (3.10) and (3.11) gives

$$d\mathbf{m}_B^E = \frac{1}{2} \mathbf{m}_{B(t)}^{E(t)} \begin{bmatrix} 0 \\ d\alpha u_x \\ d\alpha u_y \\ d\alpha u_z \end{bmatrix} \ . \tag{3.12}$$

Dividing both sides with the time differential dt leads to an expression of quaternion derivatives as a function of body rates.

$$\dot{\mathbf{m}}_B^E = \frac{1}{2} \mathbf{m}_{B(t)}^{E(t)} \begin{bmatrix} 0 \\ p \\ q \\ r \end{bmatrix} \ . \tag{3.13}$$

Replacing $\mathbf{m}_{B(t)}^{E(t)}$ by its real and complex components w, x, y, z and reordering Equation (3.13) gives

$$\begin{bmatrix} \dot{w} \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} \ . \tag{3.14}$$

As stated above, motion dynamic models usually depend on angular body rates, which are directly measurable with gyroscopes. Alternatively they can be calculated from time-dependent attitude variations determined by vector observations. [GEEPP00] and [TP02] presented such attempts. Anyhow, this approach works only for low dynamical motions due to the sensitiveness of vector observations to disturbances.
Recall from Section 3.2 that attitude determination with gyroscopes as the only type of sensor lacks long-term stability and requires accurate initial conditions. For this reason, it is common to use a combination of gyroscopes, accelerometer and magnetometer. Gyroscopes are precise during short terms and high dynamical motions, but integration of sensor errors cause a drift in time. Vector observations produce a non-drifting attitude estimate, but are prone to disturbances caused by motions or disturbing magnetic fields. Thus, it seems logical to combine the complementary sensor error characteristics with an algorithm to yield the best possible attitude estimate. This kind of sensor fusion is commonly done by a Kalman filter, which considers statistical error properties of the process model and sensors. A drawback, which is generally connected to geometric multi-spatial problems are nonlinear relationships of system variables, so an Extended or an Uncented Kalman Filter have to be used.

3.4. Human motion modeling

In order to design an observer which accurately tracks human motions it is necessary to derive a process model of human motion dynamics. Such a model ideally depends on the human skeleton, considering kinematic and dynamic constraints. A lot of research has been performed on these models, but they are complex and often combined with external image-tracking systems [WP98]. Thus an alternative model has to be found, which is simple but still accurate enough to model human motion dynamics. Yun et al. showed that a first-order linear model driven by white noise is a simple and sufficiently accurate approach to model angular velocities of limb segments [YB06]. The model equation is given by

$$\tau\dot{\omega} + \omega = v \ , \tag{3.15}$$

where τ is a time constant describing how fast the limb segment typically moves, ω is its angular velocity and v is a white noise process. This kind of model is commonly named angle random walk process. The basic idea is that human limbs have a mass and moment of inertia, thus velocities change continuously and with characteristic time constants.

3.5. Process model

3.5.1. General considerations

In order to design a state observer for attitude tracking, a system model is required. This system or process model is derived throughout the following sections, which basically follows an

approach presented by Yun et al in [YB06] and [YBM08]. This was mentioned at the beginning of the current chapter and is repeated just for clarification. Various different modeling approaches exist for a broad range of applications. The process model for the present work is selected with the aim to yield an accurate model, which is as compact as possible. The latter consideration is elementary for microcontroller implementation of the state observer.

3.5.2. Definition of state vector

In order to derive a state space representation of the process a state vector has to be defined. The state vector contains variables describing the actual attitude estimate and states representing the process dynamics. The present work is based on a quaternion attitude representation, describing how the body has to be rotated to be aligned with the NED coordinate system. Quaternions have no singularities, are computationally efficient and eliminate the need to compute trigonometric functions.
Alternatively Euler angles or a direction cosine matrix could be used to represent attitude. Euler angles have the advantage of being easy to imagine and using the minimum number of parameters to describe attitude. The major disadvantage are singularities, which occur if Pitch is close to $\pm\frac{pi}{2}$. Direction cosine matrices do not suffer from singularities, but include nine variables to describe attitude. This is not desirable, since keeping track of nine variables increase the computational demands.
The process dynamics can be considered by keeping track of the actual angular body rates. Thus, the state vector is defined to contain the angular body rates and a quaternion, representing the actual attitude:

$$\hat{\mathbf{x}} = \begin{bmatrix} p & q & r & w & x & y & z \end{bmatrix}^T , \tag{3.16}$$

where

- p : angular rate about sensor x-axis
- q : angular rate about sensor y-axis
- r : angular rate about sensor z-axis
- w: scalar part of quaternion
- x, y, z : imaginary part of quaternion

In order to improve estimation accuracy it is possible to consider sensor errors in the state equations. This is common practice, if proper sensor error models exist and if they are observable in the sensor data. Then, the state vector is enhanced by sensor errors, which typically includes variables for sensor bias and scale factors. [Far08] presents such approaches. Every additional state increases the number of necessary computations considerably, so this approach is not followed.

3.5.3. System equations

The system equations model the process dynamics and describe how the state vector evolves in time. Thus, the first derivatives of the state vector have to be found according to

$$\dot{\hat{\mathbf{x}}} = \mathbf{f}(\hat{\mathbf{x}}) \ . \tag{3.17}$$

Since it is not predictable how the sensor module will change its angular rates, a simplified human motion model described in Section 3.4 is used. Angular accelerations are modeled as an angle random walk process according to

$$\dot{p} = -\frac{1}{\tau_p} p + \frac{1}{\tau_p} s_p \tag{3.18}$$

$$\dot{q} = -\frac{1}{\tau_q} q + \frac{1}{\tau_q} s_q \tag{3.19}$$

$$\dot{r} = -\frac{1}{\tau_r} r + \frac{1}{\tau_r} s_r \ , \tag{3.20}$$

where τ_p, τ_q and τ_r are time constants, describing how fast angular rates typically change for each axis. These constants depend on the current application and could be determined for certain application setups. For the present work it is desired to use the sensor module without any additional information needed, so the time constants are equally set to τ_i= 0.5s. This value is determined iteratively and showed good results during tests. The driving noise processes s_p, s_q and s_r of the angular random walks are assumed to be independent and white, which requires a verification and further investigations.
It remains to find the quaternion derivatives, which describe how attitude changes based on the current attitude and angular body rates. This is shown in 3.3.2, where quaternion rates are related to angular body rates by

$$\begin{bmatrix} \dot{w} \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} \ . \tag{3.21}$$

Equation (3.21) is only valid for unit quaternions, so it is important to normalize the quaternion before applying the system equations. In summary, the (non-linear) state equations are given by

$$\dot{p} = -\frac{1}{\tau_p} p + \frac{1}{\tau_p} s_p \tag{3.22}$$

$$\dot{q} = -\frac{1}{\tau_q} q + \frac{1}{\tau_q} s_q \tag{3.23}$$

$$\dot{r} = -\frac{1}{\tau_r} r + \frac{1}{\tau_r} s_r \tag{3.24}$$

$$\dot{w} = \frac{1}{2}(-xp - yq - zr) \tag{3.25}$$

$$\dot{x} = \frac{1}{2}(wp - zq + yr) \tag{3.26}$$

$$\dot{y} = \frac{1}{2}(zp + wq - xr) \tag{3.27}$$

$$\dot{z} = \frac{1}{2}(-yp + xq + wr) \ . \tag{3.28}$$

It has to be mentioned, that the quaternion state equations do not include any process noise. This is because they are derived analytically and describe a correct mathematical relationship, as long as the small angle approach from Section 3.3.2 is not violated. Thus, the state equations have to be computed at a rate, such that the attitude angles only change a little in between two samples.

3.5.4. Definition of system outputs

The sensor module provides measurements of angular rates, accelerations and magnetic field components. By defining the same values as the system output, it is possible to calculate the difference of real and predicted variables and process it with a Kalman filter. So it is natural to choose the system outputs identical to the sensor measurements, which gives

$$\hat{\mathbf{y}} = \begin{bmatrix} p & q & r & a_x & a_y & a_z & m_x & m_y & m_z \end{bmatrix}^T \ , \tag{3.29}$$

where

- p : angular rate about sensor x-axis
- q : angular rate about sensor y-axis
- r : angular rate about sensor z-axis
- a_x: x component of acceleration
- a_y: y component of acceleration
- a_z: z component of acceleration
- m_x: x component of local magnetic field
- m_y: y component of local magnetic field
- m_z: z component of local magnetic field

In Section 3.5.5 it is described how the output vector can be computed from the state vector. The resulting equations are highly nonlinear so Yun et al. proposed an alternative output vector according to

$$\hat{\mathbf{y}} = \begin{bmatrix} p & q & r & w & x & y & z \end{bmatrix}^T \ , \tag{3.30}$$

where

- p : angular rate about sensor x-axis
- q : angular rate about sensor y-axis
- r : angular rate about sensor z-axis
- w: scalar part of quaternion
- x, y, z : imaginary part of quaternion

Since an attitude quaternion is not measurable directly it has to be derived by preprocessing accelerometer and magnetometer measurements. Preprocessing includes a static attitude estimation approach to provide such a "measurement" quaternion. This simplifies the output equations significantly, which is shown in Section 3.5.6.

3.5.5. Output equations I

The output equations derived in this section describe how the output vector $\hat{y}$ defined in Equation (3.29) depends on the internal states $\hat{x}$. The angular body rates are part of the state vector, so the first three elements of the output vector are identical to the first three elements of the state vector. It remains to compute gravity and magnetic field vectors in body frame.
In order to compute gravity and magnetic field components in body frame, the earth frame representations have to be rotated into body frame by the current attitude estimate. Since the state quaternion contains information how to rotate a vector from body to earth frame, the conjugate quaternion has to be used. Reconsider from Section 2.2.4, that a quaternion rotation of a vector can be written in matrix form according to

$$\vec{a}^B = \mathbf{M}(q^*)\vec{g}^N = \begin{bmatrix} w^2+x^2-y^2-z^2 & 2wz+2xy & -2wy+2xz \\ -2wz+2xy & w^2-x^2+y^2-z^2 & 2xw+2yz \\ 2wy+2xz & -2wx+2yz & w^2-x^2-y^2+z^2 \end{bmatrix} \vec{g}^N \ , \tag{3.31}$$

where the quaternion conjugate is plugged into the matrix representation $\mathbf{M}$(q). The gravity vector in body frame is denoted as $\vec{a}^b$ and the gravity vector in earth frame is $\vec{g}^e$. The same quaternion rotation has to be performed for magnetic field components in body and earth frame. Equation (3.31) is only valid for unit quaternions, so the quaternion has to be normalized before.

In summary, the output equations are given by

$$\hat{y}_1 = p + v_1 \tag{3.32}$$
$$\hat{y}_2 = q + v_2 \tag{3.33}$$
$$\hat{y}_3 = r + v_3 \tag{3.34}$$
$$\hat{y}_4 = (w^2 + x^2 - y^2 - z^2)g_x + (2wz + 2xy)g_y + (-2wy + 2xz)g_z + v_4 \tag{3.35}$$
$$\hat{y}_5 = (-2wz + 2xy)g_x + (w^2 - x^2 + y^2 - z^2)g_y + (2wx + 2yz)g_z + v_5 \tag{3.36}$$
$$\hat{y}_6 = (2wy + 2xz)g_x + (-2wx + 2yz)g_y + (w^2 - x^2 - y^2 + z^2)g_z + v_6 \tag{3.37}$$
$$\hat{y}_7 = (w^2 + x^2 - y^2 - z^2)B_x + (2wz + 2xy)B_y + (-2wy + 2xz)B_z + v_7 \tag{3.38}$$
$$\hat{y}_8 = (-2wz + 2xy)B_x + (w^2 - x^2 + y^2 - z^2)B_y + (2wx + 2yz)B_z + v_8 \tag{3.39}$$
$$\hat{y}_9 = (2wy + 2xz)B_x + (-2wx + 2yz)B_y + (w^2 - x^2 - y^2 + z^2)B_z + v_9 \ . \tag{3.40}$$

The v_i elements stand for measurement noise, modeling the uncertainty of sensor measurements. They can be taken from sensor datasheets or determined by measurements. Alternatively they can be used to model sensor uncertainties due to appearing disturbances, which is further discussed in Section 5.

3.5.6. Output equations II

The output equations derived in this section describe how the output vector $\hat{\mathbf{y}}$ defined in Equation (3.30) depends on the internal states $\hat{\mathbf{x}}$. Analogously to the output equations I, the first three elements of the output vector are identical to the first three elements of the state vector. It remains to compute a "measurement" quaternion from acceleration and magnetic field measurements. Therefore a static attitude estimation algorithm has to be designated.
For the present work the TRIAD algorithm is used, which elegantly combines two vector observations with a low computational cost. The property of TRIAD to require one observation to be very accurate fits with the available sensor types. Disturbing magnetic fields can easily overlay the earth magnetic field due to its low magnitude. Thus, it is advantageous to put more trust in accelerometer measurements and determine only the angle about the vertical axis with magnetometer measurements.
An iterative approach is not suitable for real-time implementation with limited computation resources. It can not be guaranteed, that the algorithm converges successfully during a predefined time interval. In [YB06] Yun et al. propose to use the QUEST algorithm, which optimally combines noisy vector observations, but at a higher computational cost. Furthermore, according to [Hal03] the QUEST algorithm has approximately the same accuracy as the suboptimal TRIAD given only two vector observations.
Recall from Section 3.3.1, that the TRIAD algorithm basically forms two orthonormal frames by using one vector observation as the first basis and creating the successive axis by vector orthogonality principles. The first basis vector of the body frame is simply given by measurements

of the acceleration sensors

$$\hat{r}_1 = \vec{a}^B \ . \tag{3.41}$$

The second basis vector is constructed by taking the normalized cross product of $\hat{r}_1$ and the magnetometer measurements:

$$\hat{r}_2 = \frac{(\vec{a}^B \times \vec{m}^B)}{|\vec{a}^B \times \vec{m}^B|} \ . \tag{3.42}$$

Thus $\hat{r}_2$ is orthogonal to $\hat{r}_1$ and the magnetometer measurements just add information how $\hat{r}_2$ is orientated about the axis formed by accelerometer measurements. To complete the body triad, the third basis vector is constructed by taking the cross product of $\hat{r}_1$ and $\hat{r}_2$:

$$\hat{r}_3 = \hat{r}_1 \times \hat{r}_2 \ . \tag{3.43}$$

The same procedure has to be followed to construct the earth triad from gravity and magnetic field vector in the reference frame:

$$\hat{s}_1 = \vec{g}^N \tag{3.44}$$

$$\hat{s}_2 = \frac{(\vec{g}^N \times \vec{B}^N)}{|\vec{g}^N \times \vec{B}^N|} \tag{3.45}$$

$$\hat{s}_3 = \hat{s}_1 \times \hat{s}_2 \ . \tag{3.46}$$

Body and earth frame triads are related by a orientation matrix A, which transforms the body triad into earth triad according to

$$\begin{bmatrix} \hat{s}_1 & \hat{s}_2 & \hat{s}_3 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \hat{r}_1 & \hat{r}_2 & \hat{r}_3 \end{bmatrix} \ . \tag{3.47}$$

By considering that both triads are a set of orthonormal basis vectors, their inverse are equal to their transpose, thus $\mathbf{A}$ can be expressed as

$$\mathbf{A} = \begin{bmatrix} \hat{s}_1 & \hat{s}_2 & \hat{s}_3 \end{bmatrix} \begin{bmatrix} \hat{r}_1 & \hat{r}_2 & \hat{r}_3 \end{bmatrix}^T \ . \tag{3.48}$$

The procedure to derive a quaternion representation of the orientation matrix $\mathbf{A}$ is described in Appendix A. After calculating the "measurement" quaternion the measurement equations are simply given by

$$\hat{y} = \hat{x} + v \ , \tag{3.49}$$

where $v \in \mathbb{R}_{7\times 1}$ is the measurement noise vector with covariance matrix $\mathbf{R} \in \mathbb{R}_{7\times 7}$.

3.6. Kalman Filter design

3.6.1. General considerations

In the previous sections a process model in state space representation is derived. The system equations turn out to be non-linear thus an Extended Kalman Filter has to be implemented. Figures 3.3 and 3.4 illustrate the filter setups for output equations I and II of the process model introduced by Yun et al. in [YB06].

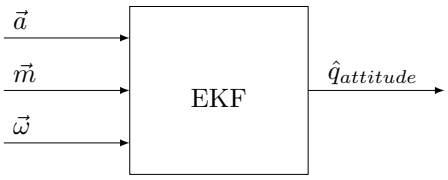

Fig. 3.3.: Kalman Filter setup I

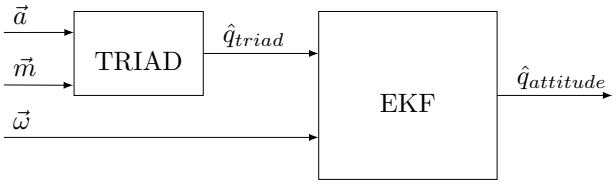

Fig. 3.4.: Kalman Filter setup II

3.6.2. Discretized process model

In order to implement the Kalman filter on a microcontroller, the process model has to be discretized. For the present work, this is done by using a zero-order hold approach

$$f(t) = f(t_k) \qquad t_k \leq t < t_{k+1} \tag{3.50}$$

and approximating derivatives with Euler's method according to

$$\dot{x} \approx \frac{x_{k+1} - x_k}{dt} \ , \tag{3.51}$$

where dt is the sampling period [AW96]. Replacing derivatives in the state equation with Eulers method and reordering the result gives

$$p_{k+1} = (1 - \frac{dt}{\tau_p})p_k + \frac{dt}{\tau_x}s_{pk} \tag{3.52}$$

$$q_{k+1} = (1 - \frac{dt}{\tau_q})q_k + \frac{dt}{\tau_y}s_{qk} \tag{3.53}$$

$$r_{k+1} = (1 - \frac{dt}{\tau_r})r_k + \frac{dt}{\tau_z}s_{rk} \tag{3.54}$$

$$w_{k+1} = \frac{dt}{2}\left(-x_k p_k - y_k q_k - z_k r_k\right) + w_k \tag{3.55}$$

$$x_{k+1} = \frac{dt}{2}\left(w_k p_k - z_k q_k + y_k r_k\right) + x_k \tag{3.56}$$

$$y_{k+1} = \frac{dt}{2}\left(z_k p_k + w_k q_k - x_k r_k\right) + y_k \tag{3.57}$$

$$z_{k+1} = \frac{dt}{2}\left(-y_k p_k + x_k q_k + w_k r_k\right) + z_k \ . \tag{3.58}$$

The measurement equations do not contain any derivatives, thus time-dependent variables are assumed to be piecewise constant over the sampling interval according to Equation (3.50).

3.6.3. Linearized process model

The non-linear state equations have to be linearized to process them with an Extended Kalman filter. This is done by taking the partial derivatives of the state equation at the actual state estimates:

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}\right|_{\hat{x}_k} = \begin{bmatrix} 1-\frac{dt}{\tau_p} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-\frac{dt}{\tau_q} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-\frac{dt}{\tau_r} & 0 & 0 & 0 & 0 \\ -\frac{dt}{2}x_k & -\frac{dt}{2}y_k & -\frac{dt}{2}z_k & 1 & -\frac{dt}{2}p_k & -\frac{dt}{2}q_k & -\frac{dt}{2}r_k \\ \frac{dt}{2}w_k & -\frac{dt}{2}z_k & \frac{dt}{2}y_k & \frac{dt}{2}p_k & 1 & \frac{dt}{2}r_k & -\frac{dt}{2}q_k \\ \frac{dt}{2}z_k & \frac{dt}{2}w_k & -\frac{dt}{2}x_k & \frac{dt}{2}q_k & -\frac{dt}{2}r_k & 1 & \frac{dt}{2}p_k \\ -\frac{dt}{2}y_k & \frac{dt}{2}x_k & \frac{dt}{2}w_k & \frac{dt}{2}r_k & \frac{dt}{2}q_k & -\frac{dt}{2}p_k & 1 \end{bmatrix} \tag{3.59}$$

$$\mathbf{L}_k = \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{s}}\right|_{\hat{x}_k} = \begin{bmatrix} \frac{dt}{\tau_p} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{dt}{\tau_q} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{dt}{\tau_r} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{3.60}$$

The output equations II are linear by nature so the do not have to be linearized. In contrast, the output equations of approach I are non-linear and it can be easily seen why this approach is computational more demanding. The partial derivatives of the output equations at the actual state estimate are given by:

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}}\right|_{\hat{x}_k} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & H44 & H45 & H46 & H47 \\ 0 & 0 & 0 & H54 & H55 & H56 & H57 \\ 0 & 0 & 0 & H64 & H65 & H66 & H67 \\ 0 & 0 & 0 & H74 & H75 & H76 & H77 \\ 0 & 0 & 0 & H84 & H85 & H86 & H87 \\ 0 & 0 & 0 & H94 & H95 & H96 & H97 \end{bmatrix} , \tag{3.61}$$

where

$$
\begin{aligned}
H44 &= 2wg_x + 2zg_y - 2yg_z & H45 &= 2xg_x + 2yg_y + 2zg_z \\
H46 &= -2yg_x + 2xg_y - 2wg_z & H47 &= -2zg_x + 2wg_y + 2xg_z \\
H54 &= -2zg_x + 2wg_y + 2xg_z & H55 &= 2yg_x - 2xg_y + 2wg_z \\
H56 &= 2xg_x + 2yg_y + 2zg_z & H57 &= -2wg_x - 2zg_y + 2yg_z \\
H64 &= 2yg_x - 2xg_y + 2wg_z & H65 &= 2zg_x - 2wg_y - 2xg_z \\
H66 &= 2wg_x + 2zg_y - 2yg_z & H67 &= 2xg_x + 2yg_y + 2zg_z \\
H74 &= 2wB_x + 2zB_y - 2yB_z & H75 &= 2xB_x + 2yB_y + 2zB_z \\
H76 &= -2yB_x + 2xB_y - 2wB_z & H77 &= -2zB_x + 2wB_y + 2xB_z \\
H84 &= -2zB_x + 2wB_y + 2xB_z & H85 &= 2yB_x - 2xB_y + 2wB_z \\
H86 &= 2xB_x + 2yB_y + 2zB_z & H87 &= -2wB_x - 2zB_y + 2yB_z \\
H94 &= 2yB_x - 2xB_y + 2wB_z & H95 &= 2zB_x - 2wB_y - 2xB_z \\
H96 &= 2wB_x + 2zB_y - 2yB_z & H97 &= 2xB_x + 2yB_y + 2zB_z \ .
\end{aligned}
$$

After linearizing the process model the Extended Kalman Filter can be implemented by simply programming the filter equations from Section 2.1.3 in a computer or microcontroller.

3.6.4. Implementation considerations

Filter initialization

The Kalman Filter and all its derivatives are recursive filters, requiring an initialization. As stated in previous chapters, this initialization should be done according to the expected values of the state and covariance. For the present work, the system is assumed to be at rest and aligned with the NED coordinate system. Thus, the initial state vector is given by:

$$\hat{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T \ . \tag{3.62}$$

The covariance matrix is commonly initialized with an identity matrix multiplied by a scalar:

$$\mathbf{P} = a\mathbf{I}_{7\times 7} \ , \tag{3.63}$$

where the scalar is chosen to $a = 1$. A bad initial guess can cause stability problems due to the linearization and discretization. However, this was not observed through all tests and simulations.

Remarks about system and measurement noise processes

The Kalman Filter is the optimal filter minimizing the estimation error variance. This requires, that all noise processes in the system equations and output equations are white, uncorrelated and zero-mean. Thus, to achieve the best performance of the filter, the statistical properties

have to be investigated and determined.
Due to the limited time for the present work these investigations are not performed. For the sake of simplicity the noise elements in the process equations are assumed to be Gaussian, independent and numerical values are determined iteratively. Initial values are taken from [Mar00]. This work also includes investigations about the time constants of the human motion model.

Sequential Kalman Filter
Considering the Kalman Filter equations in Section 2.1.3, it can be seen that a matrix inversion is included. The system model, designed in the last section contains 7 states, which requires a matrix inversion of a 7x7 in the Kalman Filter equations. This is computationally extremely demanding, so a sequential form of the Kalman Filter is considered. As long as the measurement noise covariance matrix is either diagonal or constant, the measurement update equation can be separated and each measurement processed sequentially. Then, matrix inversion of a 7x7 matrix reduces to 7 scalar divisions plus a couple of additions and multiplications. The measurement-update equations of the sequential Kalman Filter are taken from [Sim06]:

1. Initialize the a posteriori estimate and covariance as

$$\hat{\mathbf{x}}_{0k}^{+} = \hat{\mathbf{x}}_{k}^{-}$$
$$\mathbf{P}_{0k}^{+} = \mathbf{P}_{k}^{-} \ .$$

   These are the a posteriori estimate and covariance at time k after zero measurements have been processed; that is, they are equal to the a priori estimate and covariance.

2. For i=1,...,r (where r is the number of measurements), perform the following:

$$\mathbf{K}_{ik} = \frac{\mathbf{P}_{i-1,k}^{+}\mathbf{H}_{ik}^{T}}{\mathbf{H}_{ik}\mathbf{P}_{i-1,k}^{+}\mathbf{H}_{ik}^{T} + \mathbf{R}_{ik}}$$
$$\hat{\mathbf{x}}_{ik}^{+} = \hat{\mathbf{x}}_{i-1,k}^{+} + \mathbf{K}_{ik}(y_{ik} - \mathbf{H}_{ik}\hat{\mathbf{x}}_{i-1,k}^{+})$$
$$\mathbf{P}_{ik}^{+} = (\mathbf{I} - \mathbf{K}_{ik}\mathbf{H}_{ik})\mathbf{P}_{i-1,k}^{+}(\mathbf{I} - \mathbf{K}_{ik}\mathbf{H}_{ik})^{T} + \mathbf{K}_{ik}\mathbf{R}_{ik}\mathbf{K}_{ik}^{T} \ .$$

3. Assign the a posteriori estimate and covariance as

$$\hat{\mathbf{x}}_{k}^{+} = \hat{\mathbf{x}}_{rk}^{+}$$
$$\mathbf{P}_{k}^{+} = \mathbf{P}_{rk}^{+} \ .$$

4. Model implementation

4.1. Introduction

The following sections describe the verification of the derived algorithm using real sensor data and present its microcontroller implementation. In Section 4.2 it is shown how the algorithm was programmed in MATLAB and verified for different number representations. This includes a description of the experimental setups and sensor calibration procedures. Subsequently, Section 4.3 deals with the implementation of the filter equations in a microcontroller. The algorithms were investigated regarding their execution time and accuracy of the AHRS was determined by final tests through measurements.

4.2. Implementation and verification with MATLAB

4.2.1. Hardware setup

In order to calibrate sensors and to investigate the derived Kalman Filter, two experimental setups were used, which are briefly described in the following.
A data acquisition setup was created for sensor calibration and off-line testing of the attitude estimation algorithm. Analogue sensor outputs were read by a A/D-converter card, which sent the acquired values to a PC in specified time periods. The PC was equipped with MATLAB, for further processing of the data. This setup is illustrated in Figure 4.1.

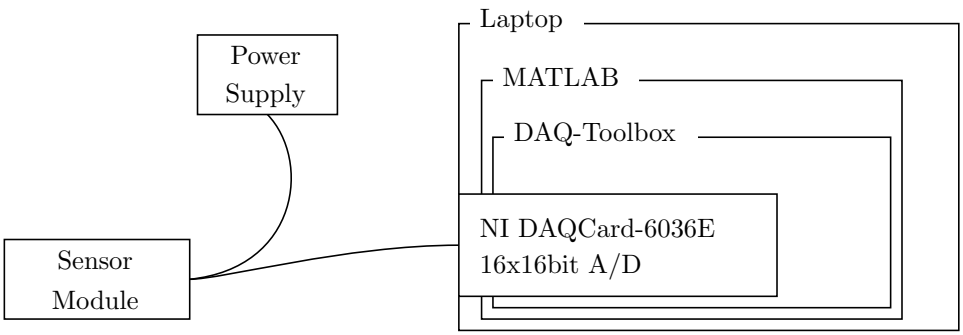

Fig. 4.1.: Experimental setup for data acquisition

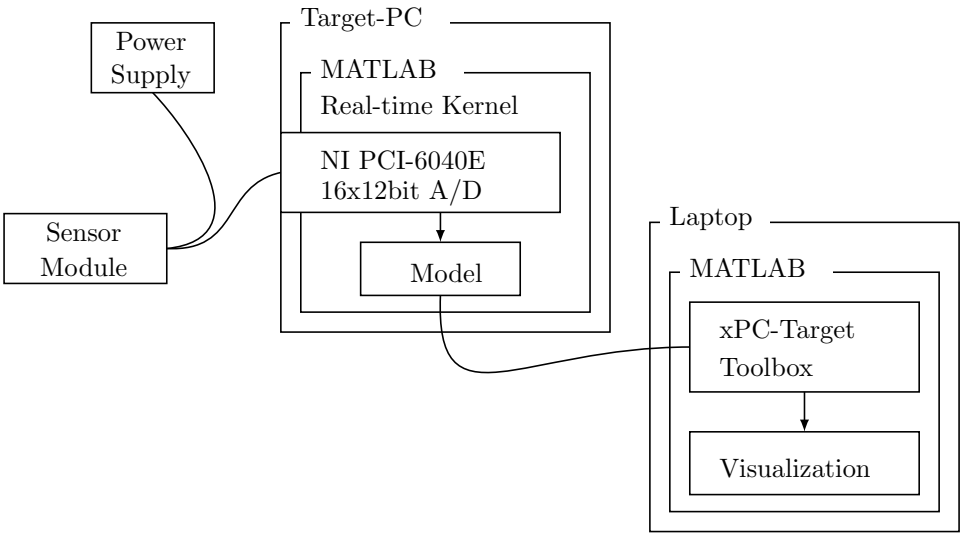

Fig. 4.2.: Experimental setup with xPC-Target

The second setup consisted of a xPC-Target environment depicted in Figure 4.2. It is a real-time, Hardware-In-the-Loop (HIL) environment used to adjust and evaluate the algorithm. The Kalman filter algorithm was computed in real-time on a Target-PC, which obtained the analogue sensor data through a data acquisition card. In specified time periods, the Target-PC sent the filter output to a PC, where MATLAB visualized the attitude estimate.

4.2.2. Sensor calibration

The accuracy of any attitude estimation algorithm highly depends on the quality of the sensor measurements. For this reason sensor bias and scaling errors have to be determined through a sensor calibration. A high precision, three axis tilt table was not available, so calibration was performed by alternative methods, briefly described in the following. For the sake of simplicity, the sensor module axes are denoted as body axes to distinguish them from the incident sensor axes.

Accelerometer calibration

The accelerometers were calibrated with a test bench described in Section 4.3.1. The test bench consisted of three rods, connected through rotational potentiometers, which provided the angles of each rod pair. The sensor module was attached to the test bench and accelerometer outputs were recorded for several orientations. The offset was determined by reading off the maximal and minimal sensor output for a full rotation around a body axis and calculating the arithmetic mean. The scaling and misalignment matrix could be found by computing the least squares solution, that relates unbiased sensor outputs to the gravity vector in body frame obtained by the incident orientation. This is illustrated for sensor measurements $\vec{a}_i^B$ and related gravity vectors in body frame $\vec{g}_i^B$, which are related by the scaling and misalignment matrix $\mathbf{M}$ according

to

$$\vec{g}_i^B = \mathbf{M}\vec{a}_i^B \ . \tag{4.1}$$

For at least three measurements ($i \geq 3$) the solution is given by

$$\mathbf{M} = \begin{bmatrix} \vec{g}_1^B & \vec{g}_2^B & \vec{g}_3^B & ... \end{bmatrix} \begin{bmatrix} \vec{a}_1^B & \vec{a}_2^B & \vec{a}_3^B & ... \end{bmatrix}^{-1} \ . \tag{4.2}$$

Gyroscope calibration

The gyroscopes were calibrated by a single axis DC-motor with a position and velocity controller. The sensor module was attached to the motor shaft and sensor outputs were recorded, while the shaft rotated with several angular velocities. Initial gyroscope bias factors were determined by observing the sensor output, when the sensor module has been at rest. This had to be repeated for each body axis, which required to dismount the sensor module from the motor shaft and attach it again to calibrate the next axis.

Magnetometer calibration

Magnetometer calibration was a more difficult task compared to gyroscope and accelerometer calibration. This was due to additional calibration parameters (hard and soft iron errors) and the magnetic earth field vector was not known initially. Additionally, the magnetic earth field vector is not constant in direction and strength. If the global sensor position is known, it is possible to calculate a theoretical approximation of the vector with latitude and longitude information. Anyhow, local magnetic field variations are common and also change over time.

Several approaches exist in literature to calibrate magnetometers with and without external magnetic field information. A brief summary can be found in [VES+08]. For the present work an intuitive and non-iterative calibration technique proposed by J.Merayo et al. in [MBP+00] was utilized. Magnetometer calibration is shown to be analogous to parameter identification of an ellipsoid. It is a geometric approach, forming a linear parameter model through a non-linear substitution of measurement values.

Measurements of a rotating, error-free 3-axis magnetometer in a constant external magnetic field lie on a sphere centered around (0,0,0), if the vector components for each measurement are plotted in a Cartesian coordinate system. Plotting vector measurements of a real, rotating magnetometer result in an ellipsoid, not centered around the origin. Thus, the combined effect of bias, scaling errors, misalignments, hard and soft iron errors cause a distortion from a sphere to a biased ellipsoid. J.Merayo's approach aims to compute a simple bias and scaling model from ellipsoid parameters, identified from magnetometer measurements, which is further explained in Appendix B.

Conclusions

Due to the lack of a three axis, precision tilt table, calibration was performed with a test bench with limited accuracy and a single axis DC-motor, which required to mount the sensor module

several times in order to calibrate all three axis. This came along with inevitable alignment errors introduced by dismounting and re-mounting. Extensive investigations of linearity and time-varying properties have not been performed due to the limited time of the present work.
Calibration of the magnetometer was performed by rotating the module by hand and finding bias and scaling values using Merayo's approach. Measurements with the calibrated magnetometer showed, that the local earth magnetic field at the measurement location was almost aligned with the gravity vector. In global frame, the normalized gravity vector is

$$\vec{g}^N = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{4.3}$$

and the normalized earth magnetic field is

$$\vec{B}^N = \begin{bmatrix} -0.13 & 0.17 & 0.98 \end{bmatrix}^T \ . \tag{4.4}$$

The angle between these vectors is the arc-cosine of the dot product of $\vec{g}^N$ and $\vec{B}^N$, which equals 11.5°. Thus, only a small component of the magnetic vector contained additional information to the gravity vector. Disturbing magnetic fields in combination with the uncertainties introduced by the calibration process made it impossible to use magnetic field measurements with the algorithm.

4.2.3. Model implementation

The EKF equations derived in the previous chapter were programmed in MATLAB and tested with the experimental setups described in Section 4.2.1. Figure 4.3 shows the principle program structure. The program starts by initializing hardware components, followed by a procedure to determine gyroscope biases. This is necessary due to bias variability from turn-on to turn-on. The biases are estimated by averaging the gyroscope outputs for 0.5 s requiring that the sensor module is at rest. Afterwards, the filter initialization assigns initial values for the states and the covariance matrix. Subsequently the Kalman Filter predicts and corrects the states and covariance matrix periodically. The state correction is performed with gyroscope sensor outputs and a computed measurement quaternion provided by the TRIAD algorithm from accelerometer and magnetometer measurements. The implemented Kalman Filter algorithm in MATLAB is shown in Appendix C.
In order to verify the general filter setup, the algorithm was implemented with double precision, which is the standard setting in MATLAB. Double precision stands for a 64 bit floating-point representation of all variables and mathematical operations are performed with this precision. In order to reduce the computational demands of the algorithm, a second implementation was created with a single or 32 bit floating-point representation.
The designated microcontroller has no FPU, thus floating point operations have to be emulated by software, which is computationally very demanding. For this reason, a third implementation using fixed-point arithmetics was programmed in C. Programming in C was necessary since a

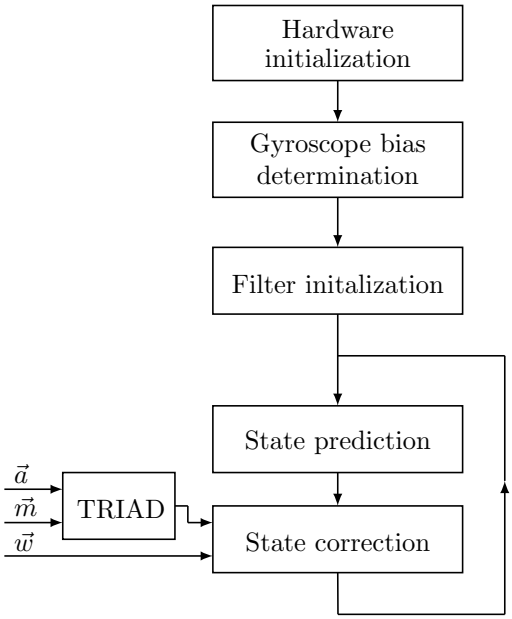

Fig. 4.3.: Principle program structure of the Kalman Filter

standard MATLAB version does not support fixed-point numbers and arithmetics. The C-code was compiled to a Mex-file, which provides a function interface for C-code in MATLAB. Thus, three different implementations were programmed and investigated:

- Implementation with double precision
- Implementation with single precision
- Implementation with fixed-point arithmetics

Sensor calibration showed, that the earth magnetic field vector was almost aligned with the gravity vector at the location, where the measurements were taken. Due to the limited calibration accuracy and the small difference of the vectors, it was not possible to use the magnetometer to stabilize the Yaw axis. For this reason, subsequent investigations were performed with artificially generated magnetometer values, which enabled the algorithm to be computable, without providing any information for Yaw stabilization. This was done by defining the earth magnetic field as

$$\vec{B}^N = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \tag{4.5}$$

and rotate it with the current attitude estimate to yield artificially generated magnetic field measurements in body frame according to Equation (3.31).

4.2.4. Tests and Results

Initially, the general functionality of the algorithm was verified with the double precision algorithm in the xPC-Target environment. The model sample rate was chosen to 1kHz and the

attitude estimates have been visualized on a PC with Microsoft Windows operating system. The animated visualization based on the attitude estimates followed the same rotations as the real sensor module. A slight delay was observable, which is partly due to the algorithm and to the visualization on the PC. Figure 4.4 shows a picture of the real and animated visualization.

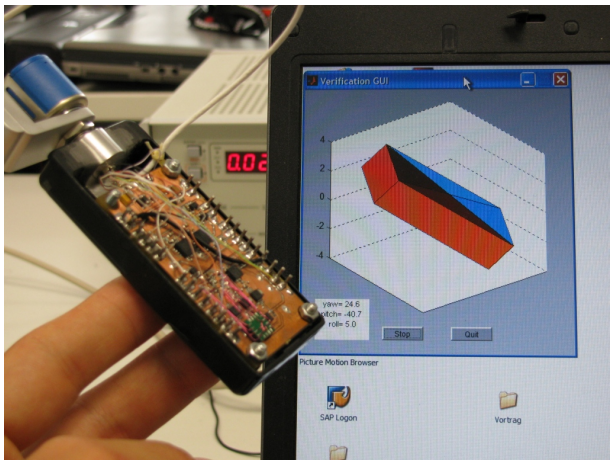

Fig. 4.4.: Verification with real and visualized sensor module

This process of subjective verification was repeated for decreasing sample rates and showed good performance down to 100Hz. Smaller sample rates violated approximations throughout the model design and limited the ability of the EKF to track high dynamical motions.
The same tests were repeated for single floating-point precision and several fixed-point implementations. In order to compare accuracy of the different implementations, test data were acquired and attitude estimates produced by the algorithms were compared off-line. Figure 4.5 shows the quaternion components w,x,y and z for a rotation around body x-axis from 0 to $\frac{\pi}{2}$ and back to 0 rad provided by the implemented number representations.
Double and single precision floating-point implementations produced the same output, whereas the fixed-point version showed slight deviations. The results of the fixed-point algorithm depicted in 4.5 are based on 32 bit variables with 27 bits fraction length.
In order to investigate the influence of fraction length on accuracy, tests were repeated for a different number of fraction bits. The difference of quaternion components produced by the fixed-point versions and the floating-point realization are presented in Figure 4.6. A fraction length of 27 bit had highest accuracy, while smaller fraction lengths diverged with time, thus accuracy decreased. Tests with 32 bit variables and fraction lengths longer than 27 bit failed due to overflows. Even though overflows have not been observed for a fraction length of 27 bit, this should be further investigated to make sure that no overflows are possible for any case of operation.
A 64 bit fixed-point implementation was realized to investigate fraction lengths longer than 27 bit. Mathematical operations were emulated in software, since the microcontroller and MATLAB have a 32 bit structure. Tests showed that accuracy did not increase significantly. Implementa-

tion of 64 bit variables was not further followed, since the accompanying software emulation of mathematical operations are computationally demanding, comparable to software floating-point implementations.

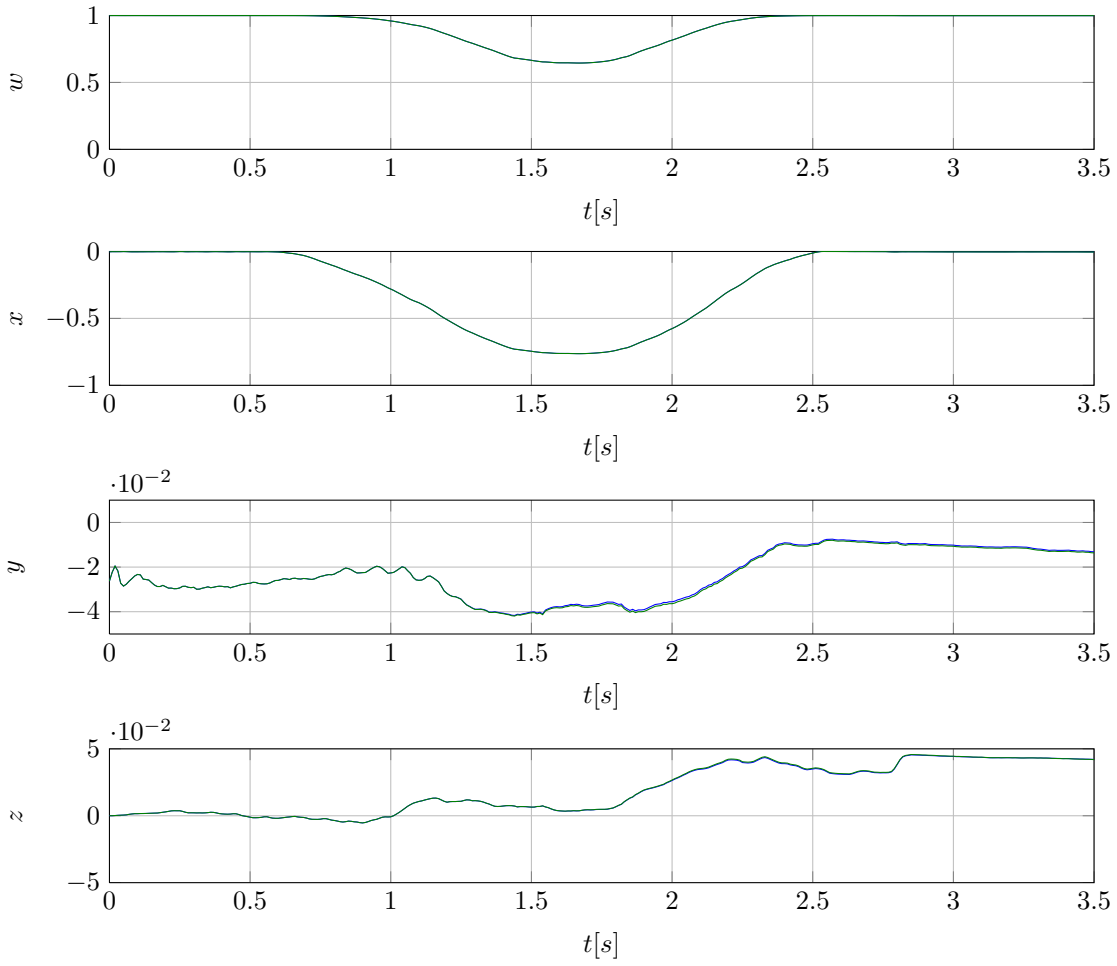

Fig. 4.5.: Precision of floating-point (green) and fixed-point (blue) implementations; the quaternion components w,x,y and z are shown for a rotation around body x-axis from 0 to $\frac{\pi}{2}$ and back to 0 rad.

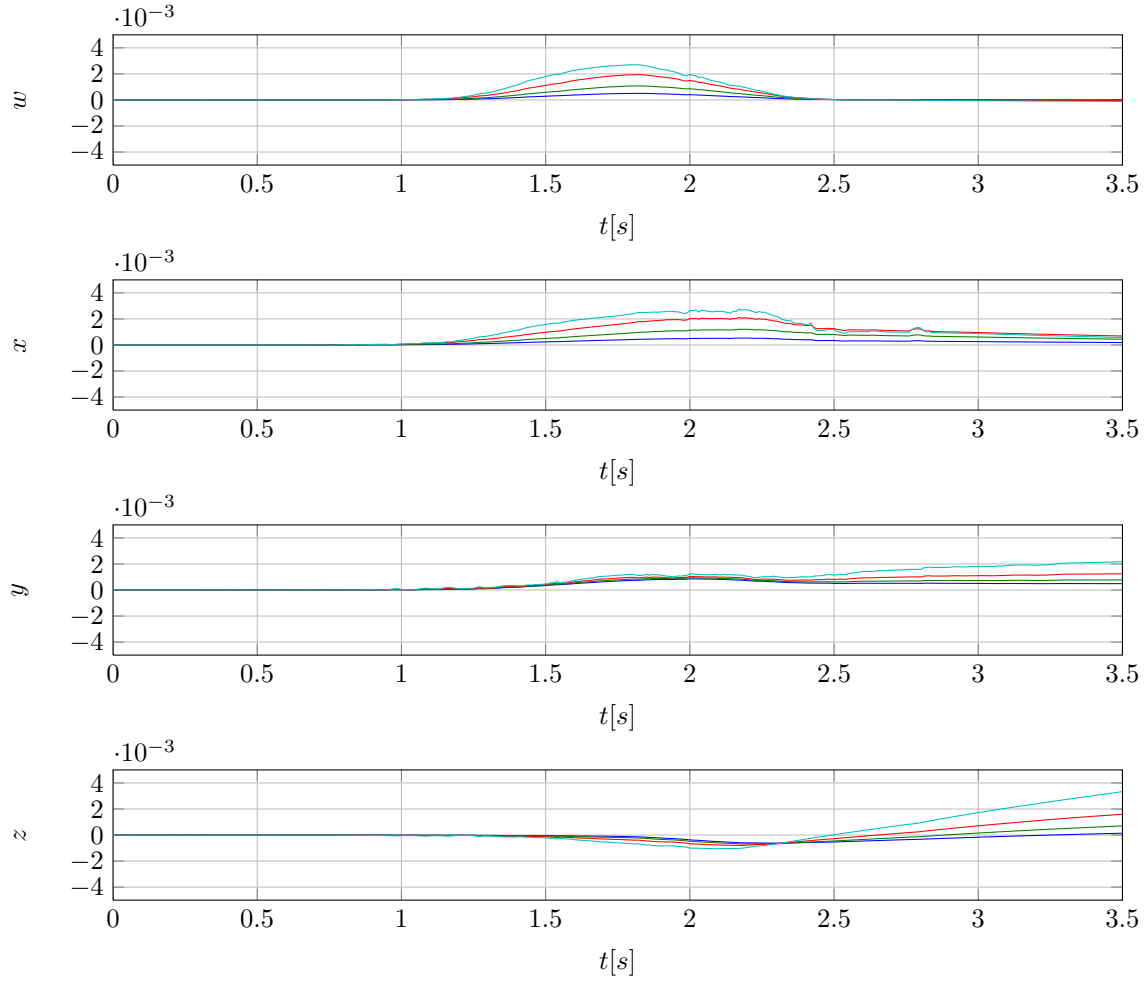

Fig. 4.6.: Deviations of quaternion components w,x,y and z for fixed-point implementations with fraction lengths of 27 bits (blue), 26 bits (green), 25 bits (red) and 24 bits (cyan) to the floating-point results are shown for a rotation around body x-axis from 0 to $\frac{\pi}{2}$ and back to 0 rad.

4.3. Microcontroller implementation

4.3.1. Hardware setup

The major challenge of implementing a Kalman Filter in a microcontroller is the limited computational resources to guarantee a sufficient sample rate of the algorithm. It is possible to attack this problem by using a special Digital Signal Processor (DSP) with a high clock frequency or a microcontroller equipped with a FPU. Both approaches are connected to a relative high power consumption, which is not desired in the present work. Thus, the aim is to implement the designed filter on a microcontroller without FPU.
A 32-bit microcontroller STM32F103 by ST with an ARM7-CortexM3 core was chosen for implementation. The STM32F103 has up to 72MHz clockrate, 16 12bit A/D-Converters and 128kB Flash memory. The IAR workbench was used as Integrated Development Environment (IDE) for programming and debugging.
The hardware setup was given as follows. An evaluation board equipped with the microcontroller was connected to the sensor board by flexible and thin cables. Generally, it is possible to mount the controller on the sensor board, which is planned for future work. In order to visualize the estimated attitude, the microcontroller sent the filter output to a PC via RS-232. The complete setup is illustrated in Figure 4.7.

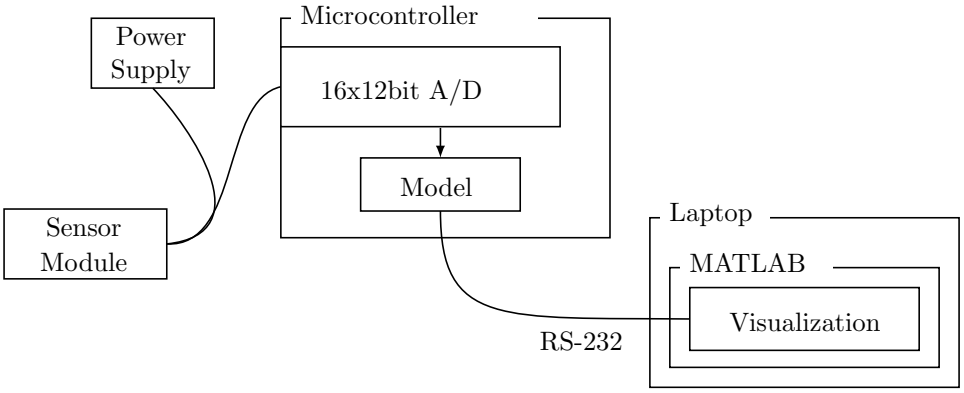

Fig. 4.7.: Experimental setup for microcontroller implementation

In order to verify the accuracy of the designed filter, a test bench was designed, which is depicted in Figure 4.8. It consisted of three rotational potentiometers connected through rods, similar to a three axis robot arm. The sensor module was attached to the end of the arm and moved and rotated by hand. The "real" orientation of the robot arm was obtained by reading off the potentiometer values, providing attitude in Euler angles. Recording the potentiometer values and algorithm outputs with MATLAB in parallel, enabled to determine accuracy of the algorithm by comparing real and estimated orientation. Data recording happened at a sample rate of 10 Hz. A higher sample rate was not realizable due to limited speed of MATLAB interfaces with the serial port and the A/D-converter card.

The test bench lacked high precision bearings and potentiometers, which introduced measurement uncertainties due to slackness and limited angle resolution. Thus, measurement accuracy was estimated to $\pm 1.5°$ for each Euler angle axis.

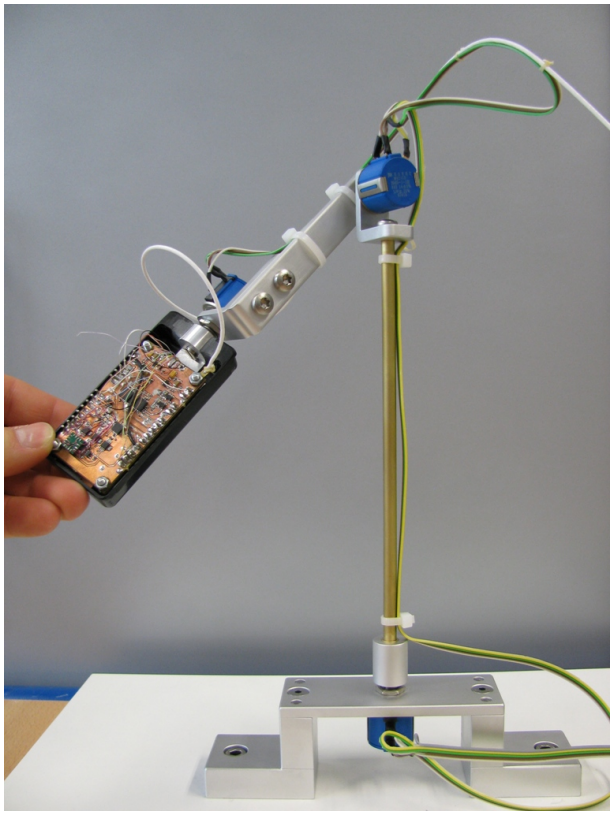

Fig. 4.8.: Test bench for accuracy determination

4.3.2. Model implementation

In order to investigate the derived Kalman Filter equations on the designated microcontroller, they were required in machine code. The IAR-Compiler was used to generate machine code from C-code versions of the algorithm for each number representation. MATLAB was used to auto-generate C-code from the double and single precision floating-point algorithms. The fixed-point algorithm was programmed in C from the beginning. Additionally, the single precision implementation was programmed "by hand", to check the performance of the auto-generated code. Tests of the microcontroller implementations were performed to investigate achievable filter update rates and estimation accuracy.

4.3.3. Tests and Results

In order to investigate achievable update rates, the number of required clock cycles for a single filter iteration were recorded for each number representation. Table 4.1 shows the averaged

results from this test. The listed update frequencies of the filter implementations consider a clock frequency of 72 MHz of the microcontroller.

Algorithm	Clock cycles	Max. update frequency
Double precision	1330000	54.1 Hz
Single precision	1004000	71.7 Hz
Single precision II	753000	95.6 Hz
Fixed-point	310000	232.3 Hz

Table 4.1.: Performance of filter algorithms in C-code

According to Section 4.2.4 a filter update rate of at least 100 Hz was required. Thus, only the fixed-point version achieved a sufficient update rate and was applicable for implementation in the designated microcontroller. Therefore, subsequent investigations about estimation accuracy were performed for the fixed-point implementation exclusively.
In order to investigate estimation accuracy, the test bench described in Section 4.2.1 was used. The filter update rate was chosen to 200 Hz, controlled by a timer interrupt of the microcontroller. The estimation errors were formed by the difference of estimated and measured orientation. For a better imagination the errors were transformed from quaternion attitude estimates to Euler angles.
The static accuracy was determined by observing the estimation error for several orientations while the sensor module was at rest. The static accuracy of the estimation algorithm lied within the measurement accuracy of the test bench of $\pm 1.5°$.
Accuracy for dynamic motions was determined by observing the estimation errors during dynamic motions and rotations. The measurements were performed "by hand", thus explicit angular rates and angular accelerations could not be set in advance. During low rotational accelerations the error lied withing the measurement accuracy, but increased to a maximum of $\pm 5°$ in times of high velocity changes. The numerical value of dynamic accuracy is the maximum error observed for the performed test scenarios. Two test scenarios are briefly described the following. The first setup contained a rotation around body y-axis, with a following rotation around body x-axis. Figure 4.9 shows the real and estimated attitude values for this setup. The second setup rotated the sensor module around body y- and x-axis in parallel. The result is shown in Figure 4.10. Static and dynamic accuracy are summarized in Table 4.2.

Euler angle	Static accuracy	Dynamic accuracy
Roll	$\pm 1.5°$	$\pm 5°$
Pitch	$\pm 1.5°$	$\pm 5°$
Yaw	not determinable	not determinable

Table 4.2.: Static and dynamic accuracy of the AHRS

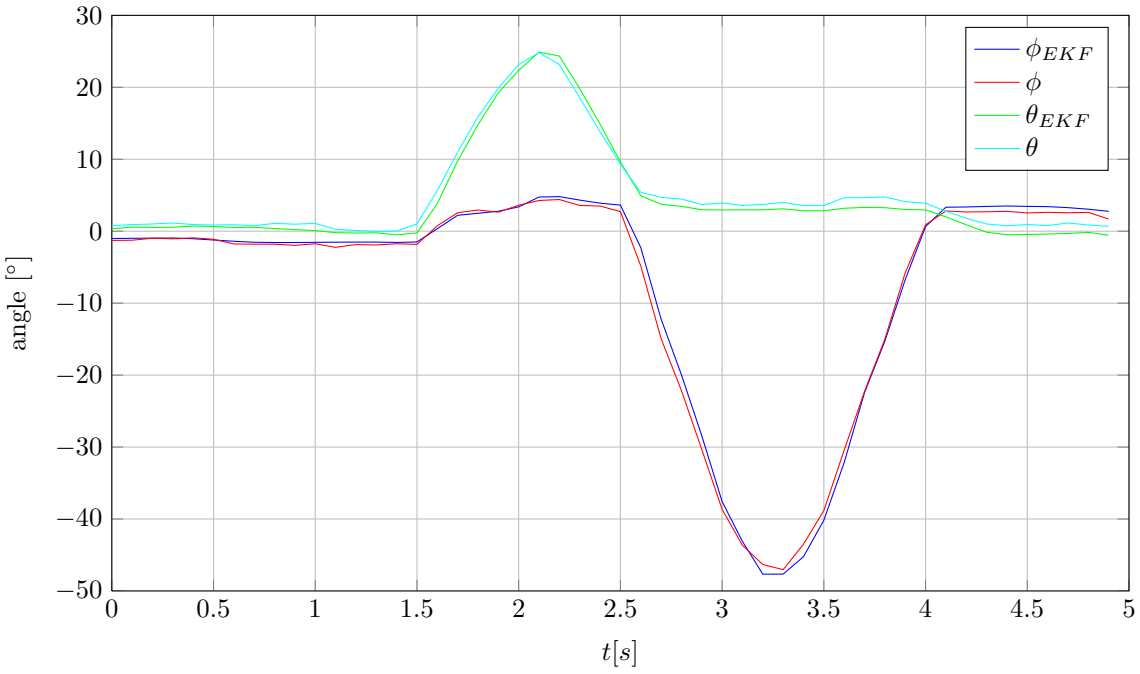

Fig. 4.9.: Dynamic accuracy of Roll and Pitch angle provided by EKF (ϕ_{EKF},θ_{EKF}) and test bench (ϕ,θ) for a rotation about sensor y-axis with a following rotation about x-axis

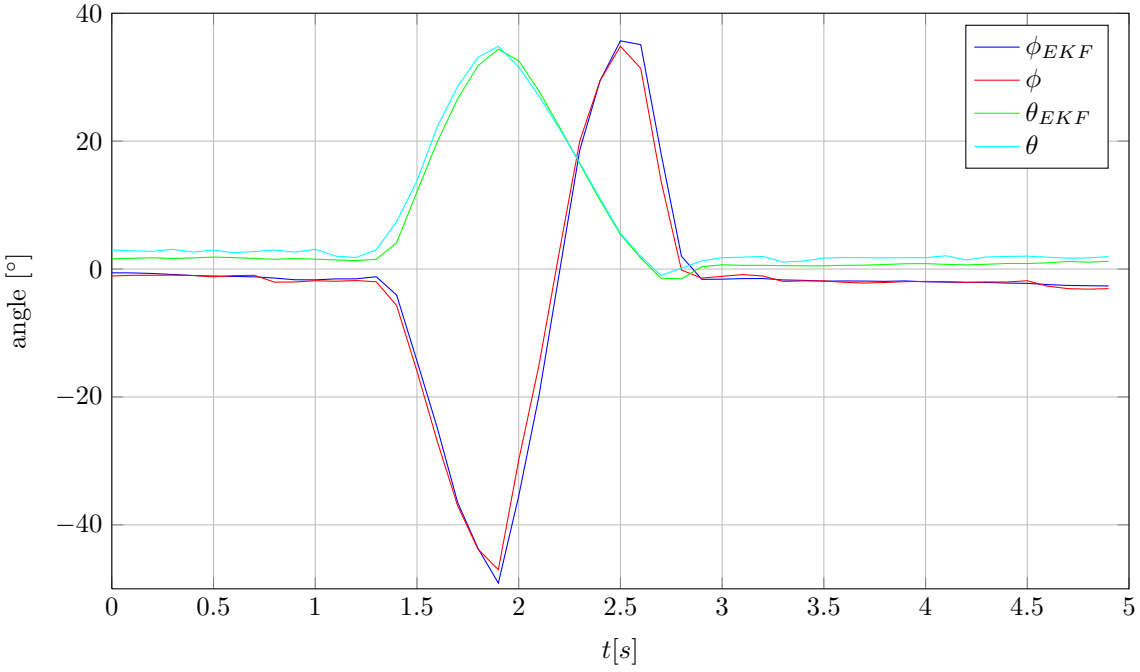

Fig. 4.10.: Dynamic accuracy of Roll and Pitch angle provided by EKF (ϕ_{EKF},θ_{EKF}) and test bench (ϕ,θ) for a parallel rotation about y- and x-axis

5. Results and Conclusions

This thesis presents the design and implementation of an attitude estimation system based on inertial and magnetic field measurements. An Extended Kalman Filter was designed to merge the sensor data with a quaternion-based system model. The algorithm was implemented on a microcontroller and its static accuracy for Roll and Pitch axes was determined to $\pm 1.5°$ and dynamic accuracy to $\pm 5°$. Due to an unfortunate direction of the earth magnetic field at the test location, magnetometer measurements turned out to be unreliable to stabilize attitude estimates about Yaw. Current products of Otto Bock Healthcare GmbH do not require absolute heading information, thus the designed system still provides a well founded basis for future work.

5.1. Mathematical modeling

The mathematical modeling included an investigation of attitude estimation approaches with inertial and magnetic field measurements. An Extended Kalman Filter setup was derived, which combines a system model with sensor measurements to take advantage of the complementary sensor errors. The system model is based on a quaternion attitude representation and derived from rigid body kinematics. The algorithm is computationally efficient and capable of tracking attitude in any orientation without singularities. The derived filter is similar to approaches proposed by Yun et al. in [YBM08] and Adiprawita et al. in [AAS07].

System model

Several alternative approaches exist to design a system model. A common approach is to include sensor errors as state variables, which is especially advantageous for tracking the gyroscope bias. This is connected to a higher computational load of the algorithm, due to the higher dimension of the state observer. Another approach, which is commonly applied in inertial navigation systems is to estimate only the sensor errors with a Kalman Filter. This setup is depicted in Figure 5.1 and follows a complementary filter methodology. According to [BH97], the filter problem becomes linear through the differencing operation, which "washes out" the nonlinear measurement relationships. The filter operates on the sensor errors and attitude is determined directly from corrected inertial output, which gives a high dynamic response. Since this complementary filter approach needs very accurate sensor models and usually relies on external aiding sources it was not considered for the present work.

In order to linearize the measurement equations of the designed Kalman Filter, the static attitude estimation approach TRIAD has been used. This is a suboptimal and deterministic approach.

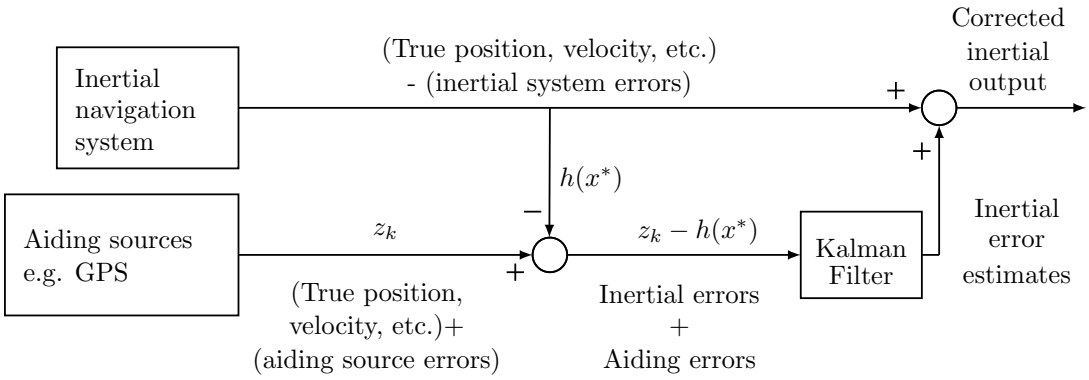

Fig. 5.1.: Complementary filter approach [BH97]

Alternatively the QUEST algorithm could be utilized, which eventually provides more accurate attitude estimates at the cost of a higher computational cost. Recently, Yun et al. proposed the factored quaternion algorithm (FQA), which is almost identical to TRIAD but produces quaternions estimates instead of a rotation matrix [YBM08].

Linearization and discretization

In order to avoid estimation errors introduced by linearization in the EKF equations, the UKF could be used. LaViola compared estimation accuracy and computational efficiency of EKF and UKF for a quaterion-based tracking application of human head and hands [LJ03]. Tests showed that accuracy for both filters is almost the same thus it can be concluded that the EKF is the preferred choice for the present application due to its lower computational cost.
Discretizing the system model was done by Euler's method. Alternatively, Tustin approximation or higher order approaches might be considered to reduce discretization errors. Additionally, a review of sensor filter cut off frequencies to avoid anti-aliasing is required. During the sensor board design, cut off frequencies for accelerometers, magnetometers and gyroscopes were chosen to 100 Hz and 145 Hz. Thus, the final sample rate for microcontroller implementation of 200 Hz violates Nyquist-Shannon's sampling theorem.

Sensor modeling

Accuracy of the designed algorithm highly depends on the sensor quality. Due to the limited time of this thesis an extensive investigation of sensors and accompanying errors was not performed. Especially time-varying error characteristics, saturation and temperature effects have to be considered and investigated.

Disturbances

An important topic, which was not investigated within this thesis is the response of the filter to disturbances in form of external magnetic fields and linear accelerations. Generally, it would

be advantageous to adapt the measurement noise properties. This would require to detect these disturbances and to find appropriate rules for adaption. According to [DWR06], popular adaptive methods can be roughly classified into covariance scaling, multi-model adaptive estimation and adaptive stochastic modeling. These methods basically investigate the estimate residuals and scale the noise properties accordingly. [RH04] presents an approach to estimate attitude with gyroscopes and accelerometers using two Kalman-Filters in parallel. In times of low acceleration, only gravity is measured by the accelerometer and the filter operates in "normal" mode. If high accelerations are detected, a switching to another filter setup takes place, which basically disregards accelerometer measurements.

Numerical values for the Extended Kalman Filter

Numerical values for noise processes have been determined iteratively, instead of analyzing them empirically. Especially the implementation with fixed-point arithmetics required an intensive adaption of the system and measurement covariance matrices. Covariance values smaller than one caused the covariance of the related state to quickly converge to zero, due to the limited range of the fixed-point variables. The presented results in this thesis were achieved by setting the system covariance matrix to

$$\mathbf{Q} = \begin{bmatrix} 1.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.1}$$

and the measurement covariance matrix to

$$\mathbf{R} = \begin{bmatrix} 1.11 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.11 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.11 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.01 \end{bmatrix} . \tag{5.2}$$

Absolute values of covariance matrices can be chosen arbitrarily, since only the ratio of system to measurement noise is relevant [Swa00]. However, according to [Sim06], non-empirical determination of numerical values for noise processes in the Kalman Filter happens at a risk of sacrificing the filters optimality properties and can lead to stability problems.

Magnetic field measurements

An elementary weakness of the designed algorithm occurred during tests with real sensor measurements. The earth magnetic field vector, which basically provides information about the magnetic north direction turned out to be almost aligned with the gravity vector at the test location. In combination with the low precision calibration procedure, the magnetometer did not generate reliable measurements. It might be advantageous to use field variations instead of absolute values.

Furthermore, the earth magnetic field is much weaker than magnetic fields generated from electronic devices and permanent magnets in the sensor environment. The effects of disturbing magnetic fields might be reduced by considering a magnetic disturbance model presented in [RBV07].

5.2. Model implementation

The software development included the process of verifying the algorithm and implementing it on a low-power microcontroller without FPU. Initially, the algorithm was programmed in MATLAB for fixed-point and floating-point number representations. Accuracy of the different number representations were compared. A fixed-point 32 bit variable with 27 bit fraction length had comparable accuracy as the floating-point versions. The general algorithm functionality was verified in a real-time, hardware-in-the-loop environment for filter update rates larger than 100 Hz. The implementation on a 32bit microcontroller with 72 MHz clock frequency showed that a sufficient update rate is only achievable for a fixed-point number representation. Accuracy of the final algorithm implementation was determined for a filter update rate of 200 Hz.

Accuracy

The achieved accuracy of the designed system can be expected to improve, if sensor calibration and accuracy tests are performed with a high precision 3-axis tilt table. Alternative calibration procedures allowed only a limited accuracy of sensor scale and misalignment determination. The designed test bench for accuracy determination was sufficient to verify the general performance of the algorithm, but lacked high precision bearings and angle sensors. Additionally, accuracy should be monitored over large time spans, to investigate the effect of time-varying gyroscope biases.

Robustness

In order to prove robustness of the designed system, stability against initial attitude errors and sensor disturbances should be investigated. Consequences of the limited range and precision of the fixed-point number representation have to be further inspected. It might be advantageous to assign an individual fraction length to each variable in the implemented algorithm. This requires to know the range of each variable in advance, which might be difficult to determine.

5.3. Future prospects

In order to improve the designed attitude estimation system, future work should consider empirical investigations to find optimal parameters for the Extended Kalman Filter. For the present work, numerical parameters of the filter were determined iteratively and adapted until the algorithm showed "good" results. Especially sensor parameters, noise properties of the TRIAD algorithm and properties of the human motion model should be investigated empirically to improve estimation accuracy.
The current implementation is not capable of providing reliable Yaw estimates, due to the earth magnetic field at the test location and an inaccurate magnetometer calibration. A high precision calibration and a magnetic field disturbance model might help to overcome these difficulties. Alternatively, a redesign of the algorithm could be considered, which uses magnetic field variations instead of absolute values.
The implemented algorithm is based on fixed-point arithmetics to achieve a sufficient update rate on the designated microcontroller. It might be advantageous to consider alternative microcontrollers with FPU or DSP. This would enable an implementation of the algorithm with floating-point arithmetic, which has higher precision and avoids problems due to overflows. Special low-power derivates might overcome the disadvantage of a higher power consumption.
Recently, the design of small, low-power and self-contained attitude estimation system gains much attention in research projects and product developments. Current advances in sensor miniaturization and accuracy permits the conclusion, that a fully integrated solution can be expected within the next five years, which contains sensors and signal processing on a single, low-cost chip.

Bibliography

[AAS07] Widyawardana Adiprawita, Adang Suwandi Ahmad, and Jaka Sembiring. Development of AHRS (Attitude and Heading Reference System) for autonomous UAV (Unmanned Aerial Vehicle). *Proceedings of the International Conference on Electrical Engineering and InformaticsInstitut Teknologi Bandung, Indonesia*, pages 714–717, June 17-19 2007.

[ARM96] ARM. Application note 33: Fixed point arithmetic on the ARM, Sep 1996. `http://infocenter.arm.com/help/topic/com.arm.doc.dai0033a/DAI0033A_fixedpoint_appsnote.pdf` (accessed on May 25th 2010).

[AW96] Karl J. Astrom and Bjorn Wittenmark. *Computer-Controlled Systems*. Prentice Hall, 3rd edition, 1996.

[Bak] Martin John Baker. Maths - conversion quaternion to matrix. `http://www.euclideanspace.com/maths/geometry/rotations/conversions/quaternionToMatrix/index.htm` (accessed on January 21st 2010).

[Ber07] Paul Berner. Orientation, rotation, velocity, and acceleration and the SRM, Jun 2007. `http://www.sedris.org/wg8home/Documents/WG80462.pdf` (accessed on May 25th 2010).

[BH97] Robert Grover Brown and Patrick Y. C. Hwang. *Introduction to random signals and applied kalman filtering.* John Wiley & Sons, 1997.

[BL08] Alain Barraud and Suzanne Lesecq. Magnetometer calibration, 2008. `http://www.mathworks.com/matlabcentral/fileexchange/23398-magnetometers-calibration` (accessed on April 5th 2010).

[Can08] Natural Resources Canada. Geomagnetism earth's magnetic field, Jan 2008. `http://gsc.nrcan.gc.ca/geomag/field/index_e.php` (accessed on June 8th 2010).

[Dum99] Ildeniz Duman. Design, implementation and testing of a real-time software system for a quaternion-based attitude estimation filter. Master's thesis, Naval Postgraduate School, 1999.

[DWR06] Weidong Ding, Jinling Wang, and Chris Rizos. Improving covariance based

adaptive estimation for GPS/INS integration. Technical report, School of Surveying and Spatial Information Systems, University of New South Wales, Sydney, Australia, 2006. `http://www.gmat.unsw.edu.au/snap/publications/ding_etal2006c.pdf` (accessed on May 31st 2010).

[Far08] Jay A. Farrell. *Aided Navigation; GPS with high rate sensors.* McGraw Hill, 2008.

[GEEPP00] Demoz Gebre-Egziabher, Gabriel H. Elkaim, J. D. Powell, and Bradford W. Parkinson. A gyro-free quaternion-based attitude determination system suitable for implementation using low cost sensors. In *In Proc. IEEE Position Location and Navigations Symposium (IEEE PLANS)*, pages 185–192, 2000.

[GWA01] Mohinder S. Grewal, Lawrence R. Weill, and Angus P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration.* Wiley, 2001.

[Hal03] Chris Hall. Attitude determination chapter 4, 2003. `http://www.aoe.vt.edu/~cdhall/courses/aoe4140/attde.pdf` (accessed on January 19th 2010).

[Hon06] Honeywell. Datasheet HMC1051/HMC1052/HMC1053, 2006. `http://www.sparkfun.com/datasheets/IC/HMC105X.pdf` (accessed on May 25th 2010).

[Kon09] Christopher Konvalin. Compensating for tilt, hard-iron, and soft-iron effects, Dec 2009. `http://www.sensorsmag.com/sensors/motion-velocity-displacement/compensating-tilt-hard-iron-and-soft-iron-effects-6475` (accessed on May 28th 2010).

[LJ03] Joseph J. LaViola and Jr. A comparison of unscented and extended kalman filtering for estimating quaternion motion, 2003. `http://www.cs.brown.edu/~jjl/pubs/laviola_acc2003.pdf` (accessed on March 19th 2010).

[Mar00] Joao L. Marins. An extended kalman filter for quaternion-based attitude estimation. Master's thesis, Naval Postgraduate School, 2000.

[Mar02] F. Landis Markley. Fast quaternion attitude estimation from two vector measurements, 2002. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.1830&rep=rep1&type=pdf` (accessed on January 20th 2010).

[MBP+00] J. M. G. Merayo, P. Brauer, F. Primdahl, J. R. Petersen, and O. V. Nielsen. Scalar calibration of vector magnetometers. *Meas. Sci. Technol.*, 11:120–132, 2000.

[Okl92] Vojin G. Oklobdzija. Computer arithmetic. Technical report, Electrical and Computer Engineering Department University of California Davis, 1992. `http://lapwww.epfl.ch/courses/comparith/Arithm-CRC.pdf` (accessed on May 25th 2010).

[RBV07] Daniel Roetenberg, Chris T. M. Baten, and Peter H. Veltink. Estimating body segment orientation by applying inertial and magnetic sensing near ferromagnetic materials. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15(3):469–471, 2007.

[RH04] Henrik Rehbinder and Xiaoming Hub. Drift-free attitude estimation for accelerated rigid bodies. *Automatica*, 40:653–659, 2004.

[Shu06] Malcolm D. Shuster. The generalized wahba problem. *The Journal of the Astronautical Sciences*, 54(2):245–259, 2006.

[Sim06] Dan Simon. *Optimal State Estimation*. Wiley-Interscience, 2006.

[ST08] ST. Datasheet LIS344ALH rev.3, Apr 2008. `http://www.st.com/stonline/products/families/sensors/motion_sensors/lis344alh.htm` (accessed on May 25th 2010).

[ST09a] ST. Datasheet LPR550AL rev. 2, Jun 2009. `http://www.st.com/stonline/products/literature/ds/15813/lpr550al.pdf` (accessed on May 25th 2010).

[ST09b] ST. Datasheet LY550ALH rev. 1, Jun 2009. `http://www.st.com/stonline/books/pdf/docs/15802.pdf` (accessed on May 25th 2010).

[Svo05] Tomas Svoboda. Least-squares solution of homogeneous equations, Dec 2005. `http://cmp.felk.cvut.cz/cmp/courses/XE33PVR/WS20072008/Lectures/Supporting/constrained_lsq.pdf` (accessed on April 2nd 2010).

[Swa00] David C. Swanson. *Signal processing for intelligent sensor systems*. Marcel Dekker, 2000.

[TP02] Chin-Woo Tan and Sungsu Park. Design and error analysis of accelerometer-based inertial navigation systems. Technical report, University of California, June 2002. `DesignandErrorAnalysisofAccelerometer-BasedInertialNavigationSystems` (accessed on May 25th 2010).

[VES+08] J.F. Vasconcelos, G. Elkaim, C. Silvestre, P. Oliveira, and B. Cardeira. A geometric approach to strapdown magnetometer calibration in sensor frame, 2008. `http://www.soe.ucsc.edu/~elkaim/Documents/ReimmanTAES08.pdf` (accessed on May 15th 2010).

[Vic01] Leandra Vicci. Quaternions and rotations in 3-space: The algebra and its geometric interpretation, 2001. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.6183&rep=rep1&type=pdf` (accessed on January 20th 2010).

[vR06] Harald von Rosenberg. Sensorfusion zur navigation eines fahrzeugs mit low-cost inertialsensorik. Master's thesis, University of Stuttgart, 2006.

[VvVB04] James M. Van Verth, Jim M. van Verth, and Lars M. Bishop. *Essential mathematics for games and interactive applications: a programmers guide*. Elsevier Inc., 2004.

[Wei03] Eric W. Weisstein. *CRC concise encyclopedia of mathematics*. Chapman & Hall, 2nd edition, 2003.

[Wer78] J. R. Wertz. *Spacecraft Attitude Determination and Control*. D. Reidel Publishing Company, 1978.

[Wha66] Grace Whaba. A least squares estimate of satellite attitude. *SIAM Review*, 8(3):384–386, July 1966.

[Woo07] Oliver J. Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, 2007. `http://www.navlab.net/Publications/Introduction_to_Inertial_Navigation.pdf` (accessed on January 14th 2010).

[WP98] Christopher R. Wren and Alex P. Pentland. Dynamic models of human motion. *IEEE Proceedings of FG*, pages 22–27, 1998.

[YB06] Xiaoping Yun and Eric R. Bachmann. Design, implementation, and experimental results of a quaternion-based kalman filter for human body motion tracking. *IEEE Transactions on Robotics*, 22(6):1216–1226, 2006.

[YBM08] Xiaoping Yun, Eric R. Bachmann, and Robert B. McGhee. A simplified quaternion-based algorithm for orientation estimation from earth gravity and magnetic field measurements. *IEEE Transaction on Instrumentation and Measurement*, 57(3):638–650, 2008.

A. Conversion from rotation matrix to quaternion representation

A matrix describing attitude or a rotation in three-dimensional space contains nine elements according to

$$\mathbf{T} = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix} .$$

A rotation about a unit quaternion $\mathbf{q}$ can be expressed in matrix form $\mathbf{M}(\mathbf{q})$, where

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & -2wz + 2xy & 2wy + 2xz \\ 2wz + 2xy & w^2 - x^2 + y^2 - z^2 & -2xw + 2yz \\ -2wy + 2xz & 2wx + 2yz & w^2 - x^2 - y^2 + z^2 \end{bmatrix} .$$

By comparing the elements of $\mathbf{T}$ and $\mathbf{M}(\mathbf{q})$ it is possible to compute the quaternion components w, x, y and z of $\mathbf{q}$ according to

$$\begin{aligned} w &= \pm 0.5\sqrt{1 + m_{00} + m_{11} + m_{22}} \\ x &= \frac{m_{21} - m_{12}}{4w} \\ y &= \frac{m_{02} - m_{20}}{4w} \\ z &= \frac{m_{10} - m_{01}}{4w} . \end{aligned}$$

In case w equals zero, which would give a division by zero, it is possible to compute x, y or z from the diagonal elements and compute the remaining quaternion components by off-diagonal elements. This is shown in [Bak].

B. Magnetometer calibration

In [MBP+00] Merayo et al. propose a magnetometer calibration method, which finds bias, scaling and misalignment errors based on magnetometer measurements taken in several orientations. Plotting the three-axes measurements in a Cartesian coordinate system yield an ellipsoid, whereas perfect measurements would result in a sphere centered around (0,0,0). The calibration is shown to be identical to parameter identification of the ellipsoid, given by

$$(\mathbf{v}-\mathbf{c})^T(\mathbf{U}^T\mathbf{U})(\mathbf{v}-\mathbf{c}) = 1 \ , \tag{B.1}$$

where $\mathbf{v}$ is a three-axes magnetometer measurement, $\mathbf{U}$ is a 3×3 upper triangular matrix and $\mathbf{c}$ is the ellipsoid center. After $\mathbf{c}$ and $\mathbf{U}$ have been found, a calibrated measurement $\mathbf{w}$ can be computed by

$$\mathbf{w} = \mathbf{U}(\mathbf{v}-\mathbf{c}) \ . \tag{B.2}$$

In order to find the required bias vector and the triangular matrix the ellipsoid equation is rewritten in a linearized parameter model form

$$\begin{bmatrix} x^2 & y^2 & z^2 & xy & xz & yz & x & y & z \end{bmatrix} \mathbf{p} = 1 \tag{B.3}$$

where $\mathbf{p}$ is a 9×1 parameter vector to estimate. Merayo et al. show that $\mathbf{U}$ can be found by Cholesky factorization of a positive definite matrix formed by components of $\mathbf{p}$:

$$\mathbf{A} = \mathbf{U}^T\mathbf{U} = \begin{bmatrix} p_1 & p_4/2 & p_5/2 \\ p_4/2 & p_2 & p_6/2 \\ p_5/2 & p_6/2 & p_3 \end{bmatrix} \ . \tag{B.4}$$

The ellipsoid center c can be found by

$$\mathbf{c} = \mathbf{A}^{-1} \begin{bmatrix} p_7 \\ p_8 \\ p_9 \end{bmatrix} \ . \tag{B.5}$$

Estimating the parameter vector $\mathbf{p}$ requires at least nine measurement in different sensor orientations. Then Equation (B.3) can be rewritten as

$$\mathbf{Dp} = \mathbf{1} \tag{B.6}$$

where $\mathbf{D}$ is a matrix containing N magnetometer measurements of the x-, y- and z- components according to

$$\mathbf{D} = \begin{bmatrix} x_1^2 & y_1^2 & z_1^2 & x_1y_1 & x_1z_1 & y_1z_1 & x_1 & y_1 & z_1 \\ x_2^2 & y_2^2 & z_2^2 & x_2y_2 & x_2z_2 & y_2z_2 & x_2 & y_2 & z_2 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x_N^2 & y_N^2 & z_N^2 & x_Ny_N & x_Nz_N & y_Nz_N & x_N & y_N & z_N \end{bmatrix}$$

The least-square solution to the estimation problem is given by

$$\mathbf{p} = [\mathbf{D}^T\mathbf{D}]^{-1}\mathbf{D1} \; . \tag{B.7}$$

In order to achieve accurate estimates, a high number of measurements should be considered, which increases the demand for computational power and precision to compute $[\mathbf{D}^T\mathbf{D}]^{-1}$. This can be avoided by reformulating Equation (B.6) according to

$$\mathbf{D'p'} = \mathbf{0} \tag{B.8}$$

where $\mathbf{D'}$ contains N magnetometer measurements, enhanced by a column of ones:

$$\mathbf{D'} = \begin{bmatrix} x_1^2 & y_1^2 & z_1^2 & x_1y_1 & x_1z_1 & y_1z_1 & x_1 & y_1 & z_1 & 1 \\ x_2^2 & y_2^2 & z_2^2 & x_2y_2 & x_2z_2 & y_2z_2 & x_2 & y_2 & z_2 & 1 \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ x_N^2 & y_N^2 & z_N^2 & x_Ny_N & x_Nz_N & y_Nz_N & x_N & y_N & z_N & 1 \end{bmatrix}$$

and $\mathbf{p'}$ is now a 10×1 parameter vector. The solution to this homogeneous equation set can be found computationally efficient by a singular value decomposition (SVD) approach, derived in [Svo05]. This method is also followed by Barraud and Lesecq, who published accompanying MATLAB-code in [BL08].

C. MATLAB code of EKF implementation

The following code presents the MATLAB implementation of the designed filter algorithm in standard double precision.

```
function [quat]= EKF(in_omega_x,in_omega_y,in_omega_z,in_acc_x,in_acc_y,in_acc_z)
% constant bias and scale factors
bias_acc_x=1.493;
bias_acc_y=1.5;
bias_acc_z=1.496;
bias_omega_x=1.206;
bias_omega_y=1.218;
bias_omega_z=1.224;
scale_acc_x=-5.2;
scale_acc_y=-5.05;
scale_acc_z=4.7;
scale_omega_x=8.95;
scale_omega_y=-9.45;
scale_omega_z=-8.8;

% scale and remove bias from gyros
omega_x=scale_omega_x*(in_omega_x-bias_omega_x);
omega_y=scale_omega_y*(in_omega_y-bias_omega_y);
omega_z=scale_omega_z*(in_omega_z-bias_omega_z);

% scale and remove bias from accelerometers
scale_acc=[scale_acc_x 0.15 0.1;...
        0.09 scale_acc_y -0.03;...
        0.04 0.06 scale_acc_z];
acc_out= (scale_acc*([in_acc_x;in_acc_y;in_acc_z]-[bias_acc_x;bias_acc_y;bias_acc_z]))';
acc_x=acc_out(1);
acc_y=acc_out(2);
acc_z=acc_out(3);

% normalize accelerometer measurements
norm_acc=1/sqrt(acc_x.^2+acc_y.^2+acc_z.^2);
acc_x=norm_acc*acc_x;
acc_y=norm_acc*acc_y;
acc_z=norm_acc*acc_z;

persistent x
persistent P
if isempty(x)|| isempty(P)
        x=[0 0 0 1 0 0 0]'; % state vector
        P=1*eye(7); % state covariance matrix
end

dt=0.01; % sample rate
tau=0.5; % constant to model angle random walk of human motion dynamics
```

```
45
46 % system covariance matrix
47 Q=[ 1.1 0   0   0   0   0   0;...
48     0   1.1 0   0   0   0   0;...
49     0   0   1.1 0   0   0   0;...
50     0   0   0   0   0   0   0;...
51     0   0   0   0   0   0   0;...
52     0   0   0   0   0   0   0;...
53     0   0   0   0   0   0   0];
54
55 % measurement covariance matrix
56 R=[ 1.01 0   0   0     0   0    0;...
57     0   1.01 0   0     0   0    0;...
58     0   0   1.01 0     0   0    0;...
59     0   0   0   1.01  0   0    0;...
60     0   0   0   0   1.01  0    0;...
61     0   0   0   0   0   1.01   0;...
62     0   0   0   0   0    0   1.01 ;
63     ];
64
65
66 %--------------- time update ----------------------------------------------
67 % compute state prediction with system equation
68 x_old=x;
69 x(1)=(1-dt/tau)*x_old(1);
70 x(2)=(1-dt/tau)*x_old(2);
71 x(3)=(1-dt/tau)*x_old(3);
72 x(4)=dt/2*  (-x_old(5)*x_old(1) - x_old(6)*x_old(2) - x_old(7)*x_old(3)) + x_old(4);
73 x(5)=dt/2*  ( x_old(4)*x_old(1) - x_old(7)*x_old(2) + x_old(6)*x_old(3)) + x_old(5);
74 x(6)=dt/2*  ( x_old(7)*x_old(1) + x_old(4)*x_old(2) - x_old(5)*x_old(3)) + x_old(6);
75 x(7)=dt/2*  (-x_old(6)*x_old(1) + x_old(5)*x_old(2) + x_old(4)*x_old(3)) + x_old(7);
76
77 % normalize quaternion part of the state vector
78 norm_x=1/sqrt(x(4).^2+x(5).^2+x(6).^2+x(7).^2);
79 x=[x(1:3);zeros(4,1)]+[zeros(3,1);x(4:7)].*norm_x;
80
81 % compute linearized system matrix
82 A_lin=[ (1-dt/tau)   0            0            0            0            0
       0;...
83         0            (1-dt/tau)   0            0            0            0
              0;...
84         0            0            (1-dt/tau)   0            0            0            0;
              ...
85         -dt/2*x(5)   -dt/2*x(6)   -dt/2*x(7)   1            -dt/2*x(1)   -dt/2*x(2)   -dt
              /2*x(3);...
86         dt/2*x(4)    -dt/2*x(7)   dt/2*x(6)    dt/2*x(1)    1             dt/2*x(3)   -dt
              /2*x(2);...
87         dt/2*x(7)    dt/2*x(4)    -dt/2*x(5)   dt/2*x(2)    -dt/2*x(3)   1            dt
              /2*x(1);...
88         -dt/2*x(6)   dt/2*x(5)    dt/2*x(4)    dt/2*x(3)    dt/2*x(2)    -dt/2*x(1)   1];
89
90 % compute covariance of prediction
91 P=A_lin*P*A_lin'+Q;
92
93 %--------measurement update----------------------------------------------
94 % compute quaternion measurement vector
95 acc_b=[acc_x;acc_y;acc_z]; % gravity vector in body frame
```

```
96  acc_e=[0.0;0.0;1.0]; % gravity vector in earth frame
97  mag_e=[0.0;1.0;0.0]; % magnetic field vector in eart frame
98
99  % in order to generate artificial magnetometer measurements, the magnetic field vector
100 % in earth frame is rotated by the current attitude estimate (q*)
101 mag_x= 2*x(4)*x(7)+2*x(5)*x(6);
102 mag_y= x(4).^2-x(5).^2+x(6).^2-x(7).^2;
103 mag_z= -2*x(4)*x(5)+2*x(6)*x(7);
104 mag_b=([mag_x;mag_y;mag_z]); % magnetic field vector in body frame
105
106 % compute measurement quaternion with TRIAD algorithm
107 acc_b_x_mag_b=cross(acc_b,mag_b);
108 acc_e_x_mag_e=cross(acc_e,mag_e);
109 M_b=[acc_b acc_b_x_mag_b cross(acc_b_x_mag_b,acc_b)];
110 M_e=[acc_e acc_e_x_mag_e cross(acc_e_x_mag_e,acc_e)];
111 Rot_m=M_e*M_b'; % Rotation matrix represents orientation derived from magnetometer and
        accelerometer
112 qw=1/2*sqrt(1+Rot_m(1,1)+Rot_m(2,2)+Rot_m(3,3)); % compute scalar quaternion from
        rotation matrix
113 % check: if qw=0, then derive qx, qy or qz from diagonal elements
114 qm=[qw;(Rot_m(3,2)-Rot_m(2,3))/4/qw;(Rot_m(1,3)-Rot_m(3,1))/4/qw;(Rot_m(2,1)-Rot_m(1,2))
        /4/qw]; % compute complex quaternion from rotation matrix
115
116 fy=[x(1);x(2);x(3);x(4);x(5);x(6);x(7)]; % create model measurement vector
117 data=[omega_x;omega_y;omega_z;qm]; % create sensor measurement vector
118
119 e=data-fy; % compute error between sensor and model output values
120
121 % compute kalman gain and state covariance without matrix inversion
122 K=zeros(7,7);
123
124 H=eye(7);% Output matrix H is identy, so may be left out
125 for i=1:7
126     % compute Kalman gain
127     K(:,i)=P(i,:)./(P(i,i)+R(i,i));
128     % update state vector
129     x=x+K(:,i)*e(i);
130     % update covariance matrix
131     P= (eye(7)-K(:,i)*H(i,:))*P*(eye(7)-K(:,i)*H(i,:))'+K(:,i)*R(i,i)*K(:,i)';
132 end
133
134 % normalize quaternion part of the state vector
135 norm_x=1/sqrt(x(4).^2+x(5).^2+x(6).^2+x(7).^2);
136 x=[x(1:3);zeros(4,1)]+[zeros(3,1);x(4:7)].*norm_x;
137
138 quat=[x(4);x(5);x(6);x(7)]; % create output vector
```