FH MÜNSTER
University of Applied Sciences

ugr | Universidad de Granada

MÜNSTER UNIVERSITY OF APPLIED SCIENCES
Department of Electrical Engineering and Computer Science

KALMAN FILTERING APPLIED TO PITCH ANGLE ESTIMATION FROM INERTIAL DATA OF THE LOWER EXTREMITIES

ROBIN WEISS

A thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Electrical Engineering

June 2015

Robin Weiß: *Kalman Filtering Applied to Pitch Angle Estimation from Inertial Data of the Lower Extremities* – A thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Electrical Engineering, © June 2015

Available online:
`http://www.robinweiss.de/static/pdf/bachelor-thesis.pdf`

SUPERVISORS:
Prof. Dr.-Ing. Peter Glösekötter
Ph. D. Alberto Olivares Vicente

LOCATION:
Granada, Spain

SUBMITTED:
3. June 2015

ABSTRACT

The analysis of human gait can assist the diagnosis of diseases, and can help to assess treatment success in rehabilitation. In order to estimate the orientation of the human body, inertial sensors have become increasingly important, as they mitigate the drawbacks of camera-based motion capture systems. Using a kinematic model of the leg, the acceleration due to motion was subtracted from the readings of a biaxial accelerometer on the shank, in order to improve the accelerometer-based pitch angle estimate. This improved estimate was fused with measurements of another biaxial accelerometer on the thigh, and two uniaxial gyroscopes on the thigh and shank in an extended Kalman filter. The filter algorithm was applied to movement data from a real subject, which walked on a treadmill at different speeds. The filter improved the overall pitch angle estimate by an average root-mean-square error that was 28.52 % smaller than the one of an existing classical Kalman filter. It was concluded that motion-based acceleration correction can benefit the accuracy of the pitch angle estimates, but further testing of the robustness of the filter algorithm with a larger set of data is proposed. Additionally, a more complete kinematic model of the leg could further improve the angle estimates.

PREFACE

This thesis was submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Electrical Engineering. It describes the implementation of a new Kalman filter-based orientation algorithm that improves the estimation of the pitch angles from inertial data of the lower extremities. I took part in the joint research project "Human Body Motion Analysis of Patients with Neurodegenerative Diseases by Means of Inertial Sensors" between the CITIC-UGR, Spain, and the Department of Neurology of the Klinikum Großhadern, which is part of the Ludwig Maximilian University of Munich, Germany. The goal of the overall project was to obtain several gait parameters by wearable inertial sensors and validate them against conventional methods, such as cameras in combination with visual markers and force measuring platforms. Prior to this thesis, I completed a three-month internship at the CITIC-UGR, in which I worked on the synchronisation of a force measuring platform with inertial sensors within the above-mentioned project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

ARW	Angle random walk
CITIC	Research Centre for Information and Communications Technologies
CITIC	University of Granada
EKF	Extended Kalman filter
IMU	Inertial measurement unit
LTSD	Long term spectral detector
MARG	Magnetic, angular rate, and gravity
MEMS	Microelectromechanical systems
MIMU	Magnetic inertial measurement unit
NED	North-east-down
RMSE	Root-mean-square error

NOTATION

a	Amplitude of an oscillating mode
$\mathbf{a}$	Acceleration vector, $\mathbf{a} \in \mathbb{R}^3$
$\mathbf{B}$	Control matrix that relates the control input to the state $\mathbf{x}$
$\mathbf{C}_{bw}$	Transformation matrix transforming a position vector from the body frame to the world frame
$\mathbf{C}_{wb}$	Transformation matrix transforming a position vector from the world frame to the body frame
d	Desired filter response of a linear discrete-time filter
D	Damping coefficient

e	Error signal of a linear discrete-time filter
$\mathbf{E}$	Orthonormal basis $\{x, y, z\} \in \mathbb{R}^3$
$\mathbf{f}$	Force vector, $\mathbf{f} \in \mathbb{R}^3$
F	One-dimensional force
$\mathbf{g}$	Gravity vector
$\|\mathbf{g}\|$	Magnitude of the gravity vector
$h_0, h_1, h_2, \ldots$	Impulse response of a linear discrete-time filter
$\mathbf{h}$	Functional denoting the *non-linear* measurement matrix function of a discrete dynamical system
$\mathbf{H}$	Measurement sensitivity matrix defining the linear relationship between the state of the dynamical system and the measurements that can be made
$\mathbf{H}^{[1]}$	Linear approximation of the measurement sensitivity matrix
$\mathbf{I}_n$	Identity matrix $\mathbf{I}_n \in \mathbb{R}^{n \times n}$
k	Discrete time normalised to sampling interval, that is sample number, $k \in \mathbb{N}^0$
k_x	Spring constant along the x-axis
K	Weighting factor
$\mathbf{K}$	Kalman gain matrix
m	Mass
μ	Mean value of a conditional probability density
n	Fixed discrete time, non-dimensional number, or number of iterations applied to recursive algorithms
ω	Scalar angular velocity
$\boldsymbol{\omega}$	Angular velocity, $\boldsymbol{\omega} \in \mathbb{R}^3$
Ω	Resonance frequency
$\boldsymbol{\Omega}_{\mathbf{E} \to \mathbf{E}'}$	Function that transforms a position vector $\mathbf{p}$ in the vector space $\mathbf{E}$ into the vector $\mathbf{p}'$ in the vector space $\mathbf{E}'$
$\mathbf{p}$	Position vector in a three-dimensional vector space
$\mathbf{P}$	Covariance matrix of state estimation uncertainty
ϕ	Roll angle that determines the rotation around the x-axis

$\boldsymbol{\phi}$	Functional denoting the *non-linear* transition matrix function of a discrete dynamical system
$\boldsymbol{\Phi}$	State transition matrix of a discrete linear dynamical system
$\boldsymbol{\Phi}^{[1]}$	Linear approximation of the state transition matrix of a discrete linear dynamical system
ψ	Yaw angle that determines the rotation around the z-axis
$\mathbf{Q}$	Covariance matrix of process noise in the system state dynamics
$\mathbf{R}$	Covariance matrix of measurement noise
σ	Standard deviation of a random variable
σ^2	Variance of a random variable
t	Continuous time
$\mathbf{T}$	Transformation matrix
θ	Pitch angle that determines the rotation around the y-axis
u	Scalar nominal velocity
$\boldsymbol{v}$	Measurement noise vector, $\boldsymbol{v} \in \mathbb{R}^m$; or mass velocity, $\boldsymbol{v} \in \mathbb{R}^3$
w	Scalar noise term
$\boldsymbol{w}$	Process noise vector
$\boldsymbol{w} \sim \mathcal{N}(\mu, \sigma^2)$	The random variable $\boldsymbol{w}$ is distributed normally with mean μ and variance σ^2
x	One-dimensional location
$\hat{x}$	Estimate of x
Δx	One-dimensional displacement in x-direction
x, y, z	Axes of the fixed world frame
X, Y, Z	Axes of the moving body frame
$\boldsymbol{x}$	State vector of a linear dynamical system
$\boldsymbol{x}_k$	The kth element of a sequence $\ldots$, $\boldsymbol{x}_{k-1}$, $\boldsymbol{x}_k$, $\boldsymbol{x}_{k+1}, \ldots$ of vectors
$\hat{\boldsymbol{x}}$	Estimate of the state vector of a linear dynamical system

$\hat{\mathbf{x}}_k^-$	A priori estimate of $\hat{\mathbf{x}}$, conditioned on all prior measurements except the one at time t_k
$z(0), z(1), z(2), \ldots$	Time series that serves as input to a linear discrete-time filter
z^{-1}	Unit-delay
$\boldsymbol{z}$	Observation or measurement vector of a dynamical system
$\hat{\boldsymbol{z}}$	Prediction of the measurement vector

1 INTRODUCTION

Monitoring and assessment of human body motion, in particular the analysis of gait, has become an integral part of medical diagnosis, therapy techniques, and rehabilitation [1]. *Gait analysis* involves the measurement and assessment of quantitative parameters that characterise human locomotion. First research in this field was conducted in the late 19th century [1]. The quantitative data enable physicians to diagnose a variety of medical conditions, validate treatment success, set goals in rehabilitation and regularly alter them when necessary. However, standard gait analysis based on multi-camera motion capture systems and force measuring platforms require specialised gait laboratories, expensive equipment, and lengthy setup times. Moreover, the assessments of gait based on measurements performed in clinical settings might not be truly representative [2].

Unobtrusive wearable sensors mitigate the aforementioned limitations. The progressive miniaturisation of inertial and magnetic field sensors has made them more acceptable to patients and has consequently led to an increasingly pervasive adoption for medical applications [3]. Low cost sensors have been successfully employed in clinical and home environments to constantly monitor the movements of patients [4]. Additionally, wearable inertial and magnetic sensors are used to capture gait kinematics, among others. *Kinematics* is a branch of classical mechanics, which is concerned with the motion of objects without reference to the forces causing the motion. The position, i. e. the orientation, the velocity, and the acceleration of a body are of particular interest in kinematics. All three can be estimated from inertial data.

The orientation of the legs is essential in gait analysis. For the application in health care accurate orientation estimates are crucial. A high degree of precision based on data from miniaturised sensors necessitates adequate signal processing, in order to mitigate the influence of disruptive factors, such as bias instability and noise, among others. The signal processing of inertial and magnetic data encompasses calibration, adaptive filtering, and sensor fusion. The latter two were carried out in an extended Kalman filter in the course of this thesis.

1.1 MOTIVATION

Gait analysis provides a powerful means to derive diagnostic information about the functioning of the musculoskeletal, vestibular, and

central and peripheral nervous system [5]. Accurate orientation estimation of the extremities by means of wearable inertial and magnetic field sensors allows objective assessment of human gait without the aforementioned constraints of camera-based motion capture systems. A more reliable and more precise orientation estimation would enable an even more informative gait analysis. Therefore, a multitude of applications in the medical field would profit from a more accurate orientation estimation [6]. The direct relation to health care and the resulting possibility to improve the quality of life of many patients was the motivation for this thesis.

1.2 GOALS

A system for human body motion analysis based on wearable sensors had been developed earlier, in order to gather and process movement data of patients. A detailed description of the so-called GaitWatch system is found in Section 4.1. The goal of this thesis was implementing a new Kalman filter based orientation algorithm proposed by Bennett, Jafari and Gans in [7], in order to improve the estimation of the orientation angles of the human leg. After the adaptation of the proposed mathematical model of the leg and the extended Kalman filter to the existing system, the algorithm should be implemented using the numerical computing environment MATLAB®. Subsequently, the results should be validated against existing algorithms by comparing their respective root-mean-square error. Other than in [7], the filter algorithms should be applied to data of real patients for testing.

1.3 METHODOLOGY

This document presents my work within the overall project in a chronological order. Subsequent to the previous introductory overview of the topic and the definition of the project objectives, this chapter ends with a description of the state of the art. To accomplish the tasks defined in the previous section, I had to acquire knowledge regarding various subjects. Chapters 2 and 3 outline the necessary fundamentals of MARG sensors and orientation estimation, and digital filters, respectively. This enables comprehension of the overall project, even for readers that are not familiar with some of the subjects. Those readers are referred to Chapters 2 and 3 at this point, before reading the state of the art. The actual implementation of the Kalman filter, including a prior theoretical design is given in Chapter 4. This chapter also encompasses the experimental setup, the results and a discussion of the latter. Finally, Chapter 5 covers conclusions and future work.

As additional means to communicate with my supervisor and in order to enable him to follow the progress of my work at any time we used Pivotal Tracker, a tool for agile project management, and

GitHub, a repository hosting service based on the distributed version control system Git. This thesis was written in LaTeX.

1.4 STATE OF THE ART

There are several research works in the literature dealing with orientation estimation by means of inertial sensors. Kalman filters have been used successfully to improve the estimation of orientation angles from inertial data. The state of the art at the commencement of the project is described below. Subsequently, applications of wearable inertial sensors in health care are presented.

1.4.1 *Kalman Filtering Applied to Orientation Estimation*

Considering the fact that inertial and magnetic field sensors are used to establish objective body motion parameters that affect medical diagnosis, therapy, and rehabilitation, the necessity of high levels of accuracy becomes obvious. In order to obtain precise orientation estimates from sensor data, it is essential to alleviate the effects of measurement noise and to combine the advantages of different sensors through sensor fusion. Therefore, a wide variety of Kalman filter algorithms have been developed in the past few years. It is common practice to fuse accelerometer and gyroscope measurements to mitigate their respective drawbacks and thus obtain more accurate angle estimates.

Luinge, Veltink and Baten [8] alleged that the gravitational component of the acceleration signal has a greater magnitude than the component caused by motion for many human movements. They estimated the tilt angle, which is defined as the angle between the sensor axes and the vertical. The separate estimates from an accelerometer and a gyroscope were fused with a Kalman filter. To test their method they moved the sensors around by hand for 30 seconds and then put them in a known position. The orientation obtained by integrating the angular rate served as an additional reference. They concluded that a fusion of accelerometer and gyroscope signals accounts for a considerable improvement of the orientation estimation. This approach lacks dynamical comparison, since it only compares the errors at specific static positions.

Due to human motion intensity usually being subject to change, Olivares Vicente implemented a *gated Kalman filter* in [9]. They modelled linear acceleration during intense motion as noise and improved the performance of the Kalman filter by dynamically adjusting the variance of both the process and measurement noise, according to the motion intensity. For that purpose, they applied a LTSD and set the variance between two predefined values. Then, the gated Kalman filter fused the information from the accelerometer and the gyroscope

signals. With this method they improved the adapting capability of the filter and consequently the precision of the orientation estimation.

Bennett, Jafari and Gans demonstrated in [7] that accelerometer angle estimates are inaccurate for typical motions of the leg. They affirmed the need to decouple the acceleration due to motion from the acceleration due to gravity, since the former cannot be neglected during fast motions. Therefore, they deployed a kinematic model of the leg to determine the acceleration that occurs due to motion and corrected the acceleration signal accordingly. An extended Kalman filter fused the corrected acceleration signal with measurements of a gyroscope. They tested the filter algorithm by moving a mechanical two link pendulum by hand. Their method improved upon the raw acceleration method during motion and at rest by an 83% smaller root-mean-square error. Their proposed approach is the foundation of the filter algorithm implemented in Section 4.2.

1.4.2 *Wearable Sensors in Health Care*

Inertial sensors can be found in smart phones, fitness trackers, and other wearable devices, among others. With increasing capability of body sensor networks and wearable computing, they have become prevalent in research environments for estimation and tracking of human body motion [7]. They are used in activity monitoring [10–12], rehabilitation [13, 14], sports training [12, 15], and localisation [16, 17]. Also, emergency falls of elderly people were detected by means of inertial sensors [18–20].

Many neurodegenerative diseases, such as Parkinson's disease, impair stable stance and gait, and reduce the patient's mobility. Thus, they diminish the quality of life significantly. *Parkinson's disease* is a movement disorder that is characterised by marked slow movements, tremors, and unstable posture. Especially in advanced stages of the disease many patients exhibit an episodic, brief inability to step, which delays gait initiation or interrupts ongoing gait. In fact, one of the most reliable diagnostic criterion of the disease is gait [1]. Hence, wearable motion sensors have been used successfully to objectively classify the severity of the disease [21–23].

Stroke patients, who regained their walking ability, need to undergo rehabilitation to recover their independent mobility. Ambulatory gait analysis provides a means to assess the function of the lower extremities of hemiparetic post-stroke patients and follow the progress of rehabilitation [1, 24]. In addition, the presence of neurologic gait abnormalities is used as a significant predictor of the risk of developing dementia [25].

2

ORIENTATION ESTIMATION USING MARG SENSORS

This chapter covers the working principals of MARG sensors, as well as the fundamentals of orientation estimation, that are necessary for the implementation of the aforementioned system. Subsequently, a mathematical construct used to express orientation – Euler angles – is explained. Towards the end of the chapter, different approaches to compute orientation estimates from magnetic and inertial data including their pros and cons are described. Finally, sensor fusion as a means to mitigate the drawbacks of each approach is introduced.

2.1 MARG SENSORS

MARG sensors is a collective term for magnetic, angular rate, and gravitational sensors, which encompasses inertial sensors, as well as magnetic field sensors, also referred to as magnetometers. Inertial sensors itself generally fall into two categories: instruments sensing linear inertial displacement, i. e. accelerometers, and rotational inertial rate sensors, that is gyroscopes. They are applied in various contexts to quantify vibration, motion, and shock [26]. Particularly, the development of MEMS opened up many medical applications as stated in Section 1.4.2. MEMS sensors have low manufacturing costs, small physical size, and low power consumption [26]. This section compiles the functional principles of different MARG sensors and introduces IMU as a combination of those.

2.1.1 *Accelerometers*

Accelerometers measure the acceleration of an object relative to an inertial frame. Since acceleration cannot be sensed directly, the force exerted on a reference mass is measured. The resultant acceleration is computed according to Newton's second law $\mathbf{f} = m \cdot \mathbf{a}$, where $\mathbf{f} \in \mathbb{R}^3$ denotes the force vector, m the mass, and $\mathbf{a} \in \mathbb{R}^3$ the acceleration vector. Usually, a single axis accelerometer consists of a small proof mass connected via a spring to the case of the instrument. The proof mass is displaced by Δx with respect to the case, when the instrument experiences a certain acceleration along its sensitive axis. Disregarding drag force, according to Hooke's law $f = -k \cdot \Delta x$, the displacement is directly proportional to the force exerted by the mass and thus to the acceleration. Therefore, by measuring the displacement of the proof mass the acceleration can be obtained. Figure

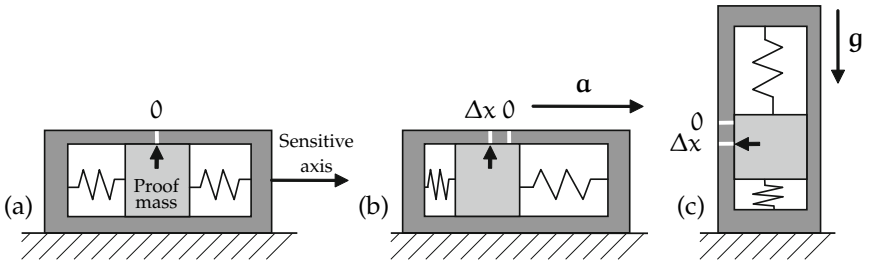

Figure 1: A mass-and-spring accelerometer under different conditions: (a) at rest or in uniform motion, (b) accelerating, and (c) at rest, being exposed to the gravity g, from [26].

1 shows the displacement Δx of the mass with respect to the case of the instrument for three different conditions: (a) at rest or in uniform motion, (b) accelerating, and (c) at rest, being exposed to the gravity g. According to how the mass displacement is sensed, accelerometers can be classified as resistive, capacitive, and piezoelectric. Besides, there are surface acoustic wave, fibre optic, vibrating beam and solid-state MEMS accelerometers. To obtain a three-dimensional accelerometer, three single-axis accelerometers are mounted together. Nowadays, most accelerometers are manufactured using MEMS technology, which was developed for the military and aerospace markets in the 1970s [26].

2.1.2 *Gyroscopes*

Gyroscopes are used for measuring and maintaining angular orientation. In essence, based on two different physical principles, namely the Sagnac and Coriolis effect, gyroscopes sense angular velocity, which is why they are also referred to as angular velocity sensors or angular rate sensors. By integrating the angular velocity the rotation angle can be obtained. Here we will only elaborate on the working principle of vibrating gyroscopes, since they are utilised in the Gait-Watch system. Armenise et al. give a comprehensive overview of current gyroscope technologies in [27].

Coriolis vibratory gyroscopes, or vibrating gyros for short, sense angular velocity based on the effect of Coriolis force on a vibrating mass. The Coriolis force is a fictitious force experienced by a mass m moving in a rotating reference frame. It can be calculated as: $f_C = -2m(\omega \times v)$, where v is the mass velocity in the rotating reference frame and ω is the angular velocity of the reference frame. As seen in this equation the Coriolis force is only present when the mass varies its distance with respect to the spin axis. Otherwise, if ω and v are parallel, the cross product becomes zero. The two degree-of-freedom spring-mass-damper system shown in Figure 2 serves as a simple model of a vibrating angular rate sensor. The mass m can move

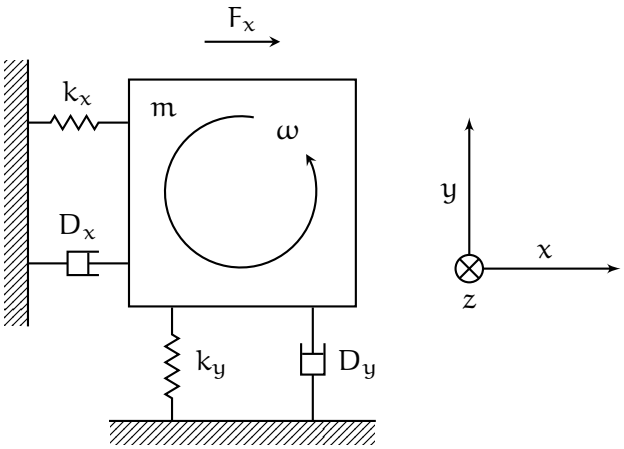

Figure 2: A simple model of a Coriolis vibratory gyroscope: A two degree-of-freedom spring-mass-damper system in a rotating reference frame, from [27].

along the x and y-axis, respectively. The angular velocity around the z-axis is denoted with ω. The drive or primary oscillating mode, that is, the oscillation along x, is excited by the force F_x directed along the x-axis. The oscillation along y, called sense or secondary oscillating mode, is due to system rotation around the z-axis. D_x and D_y are the damping coefficients and k_x and k_y are the spring constants along the x and y-axis, respectively. Typically, the primary oscillating mode is excited by a sinusoidal force with an angular frequency close to the resonance frequency, so that $\Omega_x \cong \sqrt{k_x/m}$. Its amplitude is kept constant at a_x. As shown in [27], the amplitude of the sense mode is then given by

$$a_y = -\frac{2a_x\Omega_x\omega}{\sqrt{(\Omega_x^2 - \Omega_y^2)^2 + \Omega_x^2\Omega_y^2/Q_y^2}}, \tag{1}$$

where $\Omega_y = \sqrt{k_y/m}$ is the resonance frequency of the secondary resonator and $Q_y = \sqrt{mk_y}/D_y$ its quality factor. The amplitude a_y is directly proportional to the angular rate of the two degree-of-freedom spring-mass-damper system. Thus, ω can be estimated by measuring the amplitude of the oscillation along the y-axis.

Usually, vibrating gyroscopes are manufactured using MEMS technology. MEMS gyros are of low to medium accuracy [26], but due to their size they are ideally suited for medical applications.

2.1.3 *Magnetometers*

Magnetometers measure the strength and the direction of the magnetic field at a point in space. There are numerous techniques used to produce magnetic field sensors, which exploit a broad range of physical phenomena [28]. Lenz and Edelstein give a complete survey of

common technologies used for magnetic field sensing in [28]. Many MEMS magnetometers sense mechanical motion of a MEMS structure due to Lorentz force and estimate the strength of the magnetic field according to the displacement. When an external magnetic field interacts with a current-carrying silicon MEMS structure the Lorentz force causes a displacement of this structure. Piezoresistive, capacitive, or optical sensing can be used to detect the displacement of the MEMS structure. MEMS Lorentz force magnetometers are free from hysteresis, require no specialised materials and can be monolithically integrated with other MEMS inertial sensors [29].

2.2 INERTIAL MEASUREMENT UNITS

Devices using a combination of accelerometers and gyroscopes to measure the orientation of a rigid body with up to six degrees of freedom are referred to as IMU. If they include additional magnetometers they are termed MIMU. The number of degrees of freedom states the number of independent motions, with respect to a reference frame that are allowed to the body in space. MIMU are portable and relatively inexpensive. They can be easily attached to the body and thus allow non-clinical longterm application. Their drawbacks are complex calibration procedures and drift behaviour over time, depending on intensity and duration of the measurement interval. Hence, in order to maintain a satisfactory degree of precision, periodical recomputation of the calibration parameters is required [9].

2.3 EULER ANGLES

As well as in aircraft navigation, in the motion monitoring field the position of the coordinate frame of the body, that is the *body frame*, with respect to a reference coordinate frame, termed the *world frame*, is known as *attitude*, which is used as a synonym of orientation. *Euler angles* are one of several mathematical ways to describe the attitude of an object in three-dimensional Euclidean space. They represent a sequence of three elemental rotations about the axes of the coordinate system, defined as follows:

- The *roll* angle ϕ determines the rotation around the x-axis.
- The *pitch* angle θ determines the rotation around the y-axis.
- The *yaw* angle ψ determines the rotation around the z-axis.

Figure 3 depicts the rotation about the axes z, y', X by ψ, θ, ϕ, respectively, according to the Tait-Bryan convention. The colour blue indicates the world frame $\{x, y, z\}$, which matched the body frame $\{X, Y, Z\}$ before the rotations. The colour red indicates the orientation of the

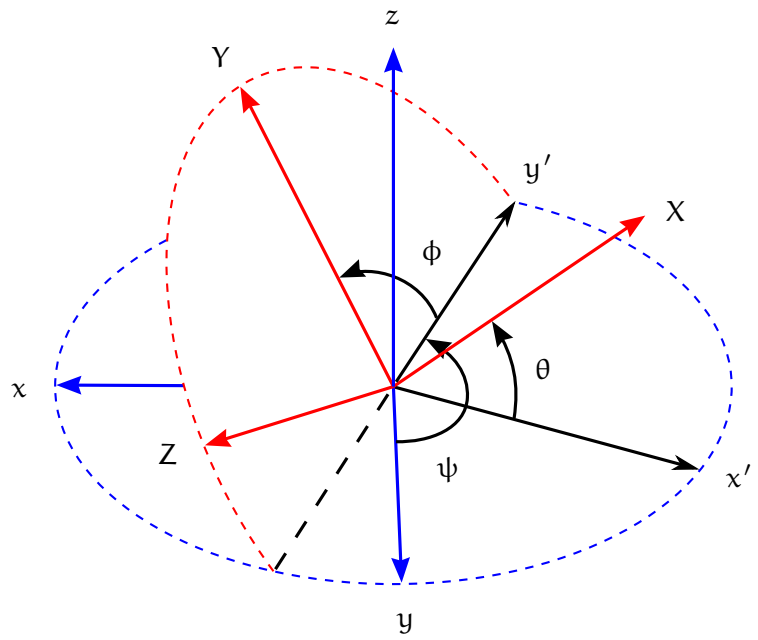

Figure 3: Representation of the body frame, depicted in red, with respect to the world frame, depicted in blue, from [30]. The body frame was rotated, by the Euler angles ψ, θ, ϕ about the axes z, y', X, respectively.

body frame after the rotations were carried out. In contrast to *extrinsic rotations*, where each of the three elemental rotations may occur about the axes of the original coordinate system, the Tait-Bryan rotations are *intrinsic rotations* that occur about the axes of the rotating coordinate system, which changes its orientation after each rotation.

Euler angles are a simple and intuitive means to represent rotations in three-dimensional space. However, for the above mentioned parameterisation they have singularities at values of $\theta = n\pi$, $n \in \mathbb{Z}$. At these points a rotation about the x-axis and the z-axis constitute the same motion, which results in the loss of one degree of freedom and makes changes in ϕ and ψ indistinguishable. This phenomenon is called *gimbal lock*.

2.3.1 *Transformation Matrix*

Coordinates representing a point in one coordinate system can be transformed to another. Such a transformation can be expressed as a multiplication of a matrix with the coordinate vector that is to be transformed. Let $\mathbf{E}$ denote the orthonormal basis $\{x, y, z\} \in \mathbb{R}^3$ and let $\mathbf{E}'$ denote the orthonormal basis $\{X, Y, Z\} \in \mathbb{R}^3$. Furthermore, let $\mathbf{p}$ denote the position vector of an arbitrary point in three-dimensional Euclidean space. The coordinate transformation from $\mathbf{E}$ to $\mathbf{E}'$ is denoted

$\mathbf{\Omega}_{\mathbf{E}\to\mathbf{E}'} : (p_1, p_2, p_3) \mapsto (p'_1, p'_2, p'_3)$. Then, the linear transformation from $\mathbf{p}$ to $\mathbf{p}'$ is given by

$$\mathbf{p}' = \mathbf{\Omega}_{\mathbf{E}\to\mathbf{E}'}(\mathbf{p}) = \mathbf{T}\mathbf{p}\,, \tag{2}$$

where $\mathbf{T}$ is the *transformation matrix*, which is a function of the rotation angles between the two coordinate systems.

In order to transform the coordinate vector from the *world frame* to the *body frame*, according to the common aerospace rotation sequence mentioned above and the NED coordinate system, the transformation matrix $\mathbf{C}_{wb}$ is given by

$$\begin{aligned}
\mathbf{C}_{wb} &= \mathbf{T}_x(\phi)\mathbf{T}_y(\theta)\mathbf{T}_z(\psi) \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix}
\end{aligned} \tag{3}$$

Plugged in Equation 2 ($\mathbf{T} = \mathbf{C}_{wb}$), the pre-multiplications of the matrices $\mathbf{T}_x(\phi), \mathbf{T}_y(\theta), \mathbf{T}_z(\psi)$ to the vector $\mathbf{p}$ represent the coordinate rotations about the single axes x, y', Z, according to the right hand rule, respectively. That is, the function $\mathbf{\Omega}_{\mathbf{E}\to\mathbf{E}'}$ maps the vector $\mathbf{p}$ to its orthogonal projection onto the axes of the coordinate system, $\mathbf{p}'$, which result from the respective two-dimensional rotation of ϕ, θ, ψ about the axes x, y', Z. This is illustrated for a single rotation around the z-axis by the angle ψ in Figure 4. Note that $\{x', y', z'\}$ denotes the coordinate frame after the first elemental rotation. The matrices $\mathbf{T}_x(\phi), \mathbf{T}_y(\theta)$, and $\mathbf{T}_z(\psi)$ are also known as direction cosine matrices, since their elements are the cosines of the unsigned angles between the body-fixed axes and the axes of the world frame, as shown in [31]. The form stated here is already simplified. The matrix $\mathbf{C}_{bw}$ that transforms a coordinate vector from the body frame to the world frame is given by

$$\mathbf{C}_{bw} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{4}$$

Note that $\mathbf{C}_{bw} = \mathbf{C}_{wb}^{\mathsf{T}} = \mathbf{C}_{wb}^{-1}$. Thus, $\mathbf{C}_{bw}$ and $\mathbf{C}_{wb}$ are orthogonal matrices so that $\mathbf{C}_{bw}\mathbf{C}_{wb} = \mathbf{I}_3$, where $\mathbf{I}_3 \in \mathbb{R}^{3\times3}$ is the identity matrix.

2.4 PROJECTION OF THE GRAVITY VECTOR

As described in Section 2.1.1, accelerometers measure the linear acceleration they experience. Under static or quasi-static conditions, that

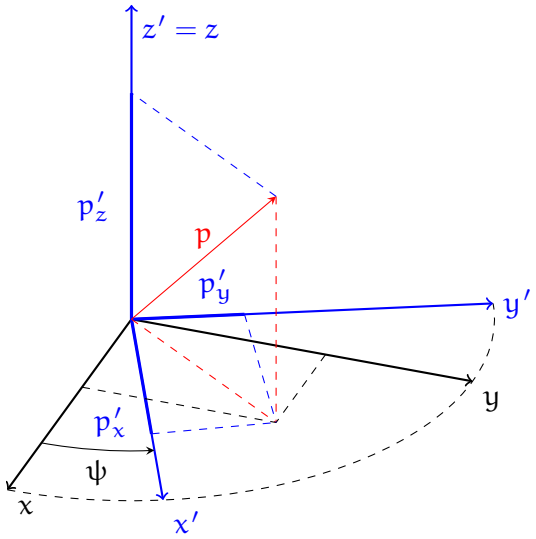

Figure 4: An exemplary coordinate rotation about the z-axis by an angle ψ, illustrating the orthogonal projection on the resulting axes x', y', z'.

is, the sensor is in uniform motion, or at low acceleration, it can be assumed that the measured acceleration is mainly that of gravity. By means of simple trigonometric functions estimates for the pitch and the roll angle can be obtained. Since the gravity vector is perpendicular to the xy-plane, and thus a rotation around the z-axis will not cause any variation in the sensed acceleration, the yaw angle cannot be obtained by this method. To solve this problem a three-dimensional magnetometer is used, which measures the variation of Earth's magnetic field while rotating around the z-axis.

When the accelerometer is motionless, its measurements will be directly related to the angle of the sensor relative to gravity, as depicted in Figure 5 (a). In that case θ with respect to the vertical is given by

$$\theta = \operatorname{atan2}(A_z, A_x)\,, \tag{5}$$

where A_x and A_z are the components of the acceleration vector in x and z-direction, respectively. However, when the sensor is in motion, in addition to the gravity, there are radial and tangential acceleration components due to motion, as depicted in Figure 5 (b). The magnitude of the gravity vector $\mathbf{g}$ is denoted with $\|\mathbf{g}\|$. Ignoring these components will cause incorrect angle estimates.

2.5 INTEGRATION OF THE ANGULAR RATE

Another way to estimate the attitude of an object is the integration of the angular rate around the x, y and z-axis, respectively. Although this would theoretically lead to very accurate orientation estimates,

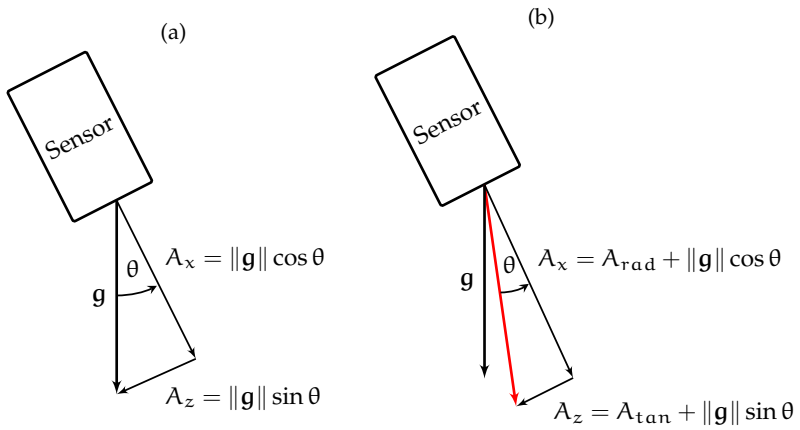

Figure 5: Acceleration seen by the sensor (b) with and (a) without motion, from [7].

they are impaired by ARW and dynamical bias in practice. ARW is an effect caused by the integration of high-frequency, thermo-mechanical noise, which leads to a random additive angle in the orientation signal. An even greater impact than AWR has the gyroscope's dynamic bias, which has its origin in low-frequency flicker noise. Both effects cause a dramatical drift in the angle signal over time.

2.6 SENSOR FUSION

Since the projection of the gravity vector is only valid under static or quasi-static conditions, or at low acceleration, and the integration of the angular rate leads to non-reliable estimates due to ARW and dynamic bias, but is not affected by the intensity of motion, a means to combine the information of both sensors is desirable. The combination of information from multiple sensors to increase the overall precision of the estimation of a certain quantity of interest is termed *sensor fusion*. Raol [32] states the following advantages of sensor fusion:

- Robust functional and operational performance is given, in case of data loss from one sensor, due to redundancy provided by multiple sensors.
- Enhanced confidence in the results inferred from the measurement of one sensor, if they are confirmed by the measurement of another sensor.
- With sensor fusion an arbitrary fine time resolution of measurements is possible, whereas single sensors need a finite time to transmit measurements and so limit the frequency of measurements.

- One sensor might be, to some extent, better in a certain state of the measured process, e. g. low or high motion intensity in attitude estimation, and thus, by fusing multiple sensor signals, a satisfactory accuracy among all states of the process could be attained.

Sensor fusion can be realised by the use of a Kalman filter. This specific digital filter is described in detail in the next chapter, and applied in Chapter 4 to fuse the sensor signals of accelerometers and gyroscopes.

3

DIGITAL FILTERS

Conceived in general terms, a filter is a physical device for removing unwanted components of a mixture. In the technical field, a filter is a system designed to extract information from noisy measurements of a process. That is, the filter delivers an estimate of the variables of principal interest, which is why it may also be called an estimator. Filter theory is applied in diverse fields of science and technology, such as communications, radar, sonar, navigation, and biomedical engineering [33].

In contrast to *analogue filters* that consist of electronic circuits to attenuate unwanted frequencies in continuous-time signals and thus extract the useful signal, a *digital filter* is a set of mathematical operations applied to a discrete-time signal in order to extract information about the hidden quantity of interest. A *discrete-time* signal is a sequence of samples at equidistant time instants that represent the continuous-time signal with no loss, provided the sampling theorem is satisfied, according to which the sample frequency has to be greater than twice the highest frequency component of the continuous-time signal.

Digital filters can be classified as *linear* and *non-linear*. If the quantity at the output of the filter is a *linear* function of its input, that is, the filter function satisfies the superposition principle, the filter is said to be *linear*. Otherwise, the filter is *non-linear*.

3.1 THE FILTERING PROBLEM

Consider, as an example involving filter theory, the continuous-time dynamical system depicted in Figure 6. The desired state vector of the system, $\mathbf{x}(t)$, is usually hidden and can only be observed by indirect measurements $\mathbf{z}(t)$ that are a function of $\mathbf{x}(t)$ and subject to noise. Equally, the equation describing the evolution of the state $\mathbf{x}(t)$ is usually subject to system errors. These could be caused by, for instance, effects not accounted for in the model. The dynamical system may be an aircraft in flight, in which case the elements of the state vector are constituted by its position and velocity. The measuring system may be a tracking radar producing the observation vector $\mathbf{z}(t)$ over an interval $[0, T]$. The requirement of the filter is to deliver a reliable *estimate* $\hat{\mathbf{x}}(t)$ of the actual state, by taking the measurement as well as a prior information into account.

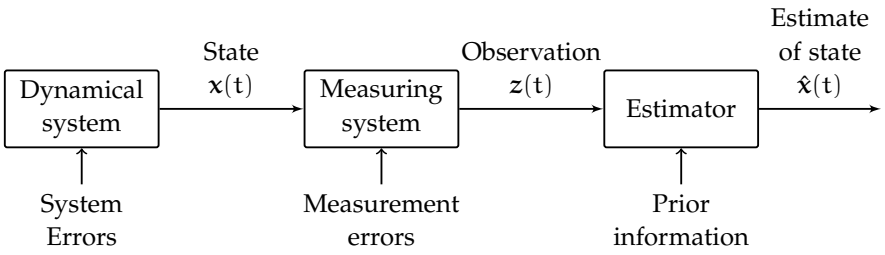

Figure 6: Block diagram depicting the components involved in state estimation, from [33].

3.2 THE WIENER FILTER

A statistical criterion, according to which the performance of a filter can be measured, is the mean-squared error. Consider the linear discrete-time filter with the impulse response $h_0, h_1, h_2, \ldots$ depicted in Figure 7. At some discrete time k it produces an output designated by $\hat{x}(k)$, which provides an estimate of a desired response denoted by $d(k)$. According to Haykin [33], the essence of the filtering problem and the resulting requirement is summarised with the following statement:

> "Design a linear discrete-time filter whose output $\hat{x}(k)$ provides an estimate of the desired response $d(k)$, given a set of input samples $z(0), z(1), z(2), \ldots$, such that the mean-square value of the estimation error $e(k)$, defined as the difference between the desired response $d(k)$ and the actual response $\hat{x}(k)$, is minimized."

Assume a *stationary* stochastic process with known statistical parameters as the mean and correlation functions of the useful signal and the unwanted additive noise. Then, the solution to this statistical optimisation problem is commonly known as the *Wiener filter*. Yet, since the Wiener filter requires a priori information about the statistics of the data to be processed, it may not be optimum for *non-stationary* processes. For such an environment, in which the statistics are time-varying, it needs a filter that constantly adapts its parameters to optimise its output.

3.3 ADAPTIVE FILTERS

A possible approach to reduce the limitations associated with the Wiener filter for non-stationary processes is the 'estimate and plug' procedure. The filter 'estimates' the statistical parameters of the relevant signals and 'plugs' them into a *non-recursive* formula for computing the filter parameters. This procedure requires excessively elaborate and costly hardware for real-time operation [33]. To overcome this

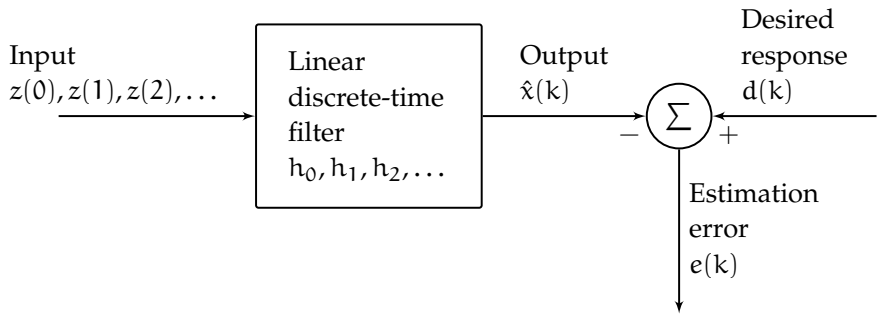

Figure 7: Block diagram representation of the statistical filtering problem, from [33].

disadvantage one may use an *adaptive filter*, which is a self-designing system that relies, in contrast, on a *recursive* algorithm. This allows the filter to perform satisfactorily, even if there is no complete knowledge of the relevant signal characteristics. Provided the variations in the statistics of the input data are sufficiently slow, the algorithm can track time variations and is thus suitable for non-stationary environments. The algorithm starts from some predetermined set of initial conditions respecting the knowledge about the system. In a stationary environment it converges to the optimum Wiener solution in some statistical sense after successive iterations. The *Kalman filter* is one such adaptive filter.

Due to the fact that the parameters of an adaptive filter are updated each iteration, they become data dependent. The system does not obey the principles of superposition, which therefore makes the adaptive filter in reality a *non-linear* system. However, an adaptive filter is commonly said to be *linear* if its input-output map satisfies the superposition principle, as long as its parameters are held fixed. Otherwise it is said to be *non-linear*.

3.4 THE KALMAN FILTER

The *Kalman filter* is a set of recursive mathematical equations that provide an efficient means to estimate the state of a *linear* dynamic system, perturbed by additive white Gaussian noise, even when the precise nature of the modelled system is unknown. It incorporates knowledge of the system and measurement device dynamics, the statistical description of the system errors and measurement noise, and available information about initial conditions of the variables of interest, in order to produce an estimate of these variables, in a way that the mean of the squared error is minimised [34].

The filter is named after Rudolf E. Kalman, who 1960 published his famous paper describing a recursive solution to the discrete-data linear filtering problem [35]. Since that time, the Kalman filter has been the subject of extensive research, due to a large extent to the

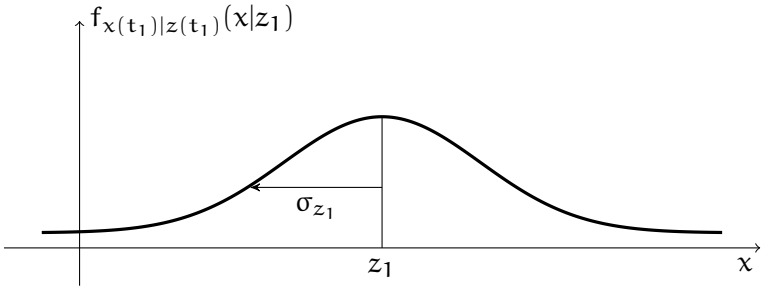

Figure 8: Conditional probability density of the position based on measurement value z_1, from [34].

advances in digital computing [36]. It finds applications in radar tracking, navigation, and orientation estimation, among others. Zarchan and Musoff [37] stated: "With the possible exception of the fast Fourier transform, Kalman filtering is probably the most important algorithmic technique ever devised."

3.4.1 *An Introductory Example*

The following introductory example from Maybeck [34] is an illustrative description of the determination of a one-dimensional position to understand how the Kalman filter works. Suppose you are lost at sea during the night and take a star sighting to determine your approximate position at time t_1 to be z_1. Your location estimate is, due to inherent measurement device inaccuracies and human error, somewhat uncertain, and thus assumed to be associated with a standard deviation σ_{z_1}. The conditional probability of $x(t_1)$, your actual position at time t_1, conditioned on the observed value z_1, is depicted in Figure 8. The best estimate of your position, based on this conditional probability density, is

$$\hat{x}(t_1) = z_1 \,, \tag{6}$$

and the variance of the error in the estimate is

$$\sigma_x^2(t_1) = \sigma_{z_1}^2 \,. \tag{7}$$

Right after you, say a trained navigator friend takes an independent fix at time $t_2 \cong t_1$, so that the true position has not changed at all. He obtains a measurement z_2 with a variance $\sigma_{z_2}^2$, which is somewhat smaller than yours, since he has a higher skill. Figure 9 depicts the conditional probability density of your position at time t_2, based only on the measurement value z_2. Combining these data, your position at time $t_2 \cong t_1$, $x(t_2)$, given both z_1 and z_2, is then a Gaussian density with mean μ and variance σ^2, as indicated in Figure 10, with

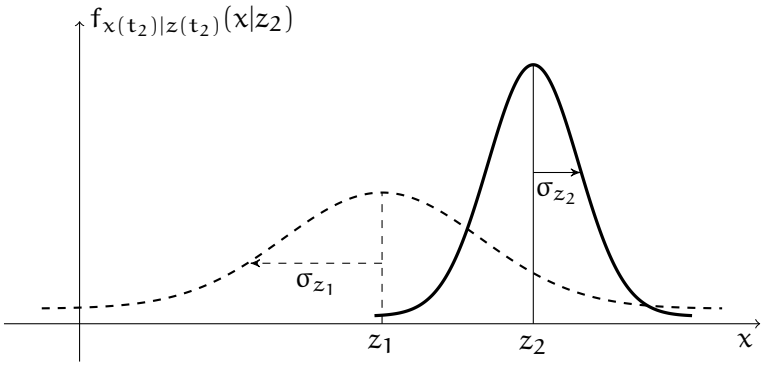

Figure 9: Conditional probability density of the position based on measurement value z_2 alone, from [34].

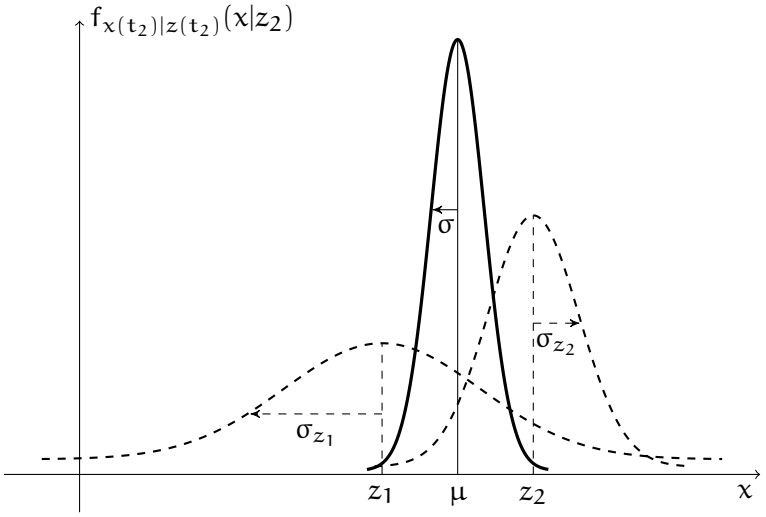

Figure 10: Conditional probability density of the position based on data z_1 and z_2, from [34].

$$\mu = z_1 \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} + z_2 \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \tag{8}$$

and

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_{z_1}^2} + \frac{1}{\sigma_{z_2}^2}\,. \tag{9}$$

The uncertainty in your estimate of position has been decreased because σ is less than either $\sigma_{z_1}^2$ or $\sigma_{z_2}^2$. Even if σ_{z_1} was very large, the variance of the estimate is less than σ_{z_2}, which means that even poor quality data increases the precision of the filter output. The best estimate, given this density, is

$$\hat{x}(t_2) = \mu\,, \tag{10}$$

with an associated error variance σ^2.

Having a closer look at the form of μ in Equation 8, one notices that it makes good sense. If the measurements were of equal precision, meaning $\sigma_{z_1} = \sigma_{z_2}$, the optimal estimate is simply the average of both measurements, as would be expected. If σ_{z_1} is larger than σ_{z_2}, the equation weights z_2 more heavily than z_1.

Equation 10 for the filter output can be written as

$$\begin{aligned} \hat{x}(t_2) &= z_1 \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} + z_2 \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \\ &= z_1 + \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}[z_2 - z_1]\,, \end{aligned} \tag{11}$$

or in a form that is used in Kalman filter implementations, with $\hat{x}(t_1) = z_1$, as

$$\hat{x}(t_2) = \hat{x}(t_1) + K(t_2)[z_2 - \hat{x}(t_1)]\,, \tag{12}$$

where

$$K(t_2) = \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\,. \tag{13}$$

These equations represent the 'predictor-corrector' structure of the Kalman filter. A prediction of the value that the desired variables and the measurements will have at the next measurement time is made, based on all previous information. Then the difference between the measurement and its predicted value is used to correct the prediction of the desired variables. According to Equation 12 the optimal estimate at time t_2, that is $\hat{x}(t_2)$, is equal to $\hat{x}(t_1)$, the best prediction of its value before z_2 is taken, plus a correction term of an optimal weighting value times the difference between z_2 and the best prediction of it before the measurement is actually taken.

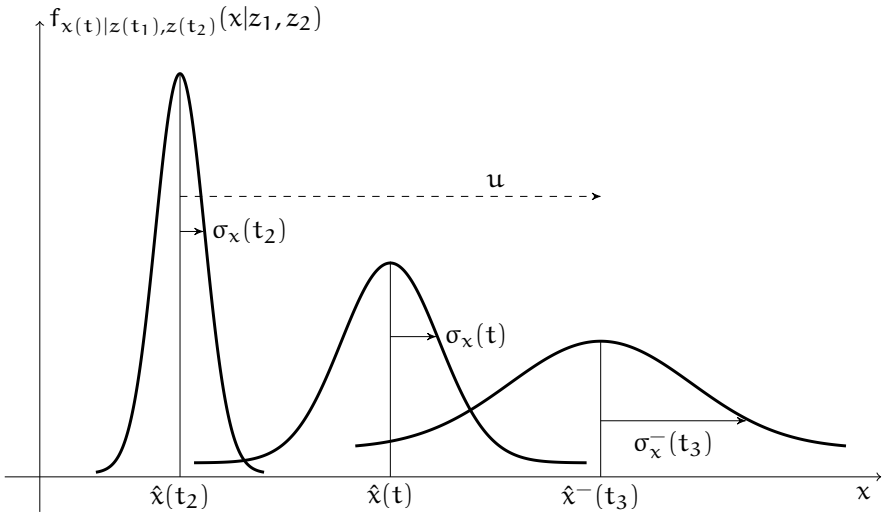

Figure 11: Propagation of conditional probability density, from [34].

To incorporate dynamics into the model, suppose you travel for some time before taking another measurement. The best model you have for your motion may be of the form

$$\frac{dx}{dt} = u + w, \tag{14}$$

where u is a nominal velocity and w is a noise term, representing the uncertainty in your knowledge of the actual velocity due to disturbances and effects not accounted for in the simple first order equation. It will be modelled as white Gaussian noise with a mean of zero and variance of σ_w^2.

The conditional density of the position at time t_2, given z_1 and z_2, was previously derived. Figure 11 shows graphically how the density travels along the x-axis as time progresses. It starts at the best estimate and moves according to the above mentioned model of dynamics. Due to the constant addition of uncertainty over time it spreads out. Thus, as the variance increases, you become less sure of your position. The Gaussian density $f_{x(t_3)|z(t_1),z(t_2)}(x|z_1,z_2)$ can be expressed mathematically by its mean and variance given by

$$\hat{x}^-(t_3) = \hat{x}(t_2) + u[t_3 - t_2], \tag{15}$$

$$\sigma_x^{2-}(t_3) = \sigma_x^2(t_2) + \sigma_w^2[t_3 - t_2], \tag{16}$$

where the superscript $^-$ denotes the prediction of $\hat{x}$ and σ_x^2, respectively. Before the measurement is taken at time t_3, $\hat{x}^-(t_3)$ is the optimal prediction of the location at t_3, associated with the variance $\sigma_x^{2-}(t_3)$ in this prediction.

Now a measurement z_3 with an assumed variance $\sigma_{z_3}^2$ is taken. As before, its conditional probability density is combined with the

density with mean $\hat{x}^-(t_3)$ and variance $\sigma_x^{2-}(t_3)$, to yield a Gaussian density with mean

$$\hat{x}(t_3) = \hat{x}^-(t_3) + K(t_3)[z_3 - \hat{x}^-(t_3)] \tag{17}$$

and variance

$$\sigma_x^2(t_3) = \sigma_x^{2-}(t_3) - K(t_3)\sigma_x^{2-}(t_3)\,, \tag{18}$$

where the gain $K(t_3)$ is given by

$$K(t_3) = \frac{\sigma_x^{2-}(t_3)}{\sigma_x^{2-}(t_3) + \sigma_{z_3}^2}\,. \tag{19}$$

Observing the form of Equation 19 the reasonableness of the filter structure becomes obvious. If the variance of the measurement noise $\sigma_{z_3}^2$ is large, then $K(t_3)$ is small, meaning that little confidence is put in a very noisy measurement and that it is weighted lightly. For $\sigma_{z_3}^2 \to \infty$, $K(t_3)$ becomes zero, and $\hat{x}(t_3)$ equals $\hat{x}^-(t_3)$. Thus, an infinitely noisy measurement is totally ignored. Likewise, if the dynamical system noise variance σ_w^2 is large, then according to Equation 16, $\sigma_x^{2-}(t_3)$ will be large, and so will be $K(t_3)$. Therefore, the measurement is weighted heavily, in case you are not very certain about the output of the system model within the filter structure. In the limit as $\sigma_w^2 \to \infty$, $\sigma_x^{2-}(t_3) \to \infty$, and $K(t_3) \to 1$, so Equation 10 yields

$$\hat{x}(t_3) = \hat{x}(t_3^-) + 1 \cdot [z_3 - \hat{x}(t_3^-)] = z_3\,. \tag{20}$$

That means that in the limit of absolutely no confidence in the system model output, solely the new measurement is taken as the optimal estimate. Finally, if you are absolutely sure of your estimate before z_3 becomes available, $\sigma_x^{2-}(t_3)$ would become zero, and so would $K(t_3)$, which means that the measurements would be left disregarded.

Extending Equations 15, 16, 17, 18, and 19 to the vector case, and allowing time varying parameters in the system and noise description leads to the general Kalman filter equations. A complete mathematical derivation can be found in Haykin [33].

3.4.2 *Formulation of the Kalman Filter Equations*

Let $\boldsymbol{x}_k \in \mathbb{R}^n$ be the state vector of a discrete-time controlled process governed by the *linear* stochastic difference equation

$$\boldsymbol{x}_k = \boldsymbol{\Phi}_{k-1}\boldsymbol{x}_{k-1} + \mathbf{B}_{k-1}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \tag{21}$$

and $\boldsymbol{z}_k \in \mathbb{R}^m$ the observation or measurement vector of this process, given by

$$\boldsymbol{z}_k = \mathbf{H}_k\boldsymbol{x}_k + \boldsymbol{v}_k\,, \tag{22}$$

where the index $k \in \mathbb{N}^0$ denotes discrete time normalised to the sampling interval. The $n \times 1$ vector $\boldsymbol{w}_k$ and the $m \times 1$ vector $\boldsymbol{v}_k$ represent the process noise and the measurement noise, respectively, modelled as zero-mean, Gaussian white noise

$$\boldsymbol{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)\,, \tag{23}$$

$$\boldsymbol{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)\,, \tag{24}$$

with the process noise covariance matrix $\mathbf{Q}_k$ and the measurement noise covariance matrix $\mathbf{R}_k$. The $n \times n$ state transition matrix $\boldsymbol{\Phi}_{k-1}$ in Equation 21 relates the state at the previous time step $k-1$ to the state at the current step k. The $n \times l$ matrix $\mathbf{B}_{k-1}$ relates the known, optional control input $\mathbf{u}_{k-1} \in \mathbb{R}^l$ to the state $\boldsymbol{x}_k$. Finally, the $m \times n$ measurement matrix $\mathbf{H}_k$ in Equation 22 relates the state $\boldsymbol{x}_k$ to the measurement $\boldsymbol{z}_k$. Both noise processes are assumed to be uncorrelated. The process noise might not always have a physical meaning. However, it represents the fact that the model of the real world is not precise. The process and measurement noise covariance matrices are related to the respective noise vectors according to

$$\mathbf{Q}_k = \mathrm{E}[\boldsymbol{w}_k \boldsymbol{w}_k^\mathrm{T}]\,, \tag{25}$$

$$\mathbf{R}_k = \mathrm{E}[\boldsymbol{v}_k \boldsymbol{v}_k^\mathrm{T}]\,, \tag{26}$$

where E denotes the expected value.

The Kalman filter solves the problem of estimating the state $\boldsymbol{x}_k$ of the given linear stochastic system, minimising the weighted mean-squared error. This problem is called the *linear quadratic Gaussian* estimation problem; the dynamic system is linear, the performance cost function is quadratic, and the random process is Gaussian.

We define the vector $\hat{\mathbf{x}}_k^- \in \mathbb{R}^n$ as the *a priori* state estimate representing knowledge of the process prior to step k and $\hat{\mathbf{x}}_k \in \mathbb{R}^n$ as the *a posteriori* state estimate at step k given the measurement $\boldsymbol{z}_k$:

$$\hat{\mathbf{x}}_k^- = \boldsymbol{\Phi}_{k-1} \hat{\mathbf{x}}_{k-1} + \mathbf{B}_{k-1} \mathbf{u}_{k-1}\,, \tag{27}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k [\boldsymbol{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-]\,. \tag{28}$$

The term $[\boldsymbol{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-]$ is called the measurement *innovation* or *residual*. It reflects the discordance between the predicted measurement $\mathbf{H}_k \hat{\mathbf{x}}_k^-$ and the actual measurement $\boldsymbol{z}_k$. The $n \times m$ matrix $\mathbf{K}_k$ is termed the Kalman gain and is given by

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\mathrm{T} [\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\mathrm{T} + \mathbf{R}_k]^{-1}\,, \tag{29}$$

with

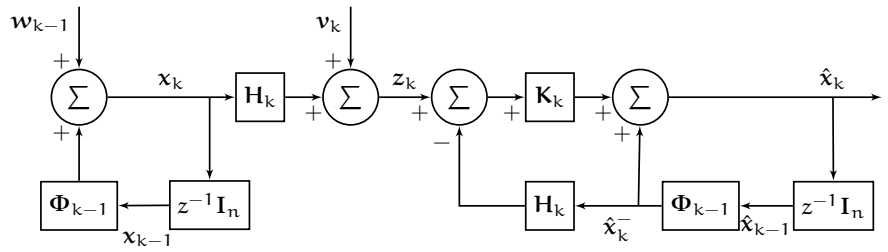

Figure 12: Block diagram depicting the relation between a discrete-time dynamical system, its observation, and the Kalman filter.

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}\mathbf{\Phi}_{k-1}^T + \mathbf{Q}_{k-1} \tag{30}$$

and

$$\mathbf{P}_k = [\mathbf{I}_n - \mathbf{K}_k\mathbf{H}_k]\mathbf{P}_k^- \,. \tag{31}$$

Figure 12 illustrates the relation of the Kalman filter to the discrete-time dynamical system with $\mathbf{u}_k = 0$, for the sake of simplicity, whereby z^{-1} denotes the unit-delay and $\mathbf{I}_n$ the $n \times n$ identity matrix.

The Kalman filter equations can be divided into two groups: *time update* Equations 27, 30 and *measurement update* Equations 28 , 29, and 31, as seen in Figure 13, which shows the 'predict and correct' behaviour of the filter algorithm. After an initialisation of the parameters, the *time update* and *measurement update* steps are repeated recursively every time step.

3.4.3 *The Extended Kalman Filter*

Up to this point the Kalman filter has solved the filtering problem for *linear* time-dynamical systems. One may extend the Kalman filter to systems with state dynamics governed by *non-linear* state transformations

$$\mathbf{x}_k = \boldsymbol{\phi}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \boldsymbol{w}_{k-1}, \quad \boldsymbol{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)\,, \tag{32}$$

and/or a *non-linear* transformation from state variables to measurement variables

$$\boldsymbol{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \boldsymbol{v}_k, \quad \boldsymbol{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)\,. \tag{33}$$

The *functional* $\boldsymbol{\phi}_{k-1}$ denotes the *non-linear* transition matrix function that may be time varying. It relates the state at the previous time step $k-1$ to the current time step k, depending on the exogenous control input $\mathbf{u}_{k-1}$. The functional $\mathbf{h}_k$ denotes a *non-linear* measurement matrix function that relates the state $\mathbf{x}_k$ to the measurement $\boldsymbol{z}_k$ and is possibly time varying, too.

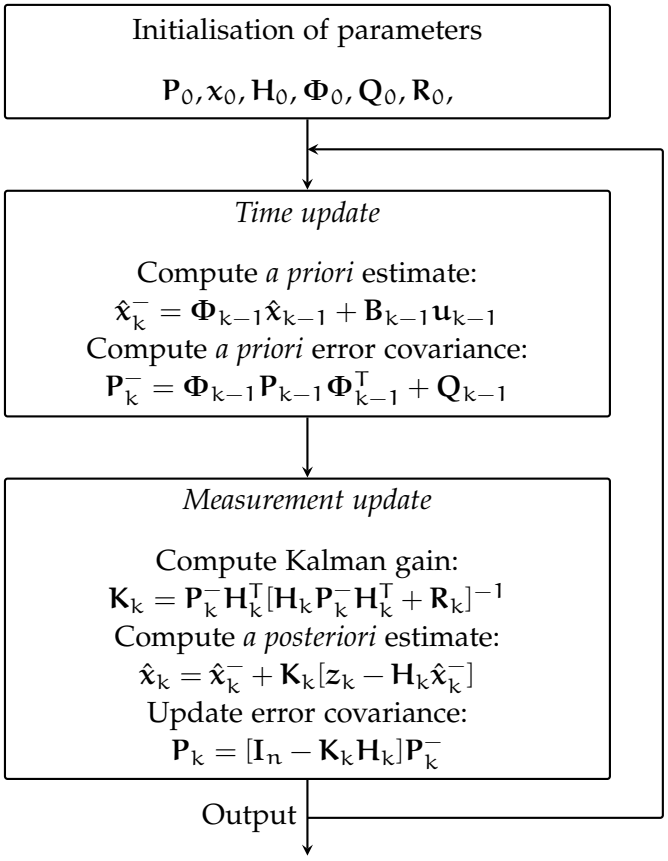

Figure 13: Operation cycle of the Kalman filter algorithm illustrating 'predict and correct' behaviour.

Some non-linear problems can be deemed *quasilinear*, which means that the variation of the non-linear functionals $\boldsymbol{\phi}_k$ and $\mathbf{h}_k$ are predominantly linear about the value $\mathbf{x}_0$. That is,

$$\boldsymbol{\phi}_k(\mathbf{x}_0 + d\mathbf{x}, \mathbf{u}) \approx \boldsymbol{\phi}_k(\mathbf{x}_0, \mathbf{u}) + d\mathbf{x} \left.\frac{\partial \boldsymbol{\phi}_k(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}, \tag{34}$$

$$\mathbf{h}_k(\mathbf{x}_0 + d\mathbf{x}) \approx \mathbf{h}_k(\mathbf{x}_0) + d\mathbf{x} \left.\frac{\partial \mathbf{h}_k(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}, \tag{35}$$

which requires that $\boldsymbol{\phi}_k$ and $\mathbf{h}_k$ are differentiable at $\mathbf{x}_0$.

Through a *linearisation* of the state-space model of Equations 32 and 33 at each time instant around the most recent state estimate, the standard Kalman filter equation from Section 3.4.2 can be applied. The filter resulting from a *linear approximation* of the state transitions and the relation of the measurement to the respective state is referred to as the *extended Kalman filter*.

The linearisation of the functionals $\boldsymbol{\phi}_k$ and $\mathbf{h}_k$ is given by their respective Jacobian matrices

$$\mathbf{\Phi}_{k-1}^{[1]} = \left.\frac{\partial \boldsymbol{\phi}_{k-1}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}, \mathbf{u}=\mathbf{u}_{k-1}}, \tag{36}$$

and

$$\mathbf{H}_k^{[1]} = \left.\frac{\partial \mathbf{h}_k(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k} . \tag{37}$$

The ij-th entry of $\mathbf{\Phi}_{k-1}^{[1]}$ is equal to the partial derivative of the i-th component of $\boldsymbol{\phi}_{k-1}(\mathbf{x})$ with respect to the j-th component of $\mathbf{x}$. The derivatives are evaluated at $\mathbf{x} = \hat{\mathbf{x}}_{k-1}, \mathbf{u} = \mathbf{u}_{k-1}$. Likewise, the ij-th entry of $\mathbf{H}_k^{[1]}$ is equal to the partial derivative of the i-th component of $\mathbf{h}_k(\mathbf{x})$ with respect to the j-th component of $\mathbf{x}$. The derivatives are evaluated at $\mathbf{x} = \hat{\mathbf{x}}_k$. The superscript $^{[1]}$ denotes the *first-order* approximation.

Similar to Equation 27, the predicted state estimate is given by

$$\hat{\mathbf{x}}_k^- = \boldsymbol{\phi}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) , \tag{38}$$

and the predicted measurement by

$$\hat{\mathbf{z}}_k = \mathbf{h}_k(\hat{\mathbf{x}}_k^-) . \tag{39}$$

The *a posteriori* estimate is then, conditioned on the actual measurement,

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k[\mathbf{z}_k - \hat{\mathbf{z}}_k] . \tag{40}$$

The corresponding *a priori* covariance matrix $\mathbf{P}_k^-$, the Kalman gain $\mathbf{K}_k$, and the *a posteriori* covariance matrix $\mathbf{P}_k$ are equal to Equations 29, 30, and 31 in Section 3.4.2. They are reproduced here with the linear approximation of the state transition and measurement matrices for convenience of presentation:

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}^{[1]} \mathbf{P}_{k-1} \mathbf{\Phi}_{k-1}^{[1]\mathsf{T}} + \mathbf{Q}_{k-1} , \tag{41}$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^{[1]\mathsf{T}} [\mathbf{H}_k^{[1]} \mathbf{P}_k^- \mathbf{H}_k^{[1]\mathsf{T}} + \mathbf{R}_k]^{-1} , \tag{42}$$

$$\mathbf{P}_k = [\mathbf{I}_n - \mathbf{K}_k \mathbf{H}_k^{[1]}] \mathbf{P}_k^- . \tag{43}$$

Figure 14 illustrates the 'predict and correct' behaviour of the extended Kalman filter algorithm. The *time update* and *measurement update* steps are repeated recursively every time step, after an initialisation of the parameters, similar to the classical Kalman filter algorithm. In addition, the Jacobian matrices have to be computed, in order to linearise the state-space model at each time instant around the most recent state estimate.

Extended Kalman filtering is commonly used. In fact, it was the first successful application of the Kalman filter [38]. Unlike its linear counterpart, the extended Kalman filter may not necessarily be an optimal estimator. Owing to its linearisation the EKF may quickly diverge, if the process is modelled incorrectly or the initial state estimate is too imprecise.

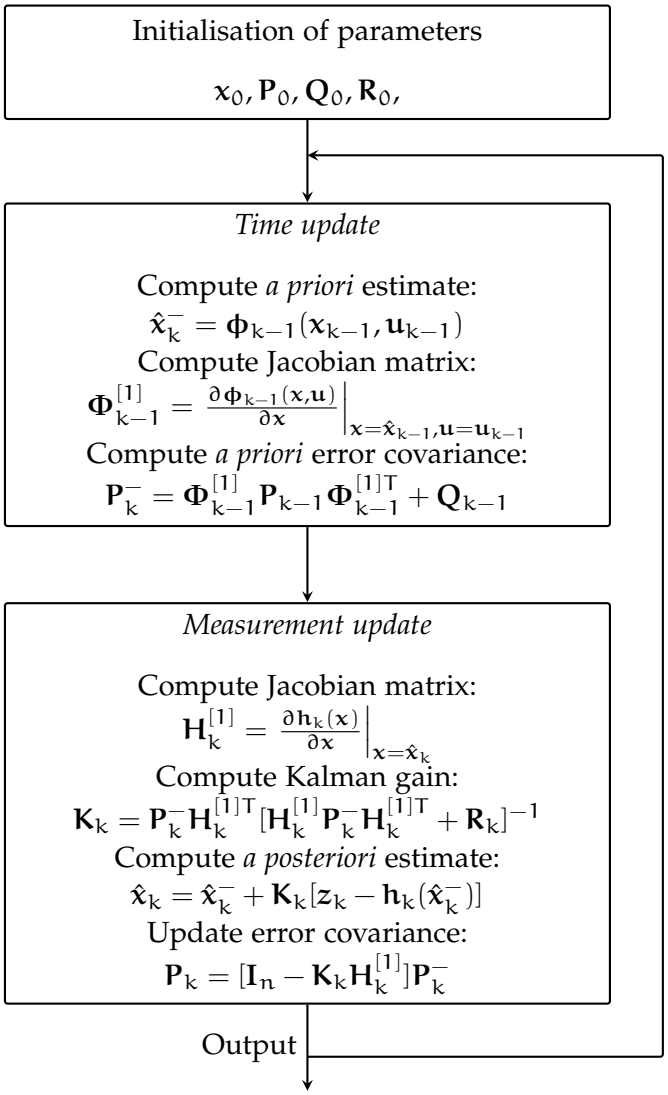

Figure 14: Operation cycle of the extended Kalman filter algorithm illustrating 'predict and correct' behaviour.

4 IMPLEMENTATION

This chapter describes the implementation of the filter algorithm proposed by Bennett, Jafari and Gans in [7], based on the fundamentals acquired in the previous chapters. Other than in [7], in which the filter algorithm was tested moving a mechanical model of the leg by hand, we used movement data from a human subject. After outlining the initial situation, i. e. describing the GaitWatch system, including orientation algorithms that were already implemented, the theoretical design of the filter is elaborated in detail. Subsequently, the software implementation and experiments are presented, followed by the results and their discussion.

4.1 INITIAL SITUATION

4.1.1 *The GaitWatch System*

As indicated above, to gather and preprocess movement data from the subject, we used a system called GaitWatch [39], which was designed to monitor the motion of patients by means of inertial sensors attached to the body. It was developed at the Department of Neurology of the Ludwig-Maximilians University in Munich, Germany, in association with the Department of Signal Theory, Telematics and Communications of the University of Granada, Spain.

4.1.1.1 *Hardware*

From the hardware perspective, the system is composed of a set of magnetic and inertial sensors wired to a box containing a microcontroller. This microcontroller is in charge of collecting data from the sensors embedded in the box, as well as from external measurement units, and storing them on a memory card. The separate units are placed on the patient's trunk, arms, thighs, and shanks as shown in Figure 15. The components of the three different kinds of subunits are listed below:

- TYPE A – THIGHS AND SHANKS:

  IMU Analog Combo Board with 5 Degrees of Freedom [40], containing an IDG500 biaxial gyroscope, from which only y-axis is actually used, with a measurement range of $\pm 500^\circ/\mathrm{s}$ [41] and a $\pm 3\,\mathrm{g}$ triaxial accelerometer, ADXL335 [42].

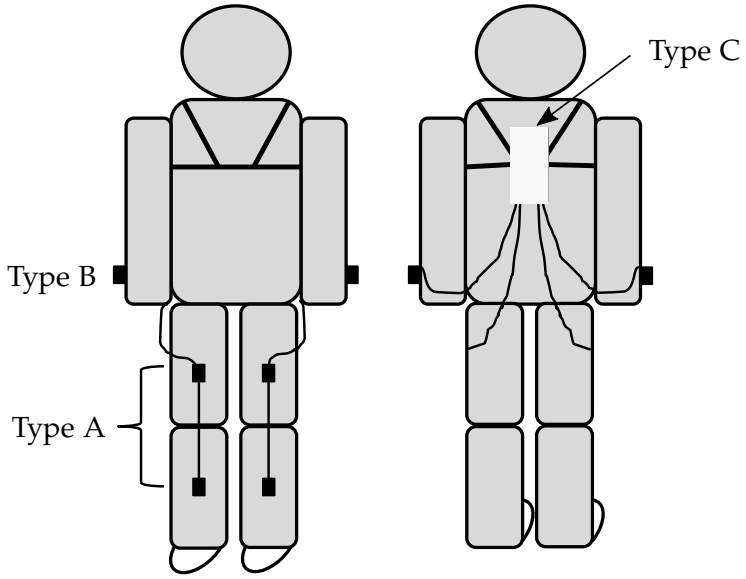

Figure 15: Placement of the GaitWatch components at the body, from [39].

- TYPE B – ARMS:

  IDG500 biaxial gyroscope with a measurement range of $\pm 500°/s$ [41].

- TYPE C – TRUNK:

  ADXL345 triaxial accelerometer with a programmable measurement range of $\pm 2/\pm 4/\pm 8/\pm 16\,g$ [43], IMU3000 triaxial gyroscope with a programmable measurement range of $\pm 250/\pm 500/\pm 1000/\pm 3000°/s$ [44], Micromag3 triaxial magnetometer with a measurement range of ± 11 Gauss [45], AL-XAVRB board containing an AVR ATxmega processor [46].

4.1.1.2 *Software*

In addition to the hardware, there was an existing MATLAB® toolbox consisting of routines for reading the data and carrying out the necessary calibration. Also, several algorithms that determine the motion intensity and compute the orientation of the human body from the movement data were already implemented. There were three algorithms for estimating the pitch angle of the thighs and shanks, as described below:

- PROJECTION OF THE GRAVITY VECTOR: One way to obtain the pitch angle from the inertial data is using the projection of the gravity vector on the axes of the accelerometer, as described in Section 2.4. The first algorithm computed the pitch angle according to Equation 5.

- INTEGRATION OF THE ANGULAR RATE: Another way to obtain the pitch angle is the integration of the angular rate, as outlined in Section 2.5. There are many different numerical integration procedures. The existing algorithm used the approximation of the integral according to the trapezoidal rule

$$\theta(n) = \theta_0 + \int_0^{nT_s} \omega(t)\,dt \approx \theta_0 + \frac{T_s}{2}\sum_{k=1}^{n}\left[\omega_{k-1} + \omega_k\right], \quad (44)$$

  where $n, k \in \mathbb{N}$ denote time normalised to the sample period T_s, ω_k the measured angular rate at instant k, and θ_0 the angle at $k = 0$. The implemented algorithm computed the angle recursively as

$$\theta(n) \approx \theta(n-1) + \frac{T_s}{2}\left[\omega_{k-1} + \omega_k\right], \quad \theta(0) = \theta_0\,. \quad (45)$$

  The initial value θ_0 can be computed from the projection of the gravity vector, assuming that the patient stands still when the records are started. Figure 16 shows exemplary the result of the first two algorithms applied to estimate the shank angle with respect to the x-axis, according to the mechanical model of the leg depicted in Figure 19. As can be seen in Figure 16, the accelerometer-based approach does not suffer from drift but delivers a very poor angle estimate during periods of motion, especially with increasing motion intensity. On the other hand, integrating the angular rate delivers an accurate angle estimate during motion, but suffers from drift over time, which accounts for an error in the estimate of more than 500° in only eighteen seconds for this exemplary signal.

- KALMAN FILTER: The third algorithm fused the orientation angle based on the projection of the gravity vector with the angle based on the integration of the angular rate measured with the gyroscope in a classical Kalman filter, without taking the motion intensity into account [39]. The angles of the thigh and shank are estimated independently. The result in comparison with the projection of the gravity vector alone is depicted in Figure 17.

In tandem with the orientation angles estimated with the algorithms mentioned above, an angle estimate based on a Qualisys® motion capture system served as a reference. The Qualisys® uses high-speed cameras in combination with optical markers placed on the legs, as depicted in Figure 18. From the recorded trace of the markers in space the reference angles of the thighs and shanks could be computed using an existing algorithm.

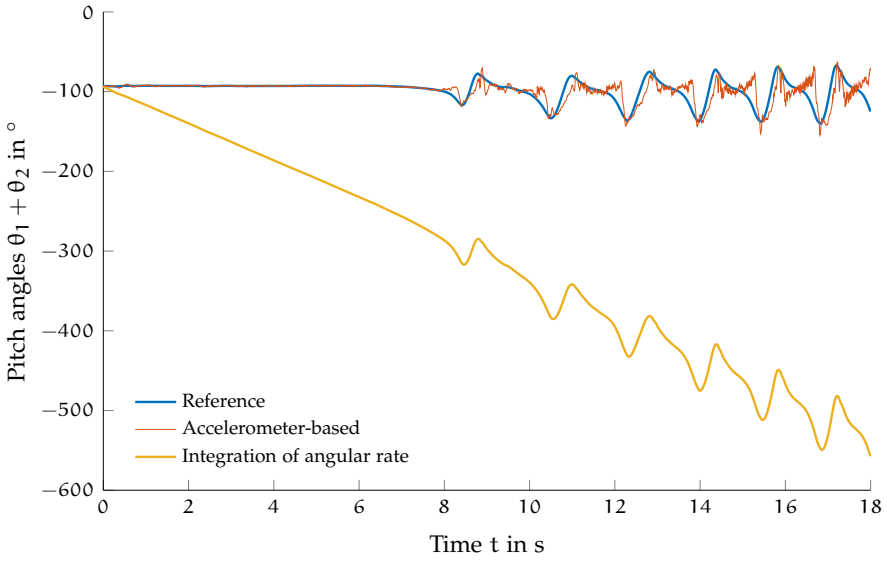

Figure 16: Pitch angle of the right shank with respect to the x-axis, obtained by the projection of the gravity vector and by integrating the angular rate, in comparison to the reference.

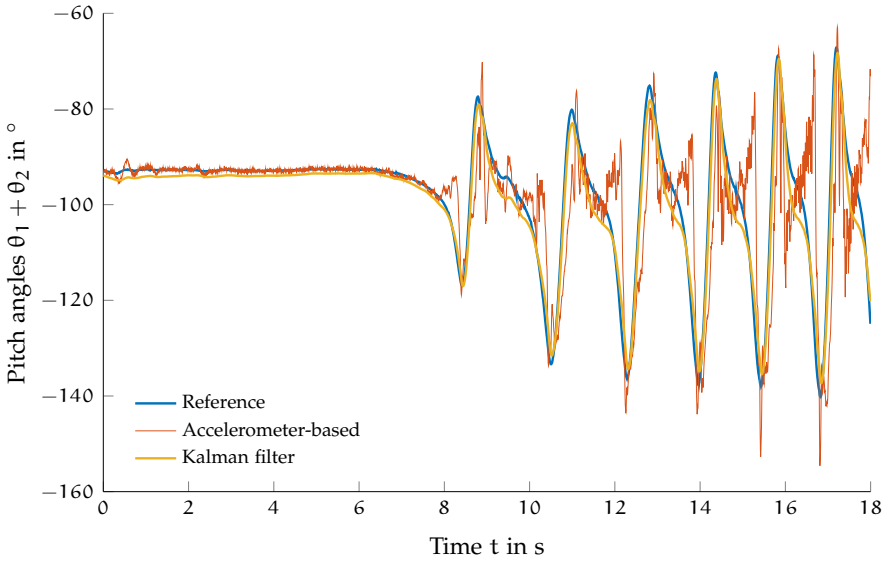

Figure 17: Pitch angle of the right shank with respect to the x-axis, obtained by the projection of the gravity vector and by sensor fusion of accelerometer and gyroscope data in a classical Kalman filter, in comparison to the reference.

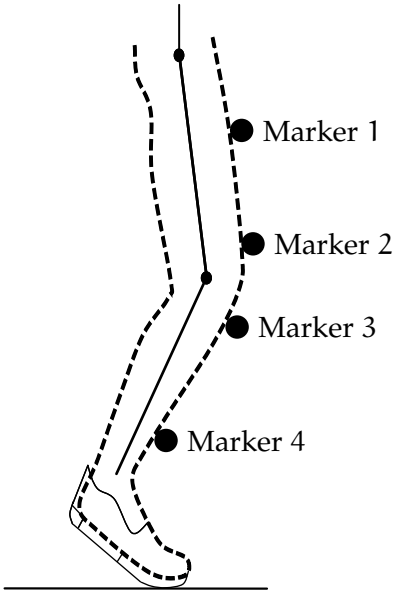

Figure 18: Human leg with optical markers, from [1].

4.2 THEORETICAL DESIGN

This section maps the theoretical design of the system proposed by Bennett, Jafari and Gans in [7] to the existing GaitWatch system. It states the assigned coordinate frames and the conventions regarding rotations about their axes. Furthermore, it presents the kinematic model used to improve the angle estimates and the extended Kalman filter algorithm with its underlying state-space model in detail.

4.2.1 *Kinematic Model*

The kinematic model relates the respective angles of the thigh and shank about the hip and knee joint to the acceleration seen by the wearable sensors. When walking in a straight line, the human leg can be modelled as a two-link planar revolute robot [7]. Then, thighs and shanks remain in a single plane, which is approximately parallel to the direction of motion. As depicted in Figure 19, the revolute joints of the pendulum robot represent the hip and knee joint, and the two links the thigh and shank, respectively. The origin of the inertial world frame is located at the base of link 1, the upper of both links. The x-axis points forward, the y-axis points out from the hip to the right, and the z-axis points down. This configuration follows the right-hand rule, which can also be used to determine the sense of rotation around the axes. The pitch angle θ_1 is measured with respect to the x-axis, and the pitch angle θ_2 of link 2 with respect to link 1.

The IMU placed on the thighs and shanks measured the angular velocity around the y-axis and the linear acceleration along the x and z-axis, respectively. According to Spong and Hutchinson [47], the

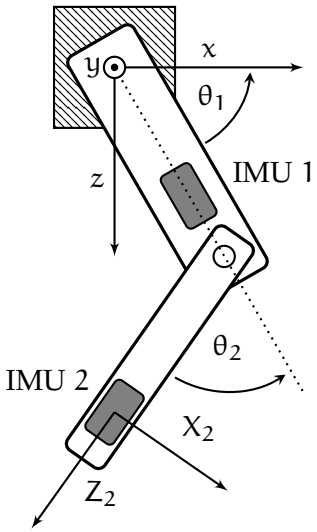

Figure 19: Kinematic model of the human leg, from [7].

x and z-displacement and its derivatives in the world frame are as follows:

$$x = +l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \tag{46}$$

$$\dot{x} = -l_1 \dot{\theta}_1 \sin(\theta_1) - l_2(\dot{\theta}_1 + \dot{\theta}_2) \sin(\theta_1 + \theta_2) \tag{47}$$

$$\ddot{x} = -l_1[\dot{\theta}_1^2 \cos(\theta_1) + \ddot{\theta}_1 \sin(\theta_1)] - l_2[(\dot{\theta}_1 + \dot{\theta}_2)^2 \cos(\theta_1 + \theta_2) + (\ddot{\theta}_1 + \ddot{\theta}_2) \sin(\theta_1 + \theta_2)] \tag{48}$$

$$z = -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) \tag{49}$$

$$\dot{z} = -l_1 \dot{\theta}_1 \cos(\theta_1) - l_2(\dot{\theta}_1 + \dot{\theta}_2) \cos(\theta_1 + \theta_2) \tag{50}$$

$$\ddot{z} = -l_1[\ddot{\theta}_1 \cos(\theta_1) - \dot{\theta}_1^2 \sin(\theta_1)] - l_2[(\ddot{\theta}_1 + \ddot{\theta}_2) \cos(\theta_1 + \theta_2) - (\dot{\theta}_1 + \dot{\theta}_2)^2 \sin(\theta_1 + \theta_2)] \tag{51}$$

in which l_1 and l_2 are the lengths of the two links, respectively. Plugging the *a priori* estimates of the angles θ_1 and θ_2 and their derivatives, i. e. the angular rates ω_1 and ω_2, and the angular accelerations α_1 and α_2, obtained with the EKF described in Section 4.2.2, into Equations 48 and 51, we can estimate the respective motion-based acceleration components a_x and a_z in x and z-direction that sensor 2 will see in the world coordinate frame. Written as a function of the state variables of the extended Kalman filter, Equations 48 and 51 yield

$$a_x = -l_1[\omega_1^2 \cos(\theta_1) + \alpha_1 \sin(\theta_1)] - l_2[(\omega_1 + \omega_2)^2 \cos(\theta_1 + \theta_2) + (\alpha_1 + \alpha_2) \sin(\theta_1 + \theta_2)] \tag{52}$$

$$a_z = -l_1[\alpha_1 \cos(\theta_1) - \omega_1^2 \sin(\theta_1)] - l_2[(\alpha_1 + \alpha_2) \cos(\theta_1 + \theta_2) - (\omega_1 + \omega_2)^2 \sin(\theta_1 + \theta_2)] \tag{53}$$

The orientation of the sensor frames at rest are different from the world frame and dynamic when the pendulum is in motion. In order to transform the values from the world frame to the dynamic body frame of IMU 2, which is depicted in Figure 19, we used the transformation matrix $\mathbf{T}_y(\theta)$ from Equation 3. The body frame of sensor two is not aligned with the world frame for $\theta_1 = \theta_2 = 0$. Thus, in order to align both frames, an offset of 90° is required. With $\theta = \theta_1 + \theta_2 + 90°$, this yields

$$\begin{aligned} \mathbf{T}_y(\theta_1 + \theta_2 + 90°) &= \begin{bmatrix} \cos(\theta_1+\theta_2+90°) & 0 & -\sin(\theta_1+\theta_2+90°) \\ 0 & 1 & 0 \\ \sin(\theta_1+\theta_2+90°) & 0 & \cos(\theta_1+\theta_2+90°) \end{bmatrix} \\ &= \begin{bmatrix} -\sin(\theta_1+\theta_2) & 0 & -\cos(\theta_1+\theta_2) \\ 0 & 1 & 0 \\ \cos(\theta_1+\theta_2) & 0 & -\sin(\theta_1+\theta_2) \end{bmatrix}. \end{aligned} \tag{54}$$

According to Equation 2, the rotated radial and tangential components of the motion-based acceleration estimates in the body frame of accelerometer 2, A_{X_2} and A_{Z_2}, are found by pre-multiplying the transformation matrix with the acceleration vector in the world frame, constituted of the results of Equations 52 and 53.

$$\mathbf{a} = \begin{bmatrix} a_{X_2} \\ a_{Y_2} \\ a_{Z_2} \end{bmatrix} = \mathbf{T}_y(\theta_1 + \theta_2 + 90°) \begin{bmatrix} a_x \\ 0 \\ a_z \end{bmatrix} \|\mathbf{g}\|^{-1} \tag{55}$$

The axial component A_{Y_2} is zero, since the leg is assumed to be oriented perpendicular to the earth's surface and thus the gravity vector $\mathbf{g}$ is perpendicular to the y-axis. The term $\|\mathbf{g}\|^{-1}$ normalises the motion-based acceleration estimate to gravity, where $\|\mathbf{g}\|$ denotes the magnitude of gravity.

Then, the motion based radial and tangential acceleration components are subtracted from the sensor readings $a_{X_1 m}$ and $a_{Z_1 m}$, which leaves an estimate of the gravity based acceleration $\mathbf{g}$ that acts on the sensor:

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \approx \begin{bmatrix} a_{X_2 m} \\ 0 \\ a_{Z_2 m} \end{bmatrix} - \begin{bmatrix} a_{X_2} \\ 0 \\ a_{Z_2} \end{bmatrix}. \tag{56}$$

The angle estimate of the shank with respect to the x-axis, based on the projection of the gravity vector on the axes of the accelerometer, is then

$$\theta_1 + \theta_2 = \text{atan2}(g_z, g_x) - 180^\circ . \tag{57}$$

This improved angle estimate is then fused with the estimate based on the integration of the angular rate measured with the gyroscope, in order to reduce the estimation error due to gyroscope drift.

4.2.2 *Extended Kalman Filter*

4.2.2.1 *The State-Space Model*

The state-space model of the extended Kalman filter is given by the state vector $\mathbf{x} \in \mathbb{R}^n$, with $n = 10$,

$$\mathbf{x} = \begin{bmatrix} x, & z, & \theta_1, & \omega_1, & \alpha_1, & \theta_2, & \omega_2, & \alpha_2, & \beta_1, & \beta_2 \end{bmatrix}^T , \tag{58}$$

where x and y correspond to the horizontal and vertical position of the end of link 2 with respect to the origin of the world frame, i. e. the hip joint. θ_1 is the angle, ω_1 the angular velocity, and α_1 the angular acceleration of the first joint, respectively. The corresponding values for the second link are θ_2, ω_2, and α_2. The biases of the gyroscopes in the first and the second IMU are β_1 and β_2, respectively. They are assumed to be constant or slowly time-varying.

The measurement vector $\mathbf{z} \in \mathbb{R}^m$, with $m = 4$, is given by

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} \omega_1 + \beta_1, & \omega_1 + \omega_2 + \beta_1 + \beta_2, & \theta_1, & \theta_1 + \theta_2 \end{bmatrix}^T + \boldsymbol{\nu}, \tag{59}$$

where $\boldsymbol{\nu}$ is the random measurement noise process, modelled as zero-mean, Gaussian white noise. The element z_1 represents the measurement of the first link angular velocity, which is the sum of the first link rotation and the gyroscope 1 bias. The element z_2 represents the measurement of the second link angular velocity, which is the sum of the first and second link rotation and the bias of gyroscope 1 and gyroscope 2. Finally, the element z_3 is the angle estimate of the first accelerometer and the element z_4 the angle estimate of the second accelerometer, which will see the angular displacement of both links.

According to Rowell [48], the plant dynamics of a system can be expressed as a set of n coupled first-order ordinary differential equations, known as the *state equations*. The modelled system is governed by the *non-linear* first-order ordinary differential equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) + \boldsymbol{w} = \begin{bmatrix} -l_1\omega_1\sin(\theta_1) - l_2(\omega_1+\omega_2)\sin(\theta_1+\theta_2) \\ -l_1\omega_1\cos(\theta_1) - l_2(\omega_1+\omega_2)\cos(\theta_1+\theta_2) \\ \omega_1 \\ \alpha_1 \\ 0 \\ \omega_2 \\ \alpha_2 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{w}\,, \quad (60)$$

where $\dot{\mathbf{x}}$ consists of the component-wise time derivatives of the state vector $\mathbf{x}$, expressed in terms of the state variables $x_1(t), \ldots, x_n(t)$. Its elements are given by

$$\dot{x}_i = f_i(x_1(t), \ldots, x_n(t), t) + w_i = \frac{dx_i}{dt} + w_i\,, \quad i = 1, \ldots, n\,. \quad (61)$$

The noise term $\boldsymbol{w}$, modelled as zero-mean, Gaussian white noise again represents the uncertainty in the model. Given this *state-space representation*, the system state at any instant may be interpreted as a point in an n-dimensional state space whose axes are the state variables. The dynamic state response $\mathbf{x}(t)$ can be interpreted as a trajectory traced out in the state space. The system described by Equation 60 is *time-invariant*, since it does not depend explicitly on time. Thus, we may leave out the t and write from now on $\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}, t)$.

For a *linear* system in state-space form given by

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x}\,, \quad (62)$$

with a time-invariant *system dynamics matrix* $\mathbf{F}$ there is a *state transition matrix* $\boldsymbol{\Phi}(t - t_0)$ that propagates the state of the system forward from any time t_0 to a time t, according to

$$\mathbf{x}(t) = \boldsymbol{\Phi}(t - t_0)\mathbf{x}(t_0)\,. \quad (63)$$

The solution to the system described by Equation 62 is

$$\mathbf{x}(t) = e^{\mathbf{F}(t-t_0)}\mathbf{x}(t_0)\,, \quad (64)$$

where $\mathbf{x}(t_0)$ is an integration constant. As outlined in [37], the state transition matrix can be found by a Taylor-series expansion of $e^{\mathbf{F}(t-t_0)}$,

$$\begin{aligned} \boldsymbol{\Phi}(t - t_0) = e^{\mathbf{F}(t-t_0)} &= \sum_{k=0}^{\infty} \frac{\mathbf{F}^n[t - t_0]^n}{k!} \\ &= \mathbf{I}_n + \mathbf{F}[t - t_0] \\ &\quad + \frac{\mathbf{F}^2[t - t_0]^2}{2!} + \frac{\mathbf{F}^3[t - t_0]^3}{3!} + \cdots\,, \end{aligned} \quad (65)$$

where $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix. Truncating the Taylor series after the first order terms yields the *linear approximation* of the fundamental matrix:

$$\boldsymbol{\Phi}(t - t_0) \approx \mathbf{I}_n + \mathbf{F}[t - t_0]\,. \tag{66}$$

The discrete fundamental matrix that propagates the state of the system forward from time step k to $k+1$ can be found by substituting T_s for $t - t_0$, which yields

$$\boldsymbol{\Phi}_k = \boldsymbol{\Phi}(T_s) \approx \mathbf{I}_n + \mathbf{F}T_s\,, \tag{67}$$

where T_s is the sampling period. In the *linear* case $\boldsymbol{\Phi}_k$ is constant.

Because the state equations of our system are *non-linear*, a first-order approximation of the system dynamics matrix $\mathbf{F}$ is used, given by the Jacobian of $\mathbf{f}(\mathbf{x})$

$$\begin{aligned}
\mathbf{F}_k^{[1]} = \mathbf{J}_\mathbf{f} &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \\
&= \begin{bmatrix}
0 & 0 & A & C & 0 & E & G & 0 & 0 & 0 \\
0 & 0 & B & D & 0 & F & H & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}_{\mathbf{x}=\hat{\mathbf{x}}_k}
\end{aligned}, \tag{68}$$

with

$$\begin{aligned}
A &= -l_1\omega_1\cos(\theta_1) - l_2(\omega_1+\omega_2)\cos(\theta_1+\theta_2)\,, \\
B &= +l_1\omega_1\sin(\theta_1) + l_2(\omega_1+\omega_2)\sin(\theta_1+\theta_2)\,, \\
C &= -l_1\sin(\theta_1) - l_2\sin(\theta_1+\theta_2)\,, \\
D &= -l_1\cos(\theta_1) - l_2\cos(\theta_1+\theta_2)\,, \\
E &= -l_2(\omega_1+\omega_2)\cos(\theta_1+\theta_2)\,, \\
F &= +l_2(\omega_1+\omega_2)\sin(\theta_1+\theta_2)\,, \\
G &= -l_2\sin(\theta_1+\theta_2)\,, \\
H &= -l_2\cos(\theta_1+\theta_2)\,.
\end{aligned}$$

The partial derivatives are evaluated at the state estimate $\hat{\mathbf{x}}_k$. The discrete state transition matrix must be recomputed every time step.

Here the subscript k denotes the state transition matrix that propagates the state at time step k to time step $k+1$. It is given by

$$\mathbf{\Phi}_k^{[1]} \approx \mathbf{I}_n + \mathbf{F}_k^{[1]} T_s \,. \tag{69}$$

4.2.2.2 *The Filter Algorithm*

The estimate $\hat{\mathbf{x}}_{k-1}$ can be propagated forward to the *a priori* estimate $\hat{\mathbf{x}}_k^-$ by integrating the *non-linear* differential equations at each sampling interval. Applying Euler integration Equation 38 yields

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \mathbf{\Phi}_{k-1}(\hat{\mathbf{x}}_{k-1}, 0) \\ &= \hat{\mathbf{x}}_{k-1} + \mathbf{f}(\hat{\mathbf{x}}_{k-1}) T_s \,, \end{aligned} \tag{70}$$

where T_s is the integration interval. The control input $\mathbf{u}_k$ is equal to zero since the system does not have any inputs. A higher-order numerical integration procedure would not improve the *a priori* estimate since the function $\mathbf{f}(\mathbf{x})$ is constant over time.

The relation between the states and the measurements is *linear*, according to Equation 22. The measurement matrix $\mathbf{H} \in \mathbb{R}^{3\times 10}$ is given by

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{71}$$

The process noise covariance matrix $\mathbf{Q} \in \mathbb{R}^{10\times 10}$ is constant and given by

$$\mathbf{Q} = \begin{bmatrix} \mathrm{Cov}(w_1, w_1) & \mathrm{Cov}(w_1, w_2) & \cdots & \mathrm{Cov}(w_1, w_n) \\ \mathrm{Cov}(w_2, w_1) & \mathrm{Cov}(w_2, w_2) & \cdots & \mathrm{Cov}(w_2, w_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(w_n, w_1) & \mathrm{Cov}(w_n, w_2) & \cdots & \mathrm{Cov}(w_n, w_n) \end{bmatrix}$$

$$= \begin{bmatrix} \sigma_d^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_d^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sigma_{\theta_1}^{18}}{9} & \frac{\sigma_{\theta_1}^{8}}{4} & \frac{\sigma_{\theta_1}^{10}}{5} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sigma_{\theta_1}^{8}}{4} & \frac{\sigma_{\theta_1}^{6}}{3} & \frac{\sigma_{\theta_1}^{4}}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sigma_{\theta 1}^{10}}{5} & \frac{\sigma_{\theta_1}^{4}}{2} & \sigma_{\theta_1}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\sigma_{\theta_2}^{18}}{9} & \frac{\sigma_{\theta_2}^{8}}{4} & \frac{\sigma_{\theta_2}^{10}}{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\sigma_{\theta_2}^{8}}{4} & \frac{\sigma_{\theta_2}^{6}}{3} & \frac{\sigma_{\theta_2}^{4}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\sigma_{\theta_2}^{10}}{5} & \frac{\sigma_{\theta_2}^{4}}{2} & \sigma_{\theta_2}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\beta^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\beta^2 \end{bmatrix}. \tag{72}$$

The diagonal elements represent the respective variances of the elements $w_1, \ldots, w_n$ of the process noise vector $\boldsymbol{w}$, due to the relation $\mathrm{Cov}(w_i, w_i) = \mathrm{Var}(w_i)$. The other elements are the covariances of all possible pairs of the random variables of the process noise vector. The noise processes interfering with the state variables x and y, and β_1 and β_2, respectively, were modelled as independent. In contrast, the covariances of the noise components interfering with the state variables $\theta_i, \omega_i, \alpha_i, i \in 1, 2$, which account for the block-diagonal structure, reflect a random walk process, that is, the integration of a signal perturbed by Gaussian white noise. A detailed derivation of the form of the elements is found in [49]. The parameters $\sigma_d, \sigma_{\theta_1}, \sigma_{\theta_2}$, and σ_β were found by an optimiser, which is described in Section 4.3.3.

The measurement noise covariance matrix $\mathbf{R} \in \mathbb{R}^{4\times 4}$, is given by

$$\mathbf{R} = \begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \sigma_3^2 & 0 \\ 0 & 0 & 0 & \sigma_4^2 \end{bmatrix}, \tag{73}$$

The parameters σ_1^2 and σ_2^2 are constant. They are determined by computing the sample variance of the measurement data during an initialisation stage of $T_{\text{init}} = 2\,\text{s}$ seconds, while the subject stands still.

The variance of a finite data set with n samples $x_1, x_2, \ldots, x_n$ is given by

$$\sigma^2 = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \mu)^2, \quad \text{where} \quad \mu = \frac{1}{n}\sum_{k=1}^{n} x_k\,, \tag{74}$$

with

$$n = T_{\text{init}} f_s\,. \tag{75}$$

The variances σ_3^2 and σ_4^2 of the accelerometer-based angle estimates were found by the optimiser and were set dynamically at each time step k, based on the motion intensity. It toggles between σ_s and σ_f, according to slow or fast motion, distinguished by a marker signal m_k:

$$\sigma_3^2 = \begin{cases} \sigma_{3s}^2 & m_k = 0 \\ \sigma_{3f}^2 & m_k = 1 \end{cases}. \tag{76}$$

$$\sigma_4^2 = \begin{cases} \sigma_{4s}^2 & m_k = 0 \\ \sigma_{4f}^2 & m_k = 1 \end{cases}. \tag{77}$$

In order to determine the marker signal we used the long term spectral detector LTSD developed in [39], which distinguishes between fast and slow motion by computing the long term spectral envelope of the signal. The output of the LTSD is a marker signal which value toggles between 1 and 0.

4.2.2.3 *Summary of the Entire Filter Algorithm*

The state estimates are computed recursively according to the *time update* Equations 41, 70 and the *measurement update* Equations 28, 29, and 31. The entire computation steps of the recursive filter algorithm are summarised in Figure 20. Listing 1 in the appendix shows the MATLAB® implementation of the filter function.

4.3 EXPERIMENTS

The extended Kalman filter was tested and its results were compared to the existing classical Kalman filter and the reference signal, by interrelating the RMSE of each method. The orientation algorithms were applied to movement data from a real subject. These data were gathered at the Department of Neurology of the Klinikum Großhadern in Munich, while the subject performed the following trial.

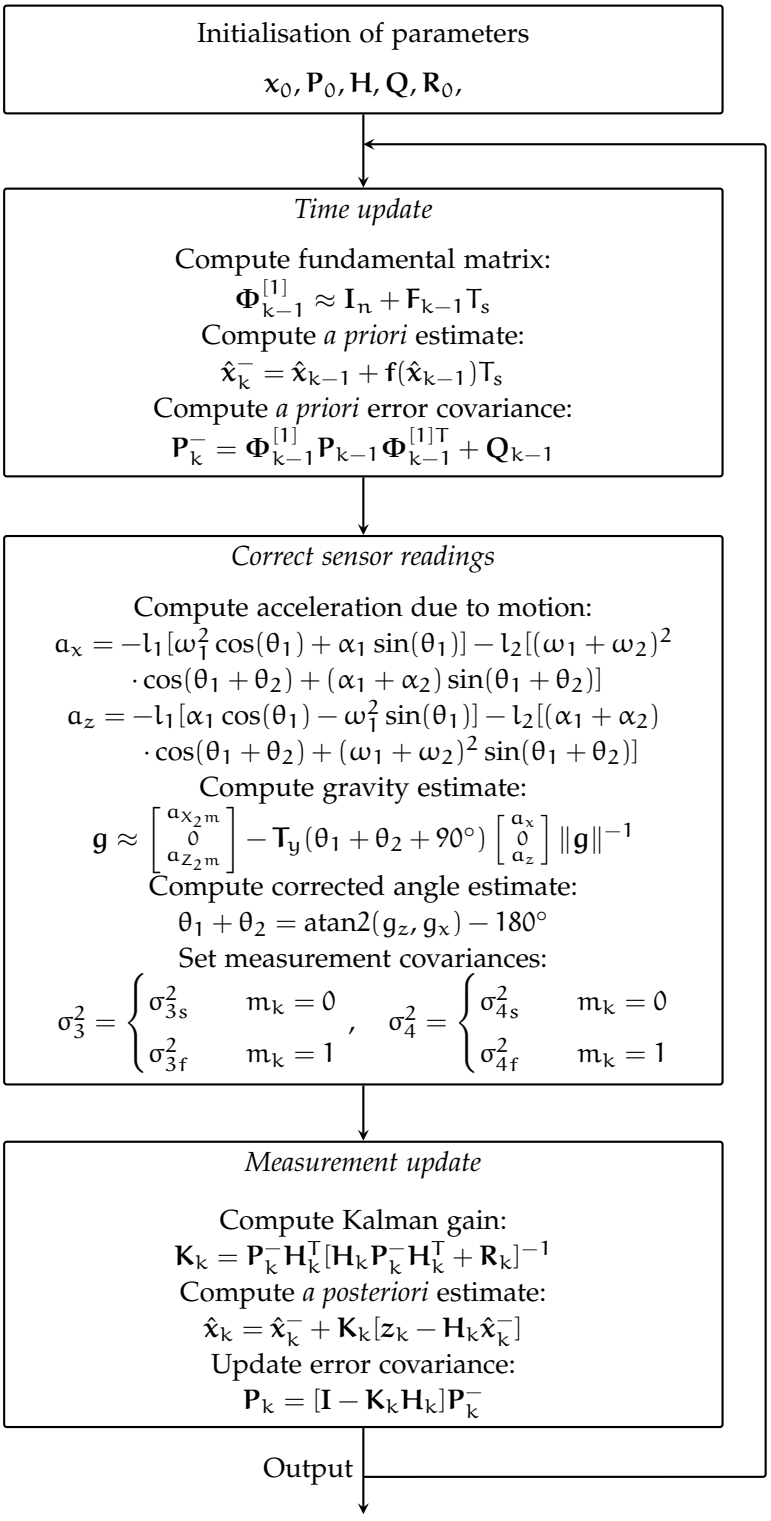

Figure 20: Entire computation steps of the recursive filter algorithm.

4.3.1 *Data Collection Protocol*

The subject stood on a treadmill wearing the GaitWatch system on its body. Then, the GaitWatch record was started and shortly afterwards the treadmill was turned on. After a variable time, depending on the treadmill speed and the walking distance, first the treadmill was switched off, and then the GaitWatch records. The trial was performed at walking speeds of $2\,\mathrm{km/h}$, $4\,\mathrm{km/h}$, and $6\,\mathrm{km/h}$.

4.3.2 *Initial Conditions*

Each trial began with the subject standing still and it was assumed that the subject's legs were fully stretched. This leads to the following initial state estimates:

$$\begin{aligned} x &= 0\,\mathrm{m}, & z &= -(l_1 + l_2)\,\mathrm{m}, \\ \theta_1 &= -90^\circ, & \theta_2 &= 0^\circ, \\ \omega_1 &= 0^\circ/\mathrm{s}, & \omega_2 &= 0^\circ/\mathrm{s}, \\ \beta_1 &= \mu_1\,{}^\circ/\mathrm{s}, & \beta_2 &= \mu_2\,{}^\circ/\mathrm{s}, \\ \alpha_1 &= 0^\circ/\mathrm{s}^2, & \alpha_2 &= 0^\circ/\mathrm{s}^2, \end{aligned} \tag{78}$$

where μ_1 and μ_2 are the mean values of the gyroscope signals collected during the initial $T_{\text{init}} = 2\,\mathrm{s}$ of the rest period before the subject started moving. The mean value is given by

$$\mu = \frac{1}{n}\sum_{k=1}^{n} \omega_k\,, \quad n = T_{\text{init}} f_s\,, \tag{79}$$

where ω_k is the angular velocity at instant k and f_s the sampling frequency.

The initial error covariance matrix was the identity matrix, $\mathbf{P}_0 = \mathbf{I}_{10} \in \mathbb{R}^{10\times 10}$. Its diagonal elements, that is, the variances of the initial state estimates, represent the confidence in the knowledge about the initial state of the system and is crucial for the convergence of the filter. Unlike the initial error covariance matrix of the classical Kalman filter, it may cause the filter to diverge, in case the confidence in the estimates is too small. Equally, too vague initial state estimates can cause the filter to diverge.

4.3.3 *Test Preparation*

To test the filter algorithm and compare it with the existing classical Kalman filter, a parameter optimiser was implemente. Based on the optimal set of parameters for a given signal, we compared the newly implemented filter algorithm against the classical Kalman filter. The parameter optimisation function given by Listing 2 called an adaptive Nelder-Mead simplex algorithm, which minimised the error func-

	2 m/s	4 m/s	6 m/s
σ_d^2	10.0000	10.0000	10.0000
$\sigma_{\theta_1}^2$	0.0514	0.0209	0.4342
$\sigma_{\theta_2}^2$	0.0514	0.0209	0.4342
σ_β^2	0.0011	0.0001	0.0007
σ_1^2	2.1767	3.6758	2.7102
σ_2^2	2.9788	3.2287	1.7409
σ_{3s}^2	11.5799	7.2881	5.6341
σ_{3f}^2	50.8859	117.2099	56.9627
σ_{4s}^2	45.1827	24.5344	88.4655
σ_{4f}^2	265.8502	224.4809	222.4782

Table 1: Filter parameters.

tion E. Since we are interested in an accurate angle estimate of both angles and thus need parameters that optimise both angle estimates, we scalarise this multi-objective optimisation problem. Using linear scalarisation yields

$$\begin{aligned} E(\hat{\theta}_t, \hat{\theta}_s) &= \mathrm{RMSE}_{\hat{\theta}_t} + \mathrm{RMSE}_{\hat{\theta}_s} \\ &= \sqrt{\frac{\sum_{k=1}^{n} (\hat{\theta}_{t,k} - \theta_{t,k})^2}{n}} + \sqrt{\frac{\sum_{k=1}^{n} (\hat{\theta}_{s,k} - \theta_{s,k})^2}{n}}, \end{aligned} \tag{80}$$

where $\hat{\theta}_{t,k}$ and $\hat{\theta}_{s,k}$ denote the thigh and shank angle at instant k, estimated by the extended Kalman filter, and $\theta_{t,k}$ and $\theta_{s,k}$ the reference angles at instant k, respectively. The implementation of the error function is shown in Listing 3. The optimiser returns the parameters that minimise the error function.

4.3.3.1 *Parameterisation*

The parameters for the three different walking speeds in Table 1 were determined by the optimisation routines. In order to improve the legibility, all parameters are normalised to their respective units. That is, since the variance is the square of the standard deviation, they are normalised to the square of the unit of the corresponding element of the state vector as stated in Section 4.3.2. For instance, the variance σ_d^2 of the displacement is normalised to m^2.

4.3.4 *Test Execution*

The testing procedure was as follows: First, the movement data of the subject were loaded into the workspace. Next, the parameters of the classical Kalman filter were optimised by an existing optimiser. Then, the parameters of the extended Kalman filter were optimised with the aforementioned EKF-optimiser. After each optimisation process the respective angle estimation was computed. Subsequently, the absolute RMSEs and the relative RMSE related to the one obtained with the classical Kalman filter were computed. Finally, the results were displayed in the console and depicted in four plots: The thigh and shank angle estimate, respectively, the RMSE comparison, and the acceleration correction. The MATLAB® code for the first data set is given in Listing 4. The results are presented in the next section.

4.4 RESULTS

The results of the experiments are presented as follows:

- THIGH ANGLES: Comparison of the temporal courses of the pitch angles of the right thigh with respect to the x-axis obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering, in comparison to the reference, for the three different walking speeds, depicted in Figures 21, 25, and 29.
- SHANK ANGLES: Comparison of the temporal courses of the pitch angles of the right shank with respect to the x-axis obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering, in comparison to the reference, for the three different walking speeds, depicted in Figures 22, 26, and 30.
- RMSE COMPARISON: Root-mean-square error comparison of angle estimations by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering, for the three different walking speeds, depicted in Figures 23, 27, and 31.
- ACCELERATION CORRECTION: Comparison of the accelerometer-based shank angles with respect to the x-axis with and without correction of the acceleration signal, for the three different walking speeds, depicted in Figures 24, 28, and 32.

The entire absolute and relative RMSEs are summarised in Table 2. This leads to an average improvement of the extended Kalman filter output with respect to the classical Kalman filter by a relative RMSE of 28.52% and an average improvement of the motion-corrected accelerometer based angle estimate by a relative RMSE of 8.96%.

	2 m/s		4 m/s		6 m/s	
	KF	EKF	KF	EKF	KF	EKF
$\mathrm{RMSE}_{\hat{\theta}_t}$ /°	1.5324	0.9472	2.3658	1.8721	10.1907	8.1939
$\mathrm{RMSE}_{\hat{\theta}_s}$ /°	4.4460	3.1816	3.3271	2.3736	14.5714	9.3381
SUM /°	5.9784	4.1287	5.6928	4.2457	24.7621	17.5321
$\frac{\mathrm{RMSE}_{Acc-corr}}{\mathrm{RMSE}_{Acc}}$	1.0182		0.9687		0.7444	
$\frac{\mathrm{SUM}_{EKF}}{\mathrm{SUM}_{KF}}$	0.6906		0.7458		0.7080	

Table 2: Root-mean-square errors of the Kalman filter and the extended Kalman filter.

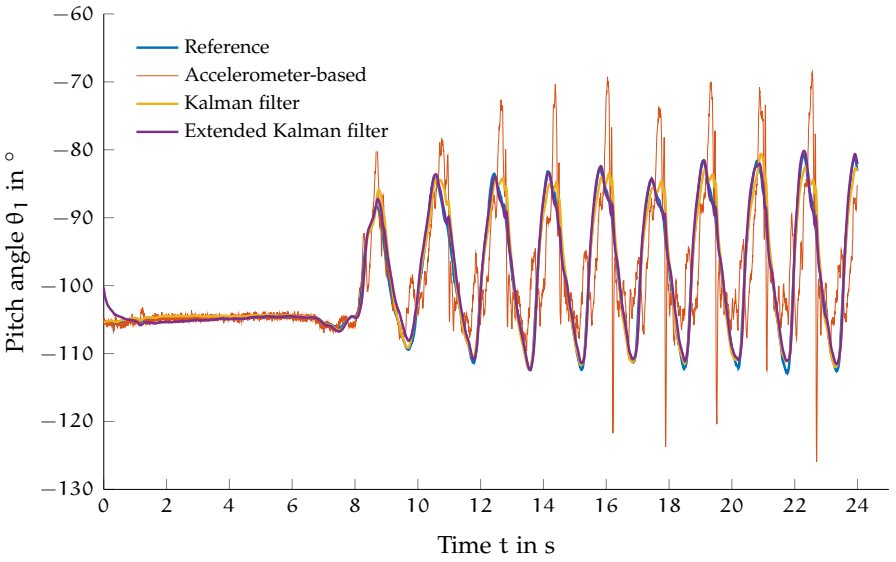

Figure 21: Pitch angle of the right thigh with respect to the x-axis, obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering, in comparison to the reference. Walking speed: 2 km/h.

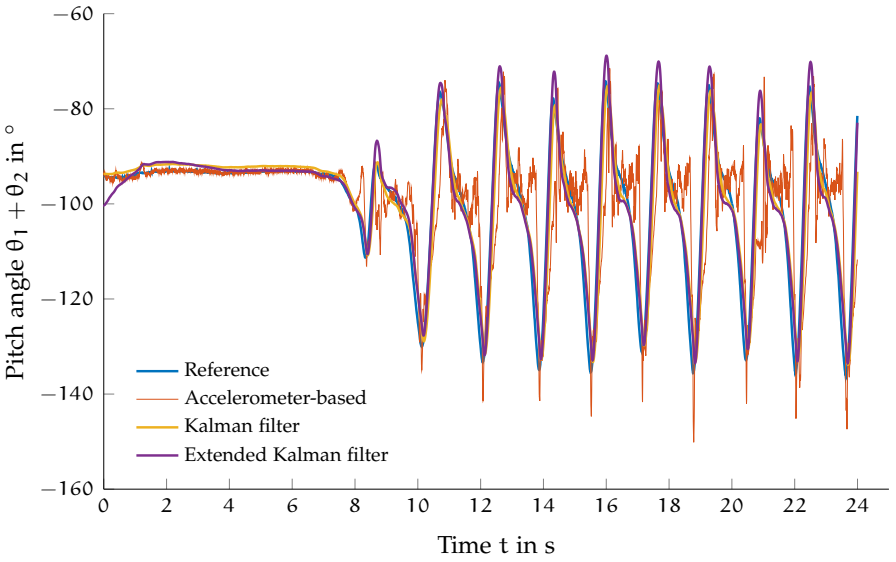

Figure 22: Pitch angle of the right shank with respect to the x-axis, obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering, in comparison to the reference. Walking speed: 2 km/h.

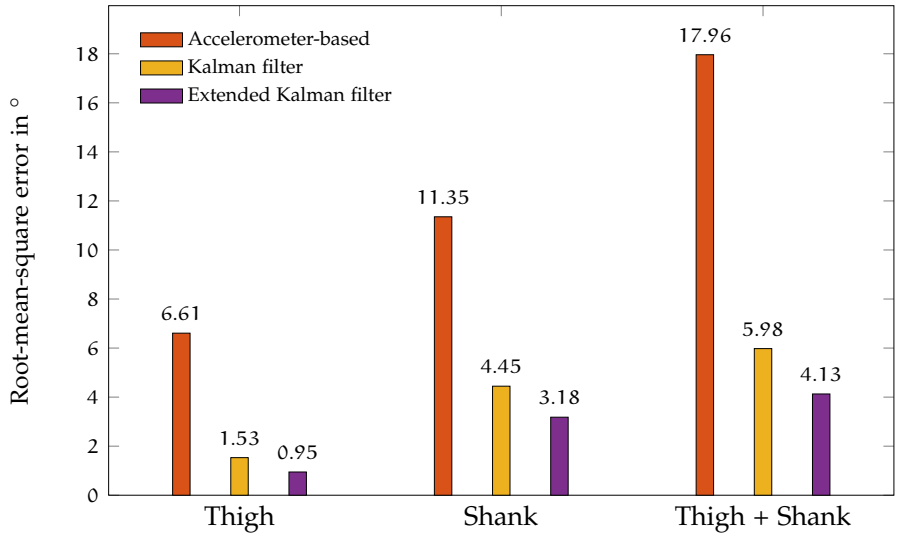

Figure 23: Root-mean-square error comparison of angle estimation, obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering. Walking speed: 2 km/h.

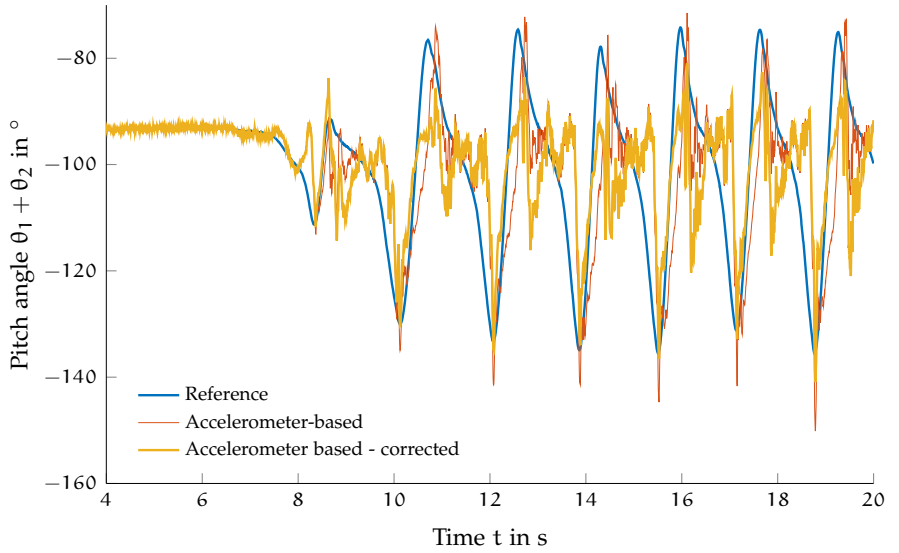

Figure 24: Accelerometer-based shank angle with respect to the x-axis with and without correction of acceleration signal. Walking speed: 2 km/h.

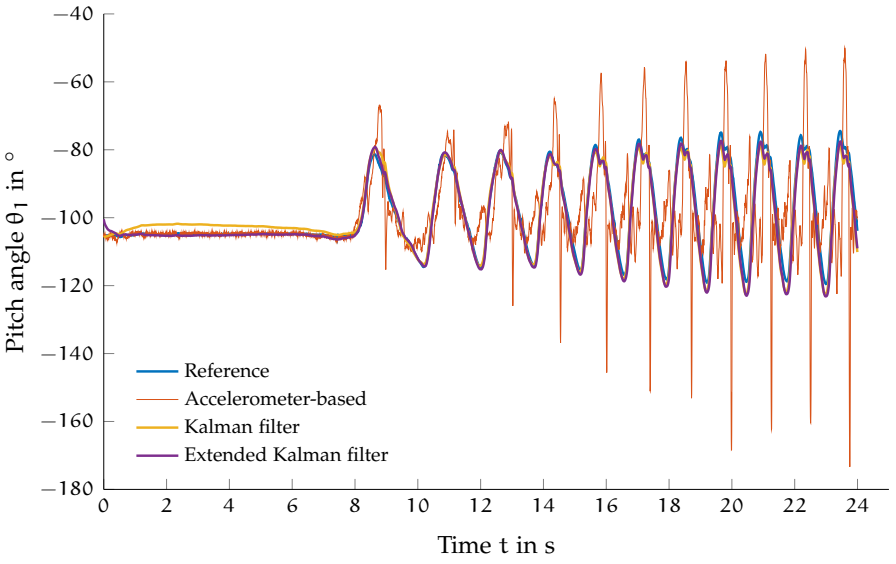

Figure 25: Pitch angle of the right thigh with respect to the x-axis, obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering, in comparison to the reference. Walking speed: 4 km/h.

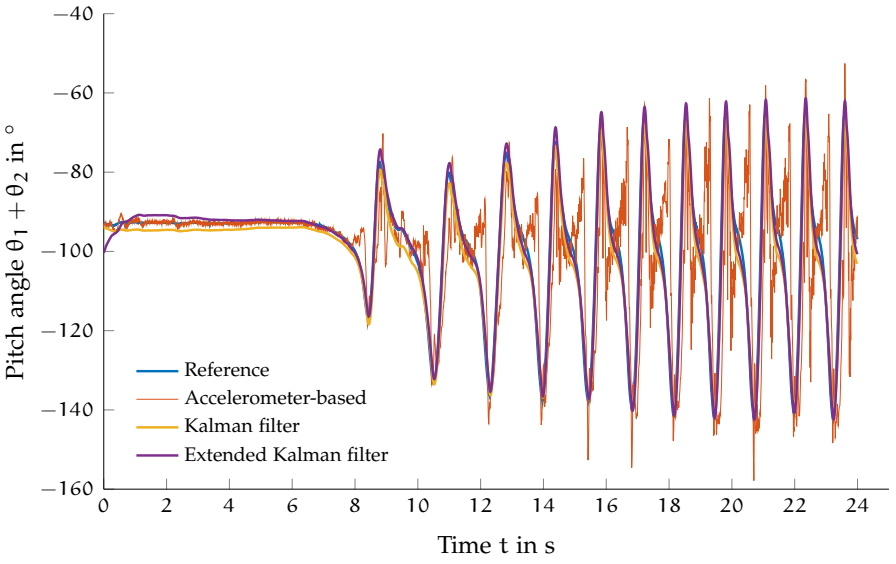

Figure 26: Pitch angle of the right shank with respect to the x-axis, obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering, in comparison to the reference. Walking speed: 4 km/h.

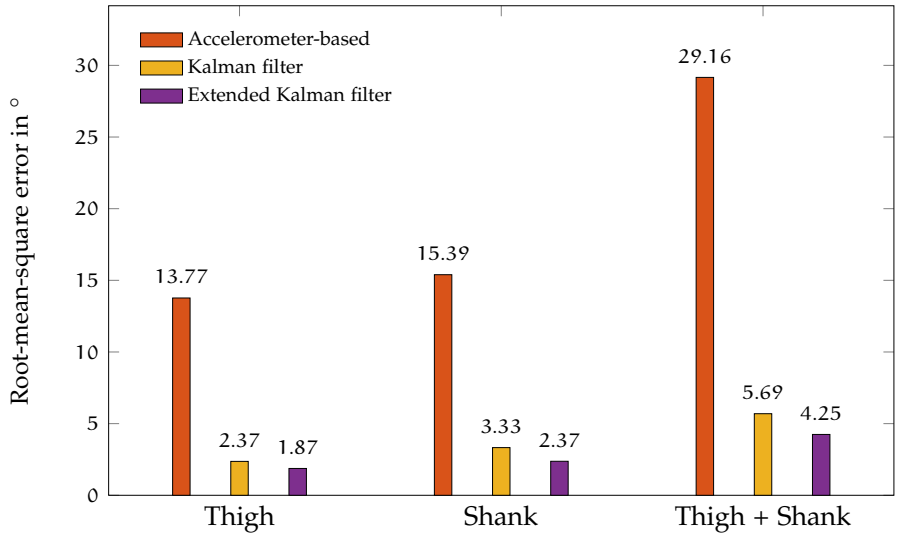

Figure 27: Root-mean-square error comparison of angle estimation, obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering. Walking speed: 4 km/h.

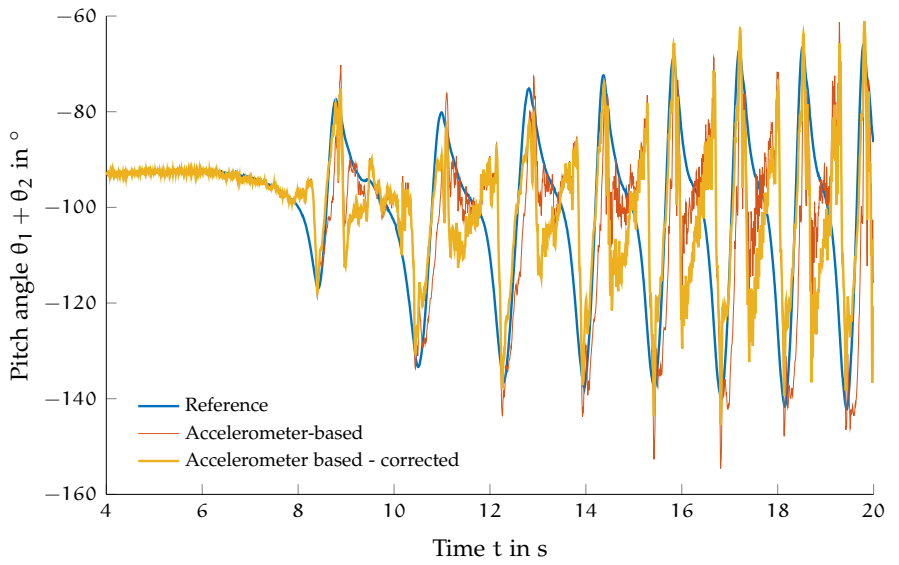

Figure 28: Accelerometer-based shank angle with respect to the x-axis with and without correction of acceleration signal. Walking speed: 4 km/h.

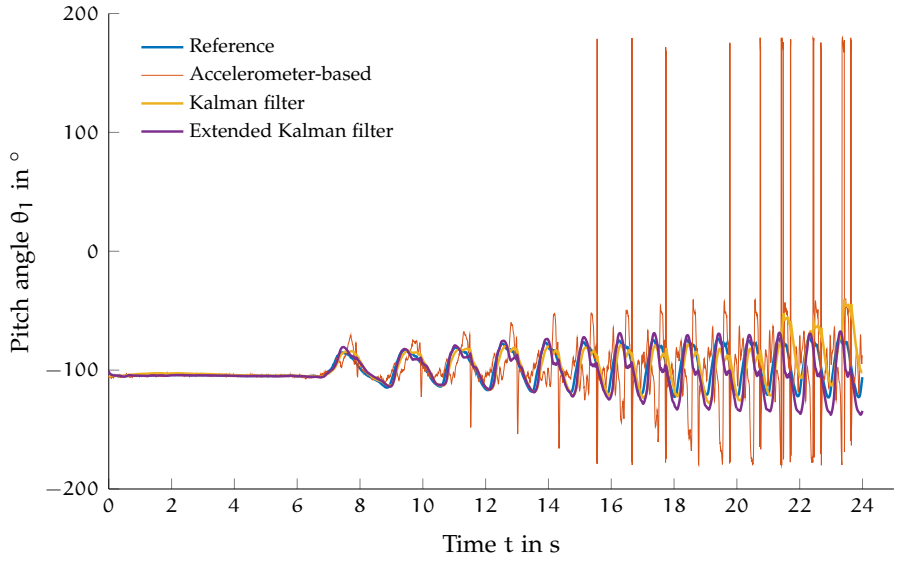

Figure 29: Pitch angle of the right thigh with respect to the x-axis, obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering, in comparison to the reference. Walking speed: 6 km/h.

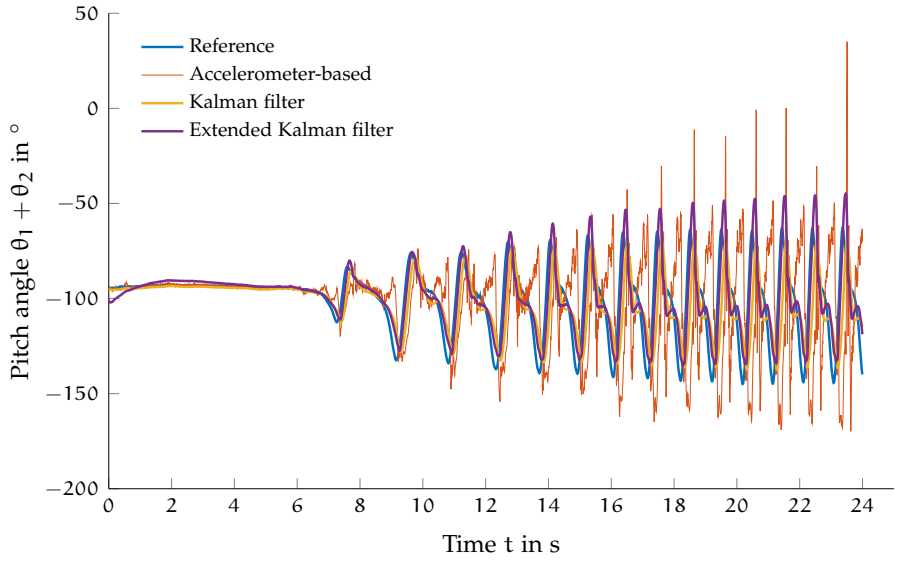

Figure 30: Pitch angle of the right shank with respect to the x-axis, obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering, in comparison to the reference. Walking speed: 6 km/h.

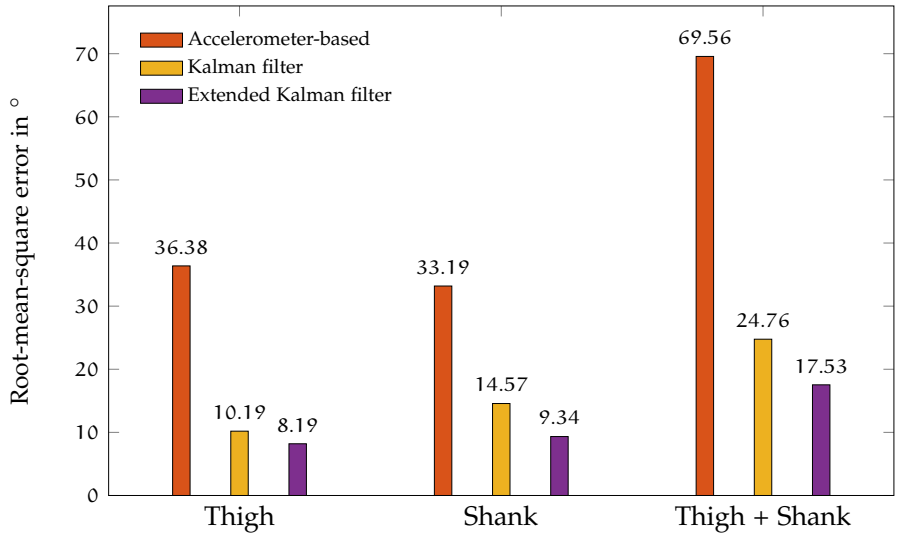

Figure 31: Root-mean-square error comparison of angle estimation, obtained by projection of the gravity vector, classical Kalman filtering, and extended Kalman filtering. Walking speed: 6 km/h.

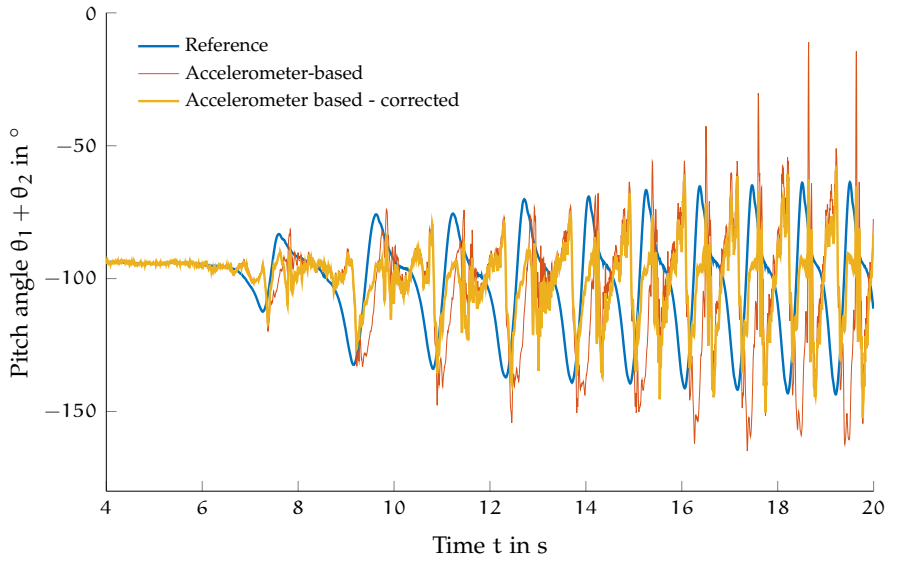

Figure 32: Accelerometer-based shank angle with respect to the x-axis with and without correction of acceleration signal. Walking speed: 6 km/h.

4.5 DISCUSSION

After the presentation of the results, we now proceed to analysing them in detail. According to Table 2 and Figures 24 and 28, the RMSEs of the motion-corrected accelerometer-based angle estimates while walking at $2\,\mathrm{km/h}$ and $4\,\mathrm{km/h}$ were only slightly different from the raw accelerometer-based angle estimates. This could be explained by the simple kinematic model of the leg. Even though it considers the motion of the leg in x and z-direction with respect to the origin of the world coordinate frame, that is the hip joint, it does not consider motion of the entire system itself. However, while walking, the human body rotates about the hip joint, which causes motion of the respective opposite hip joint. Additionally, the tension of the calf muscles and the resulting stretching of the foot moves the entire body upwards, including the hip joint. That means that the origin of the world coordinate frame of the kinematic model moves, contrary to the assumption made in the model. This movement causes not only an additional acceleration component in the sensor signal while moving the leg in the air, but also a quite severe impact on the acceleration signal when touching the ground with the foot. Another fact that is not considered in the simple model is motion beyond the xz-plane. Moreover, the lengths of the links was unknown and thus only estimated. Finally, the sensor position on the thigh and shank was not exactly known because the sensors were attached to the leg with elastic straps while the subject wore cloths, but it was assumed that they are perfectly aligned with the thighs and shanks. As the trend of the relative RMSEs of the accelerometer-based angle estimates in the fourth row of Table 2 shows, the motion correction works better for faster motions, which indicates the importance of the correction at higher walking speeds.

Even though the acceleration correction does not benefit the RMSE of the accelerometer-based angle estimate for walking speeds of $2\,\mathrm{km/h}$ and $4\,\mathrm{km/h}$, the overall improvement of the filter output is significant for all three walking speeds, as stated in the fifth row of Table 2. This improvement could be caused by a better filter tuning of the extended Kalman filter, and suboptimal tuning of the classical Kalman filter. The parameters found in an optimisation process, are not necessarily optimal. Due to the fact that the employed optimiser does not consider all possible combinations of the parameters, but instead finds local minima of the error function, the parameters may still be somewhat less than optimal and are strongly dependent on the initial guess. As one can see in Figures 24, 28, and 32, the motion-based acceleration correction improves the angle estimate in some intervals, but even worsens it in others. The Kalman filter compensates those deviations trusting on the state-space model. The newly implemented filter algorithm is based on a more complete state-space model.

For instance, it combines information about both thigh and shank in the same state-space model, whereas the classical Kalman filter estimates the thigh and shank angle independently. This could explain a more accurate estimate, compared to the classical Kalman filter, for intervals in which the acceleration correction does not decrease the RMSE of the accelerometer-based angle estimate.

5 CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

Gait analysis is a useful tool both in clinical practice and biomechanical research. In this work an extended Kalman filter based on a kinematic model of the leg was implemented in MATLAB®, along with a parameter optimiser that was necessary for testing. Then, the filter algorithm was applied to movement data of a human subject walking on a treadmill at different speeds. The experiments have shown that the filter algorithm can improve the angle estimates by an average RMSE that was 28.52% smaller, compared to the classical Kalman filter. However, the motion-based acceleration correction works better for faster motion. In order to replace a camera-based motion capture system with low cost wearable MARG sensors, some technical details still need to be improved. Numerous possible reasons for the behaviour seen in the results of the experiments were proposed. These include a lack of completeness of the kinematic model of the leg, unknown lengths of the thighs and shanks, and a non-optimal filter tuning of the classical Kalman filter, which served as a reference.

Personally, I have learned a lot in the eight months that I spent in Granada for writing this thesis and completing the internship in advance. Among others, I have come to know many new work methods, due to being exposed to people from different cultures. I gained experience in scientific research and a deeper understanding of human body motion analysis, in particular gait analysis, as well as how Kalman filtering benefits the accuracy of the orientation estimation. I improved my MATLAB® skills and I am now familiar with tools such as GitHub and Pivotal Tracker, which make working in a team much easier and significantly more efficient. While working at the research centre, I could improve my oral and written English skills.

All in all it was a great experience, professionally as well as personally. I truly and unreservedly recommend such a stay to *every single* university student.

5.2 FUTURE WORK

As discussed above, there are a variety of possible improvements regarding the kinematic model and the filter tuning, which lead to the following possible future work. One could extend the kinematic model of the leg, in order to take more complex movements of the body and especially the movement of the hip into account. Further-

more, the implemented orientation algorithm needs to be tested with a larger set of data from real subjects, in order to proof its validity and robustness. Especially, whether it delivers accurate orientation estimations without fine-tuning the parameters by means of an optimiser and a known reference for every patient and trial, needs to be tested. Finally, the acceleration correction based on a kinematic model could be applied to the accelerometer-based thigh angle estimation as well.

Regarding the GaitWatch system itself, the acceleration correction could be extended to the orientation estimation of the arms and the trunk. Then, the GaitWatch system could replace the camera-based motion capture systems in the future and would thus add a significant value to its medical applications in gait analysis.

APPENDIX

A.1 MATLAB® CODE

Listing 1: MATLAB® code file 'fusion_EKF.m'

```
function [theta1, theta2, theta12_c, a_m, X, par] = ...
            fusion_EKF(gyro_thigh_y, gyro_shank_y, ...
                       acc_thigh_x, acc_thigh_z, ...
                       acc_shank_x, acc_shank_z, ...
                       fs, l1, l2, p)

% FUNCTION fusion_EKF applies an extended Kalman filter in order
% to fuse the accelerometer and gyroscope dataand thus obtain an
% accurate orientation estimate of the thighs and shanks.
%
% Input arguments:
% |_ 'gyro_thigh_y':  Row vector containing the angular rate of
%                     the thigh about the y-axis in degrees per
%                     second.
% |_ 'gyro_shank_y':  Row vector containing the angular rate of
%                     the shank about the y-axis in degrees per
%                     second.
% |_ 'acc_thigh_x':   Row vector containing the linear
%                     acceleration of the thigh along the x-axis
%                     in g.
% |_ 'acc_thigh_z':   Row vector containing the linear
%                     acceleration of the thigh along the z-axis
%                     in g.
% |_ 'acc_shank_x':   Row vector containing the linear
%                     acceleration of the shank along the x-axis
%                      in g.
% |_ 'acc_shank_z':   Row vector containing the linear
%                     acceleration of the shank along the z-axis
%                     in g.
% |_ 'fs':            Sampling frecuency in Hertz. Must be real
%                     positive.
% |_ 'l1':            Length of the thigh in m. Must be real
%                     positive.
% |_ 'l2':            Length of the shank in m. Must be real
%                     positive.
% |_ 'p':             Row vector consisting of the filter
%                     parameters:
%                     'sigma_s_3', 'sigma_s_4',
%                     'sigma_f_3', 'sigma_f_4',
%                     'sigma_b', 'sigma_t1'.
%                     These parameters can be found by means of
```

```
42 %                    the parameter optimiser 'optimize_EFK'
43 %
44 % Output:
45 % |_ 'theta1':       Row vector containing the thigh angle with
46 %                    respect to the x-axis of the world frame
47 %                    in radians.
48 % |_ 'theta2':       Row vector containing the shank angle with
49 %                    respect to the thigh in degrees.
50 % |_ 'theta12_c':    Row vector containing the motion- corrected
51 %                    angle theta_1 + theta_2 in degrees.
52 % |_ 'a_m':          Row vector containing the estimate of the
53 %                    acceleration that sensor 2 will see due to
54 %                    motion.
55 % |_ 'X':            10 x length(gyro_thigh_y) matrix
56 %                    containing the internal state vector at
57 %                    each instant as columns.
58 %
59 % IMPORTANT NOTE:    'gyro_thigh_y', 'gyro_shank_y',
60 %                    'acc_thigh_x', 'acc_thigh_z',
61 %                    'acc_shank_x', and 'acc_shank_z'
62 %                    must have the same length. Otherwise, an
63 %                    error will be returned.
64 % ------------------------------------------------------
65 % Authors:           Robin Weiss
66 % Entity:            University of Applied Sciences
67 %                    Munster, Munster, Germany
68 % Last modification: 13/05/2015
69 % ------------------------------------------------------
70
71 % 1) Check input arguments.
72 if ~isequal(length(gyro_thigh_y), ...
73             length(gyro_shank_y), ...
74             length(acc_thigh_x), ...
75             length(acc_thigh_z), ...
76             length(acc_shank_x), ...
77             length(acc_shank_z))
78     error(['Input arguments ''gyro_thigh_y'', ', ...
79            '''acc_thigh_x'', ''acc_thigh_z'', ', ...
80            '''acc_shank_x'', ''acc_shank_z'', ', ...
81            'must have the same length.']);
82 end
83
84 if (fs <= 0 || ~isreal(fs))
85     error(['Input argument ''fs'' must be real', ...
86            'positive.']);
87 end
88
89 if (l1 <= 0 || ~isreal(l1))
90     error(['Input argument ''a1'' must be real', ...
91           ' positive.']);
92 end
93
```

```
94  if (l2 <= 0 || ~isreal(l2))
95      error(['Input parameter ''a2'' must be real', ...
96             ' positive.']);
97  end
98
99  % 2) Import GaitWatch and WaGyroMag functions library.
100 %    All existing functions have to be called using either
101 %    'gw.functionName' or 'wag.functionName'.
102 gw = gwLibrary;
103 wag = wagLibrary;
104
105 % 3) Compute the sampling period and the length of the signal
106 %    vectors.
107 Ts = 1 / fs;
108 len = length(gyro_thigh_y);
109
110 % 4) Set the magnitude of gravity.
111 gravity = 9.81;
112
113 % 5) Compute correction factor. This factor will be used
114 %    throughout the entire code in order to convert degrees into
115 %    radians.
116 c_w = pi / 180;
117
118 % 6) Initialise output vectors.
119 theta1 = zeros(1, len);
120 theta2 = zeros(1, len);
121 theta12_c = zeros(1, len);
122 a_m = zeros(3, len);
123 X = zeros(10, len);
124
125 % 7) Compute intensity level.
126 lwin_ltsd = 20;
127 threshold_ltsd = 4;
128 shift_ltsd = 19;
129 input_signal = sqrt(acc_shank_x.^2+acc_shank_z.^2);
130 [V_fsd, T_fsd] = wag.ltsd(input_signal', lwin_ltsd, ...
131                       shift_ltsd, 512, threshold_ltsd);
132
133 % 8) Determine marker signal.
134 [marker, ~] = gw.compEstMark(V_fsd, T_fsd, ...
135                              input_signal, lwin_ltsd, ...
136                              shift_ltsd);
137
138 % INITIALISATION OF PARAMETERS %
139
140 % 9) Compute mean of the first two seconds of the gyroscope
        signals.
141 mu1 = mean(gyro_thigh_y(1:2*fs));
142 mu2 = mean(gyro_shank_y(1:2*fs));
143
144 % 10) Initialise the state vector.
```

```
145 x = [0, -(l1+l2), -100, 0, 0, 0, 0, 0, mu1, mu2]';
146
147 % 11) Initialise the error covariance matrix.
148 P = diag(ones(1, 10) * 1);
149
150 % 12) Define the measurement matrix.
151 H = [0 0 0 1 0 0 0 0 1 0; ...
152      0 0 0 1 0 0 1 0 1 1; ...
153      0 0 1 0 0 0 0 0 0 0; ...
154      0 0 1 0 0 1 0 0 0 0];
155
156 % 13) Define process noise covariance matrix.
157 sigma_d_sq = 10;
158 sigma_t1_sq = p(6);
159 sigma_t2_sq = p(6);
160 sigma_b_sq = p(5);
161 Q = [...
162 sigma_d_sq 0 0         0          0      0 0 0 0 0; ...
163 0 sigma_d_sq 0         0          0      0 0 0 0 0; ...
164 0 0 sigma_t1_sq^9/9 sigma_t1_sq^4/4 sigma_t1_sq^5/5 0 0 0 0 0;
        ...
165 0 0 sigma_t1_sq^4/4 sigma_t1_sq^3/3 sigma_t1_sq^2/2 0 0 0 0 0;
        ...
166 0 0 sigma_t1_sq^5/5 sigma_t1_sq^2/2 sigma_t1_sq   0 0 0 0 0; ...
167 0 0 0 0 0 sigma_t2_sq^9/9 sigma_t2_sq^4/4 sigma_t2_sq^5/5 0 0;
        ...
168 0 0 0 0 0 sigma_t2_sq^4/4 sigma_t2_sq^3/3 sigma_t2_sq^2/2 0 0;
        ...
169 0 0 0 0 0 sigma_t2_sq^5/5 sigma_t2_sq^2/2 sigma_t2_sq   0 0; ...
170 0 0 0 0 0       0             0          0 sigma_b_sq 0; ...
171 0 0 0 0 0       0             0          0 0 sigma_b_sq];
172
173 % 14) Compute sample variance of the first two
174 %     seconds of the gyroscope signals.
175 sigma_1_sq = var(gyro_thigh_y(1:2*fs));
176 sigma_2_sq = var(gyro_shank_y(1:2*fs));
177
178 % 15) Define measurement noise covariance matrix.
179 sigma_s3_sq = p(1);
180 sigma_s4_sq = p(2);
181 sigma_f3_sq = p(3);
182 sigma_f4_sq = p(4);
183 R = [sigma_1_sq    0       0       0; ...
184         0    sigma_2_sq    0       0; ...
185         0       0    sigma_s3_sq   0; ...
186         0       0       0   sigma_s4_sq];
187
188 % Map parameter vector to output
189 par = [sigma_d_sq, sigma_t1_sq, sigma_t2_sq, ...
190        sigma_b_sq, sigma_1_sq, sigma_2_sq, ...
191        sigma_s3_sq, sigma_f3_sq, sigma_s4_sq, ...
192        sigma_f4_sq];
```

```
193
194 % 16) Define matrix function f.
195 function f_k = f
196
197         f_k = [- l1 * c_w * x(4) * sind(x(3)) ...
198            - l2 * c_w * (x(4) + x(7)) * sind(x(3) + x(6)); ...
199            - l1 * c_w * x(4) * cosd(x(3)) ...
200            - l2 * c_w * (x(4) + x(7)) * cosd(x(3) + x(6)); ...
201            x(4);
202            x(5);
203            0;
204            x(7);
205            x(8);
206            0;
207            0;
208            0];
209
210 end
211
212 % 17) Define Jacobian of F.
213 function F_k = F
214
215     A = - l1 * c_w * x(4) * cosd(x(3)) ...
216         - l2 * c_w * (x(4) + x(7)) * cosd(x(3) + x(6));
217     B = + l1 * c_w * x(4) * sind(x(3)) ...
218         + l2 * c_w * (x(4) + x(7)) * sind(x(3) + x(6));
219     C = - l1 * sind(x(3)) - l2 * sind(x(3) + x(6));
220     D = - l1 * cosd(x(3)) - l2 * cosd(x(3) + x(6));
221     E = - l2 * c_w * (x(4) + x(7)) * cosd(x(3) + x(6));
222     F = + l2 * c_w * (x(4) + x(7)) * sind(x(3) + x(6));
223     G = - l2 * sind(x(3) + x(6));
224     h = - l2 * cosd(x(3) + x(6));
225
226     F_k = [0, 0, A, C, 0, E, G, 0, 0, 0; ...
227            0, 0, B, D, 0, F, h, 0, 0, 0; ...
228            0, 0, 0, 1, 0, 0, 0, 0, 0, 0; ...
229            0, 0, 0, 0, 1, 0, 0, 0, 0, 0; ...
230            0, 0, 0, 0, 0, 0, 0, 0, 0, 0; ...
231            0, 0, 0, 0, 0, 0, 1, 0, 0, 0; ...
232            0, 0, 0, 0, 0, 0, 0, 1, 0, 0; ...
233            0, 0, 0, 0, 0, 0, 0, 0, 0, 0; ...
234            0, 0, 0, 0, 0, 0, 0, 0, 0, 0; ...
235            0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
236
237 end
238
239 % 18) Filter loop.
240 for i=1:1:len
241
242     % TIME UPDATE %
243
244     % Compute fundamental matrix.
```

```
245     Phi = eye(10) + F * Ts;
246
247     % Compute a priori state estimate.
248     x = x + f * Ts;
249
250     % Compute a priori error covariance matrix.
251     P = Phi * P * Phi' + Q;
252
253     % CORRECT SENSOR READINGS %
254
255     % Compute acceleration in the world frame that occurs due to
256     % motion, based on a priori state estimate.
257     ax = - l1 * ((c_w * x(4))^2 * cosd(x(3)) ...
258          + c_w * x(5) * sind(x(3))) ...
259          - l2 * ((c_w * x(4) + c_w * x(7))^2 ...
260          * cosd(x(3) + x(6)) ...
261          + c_w * (x(5) + x(8)) * sind(x(3) + x(6)));
262     az = - l1 * (c_w * x(5) * cosd(x(3)) ...
263          - (c_w * x(4))^2 * sind(x(3))) ...
264          - l2 * (c_w * (x(5) + x(8)) ...
265          * cosd(x(3) + x(6)) ...
266          - (c_w * x(4) + c_w * x(7))^2 ...
267          * sind(x(3) + x(6)));
268
269     % Normalise acceleration to gravity.
270     ax_n = ax / gravity;
271     az_n = az / gravity;
272
273     % Compute transformation matrix
274     Tz = [cosd(x(3) + x(6) + 90), 0, ...
275          -sind(x(3) + x(6) + 90); 0, 1, 0; ...
276           sind(x(3) + x(6) + 90), 0, ...
277           cosd(x(3) + x(6) + 90)];
278
279     % Rotate acceleration to body frame.
280     a_mb = Tz * [ax_n; 0; az_n];
281
282     % Compute gravity estimate by subtracting motion-based
283     % acceleration from sensor readings.
284     g_c = [acc_shank_x(i); 0; acc_shank_z(i)] - a_mb;
285
286     % Constitute the measurement vector from the gyroscope
287     % signals, theta_1, and the corrected angle estimate
288     % theta_1 + theta_2.
289     z = [gyro_thigh_y(i); gyro_shank_y(i); 0; 0];
290     z(3) = atan2d(acc_thigh_z(i), acc_thigh_x(i)) - 180;
291     z(4) = atan2d(g_c(3), g_c(1)) - 180;
292
293     % Output map.
294     theta12_c(i) = z(4);
295     a_m(:, i) = a_mb;
296
```

```
297     % Set sigma_3 in measurement noise covariance matrix
298     % according to motion intensity.
299     if marker(i) == 1
300        R(3, 3) = sigma_f3_sq;
301        R(4, 4) = sigma_f4_sq;
302     end
303
304     if marker(i) == 0
305        R(3, 3) = sigma_s3_sq;
306        R(4, 4) = sigma_s4_sq;
307     end
308
309     % MEASUREMENT UPDATE %
310
311     % Compute Kalman gain.
312     K = P * H' / (H * P * H' + R);
313
314     % Compute a posteriori estimate.
315     x = x + K * (z - H * x);
316
317     % Update error covariance matrix.
318     P = (eye(10) - K * H) * P;
319
320     % Map internal states to output vector.
321     theta1(i) = x(3);
322     theta2(i) = x(6);
323
324     % Map the entire state vector to the output.
325     X(:, i) = x;
326
327 end
328
329 end
```

Listing 2: MATLAB® code file 'optimise_EKF.m'

```
1  function [x_min, f_min, ct] = optimize_EKF( ...
2                          gyro_thigh_y, gyro_shank_y, ...
3                          acc_thigh_x, acc_thigh_z, ...
4                          acc_shank_x, acc_shank_z, ...
5                          fs, l1, l2, ref_angles, p0, ...
6                          rmse_off)
7
8  % FUNCTION OPTIMIZE_EKF uses an adaptive Nelder-Mead
9  % simplex ANMS algorithm to find the optimal parameters
10 % of the Extended Kalman filter.
11 %
12 % Input arguments:
13 % |_ 'gyro_thigh_y':  Row vector containing the angular
14 %                     rate of the thigh about the
15 %                     y-axis in radians per second.
16 % |_ 'gyro_shank_y':  Row vector containing the angular
```

```
17 %                    rate of the shank about the
18 %                    y-axis in radians per second.
19 % |_ 'acc_thigh_x':  Row vector containing the linear
20 %                    acceleration of the thigh along
21 %                    the x-axis in g.
22 % |_ 'acc_thigh_z':  Row vector containing the linear
23 %                    acceleration of the thigh along
24 %                    the z-axis in g.
25 % |_ 'acc_shank_x':  Row vector containing the linear
26 %                    acceleration of the shank along
27 %                    the x-axis in g.
28 % |_ 'acc_shank_z':  Row vector containing the linear
29 %                    acceleration of the shank along
30 %                    the z-axis in g.
31 % |_ 'fs':           Sampling frecuency in Hertz. Must
32 %                    be real positive.
33 % |_ 'l1':           Length of the thigh in m. Must be
34 %                    real positive.
35 % |_ 'l2':           Length of the shank in m. Must be
36 %                    real positive.
37 % |_ 'p':            Row vector consisting of the
38 %                    filter parameters sigma_t1,
39 %                    sigma_t2, sigma_b, sigma_f_1,
40 %                    sigma_f_2, sigma_s_1, sigma_s_2.
41 % |_ 'ref_angle':    Orientation angle reference.
42 % |_ 'p0':           Initial value of the parameters
43 %                    to be optimized.
44 % |_ 'rmse_off':     Offset in the RMSE computation.
45 %
46 % % Output:
47 % |_ 'xmin':         Value of the parameters that
48 %                    minimize the error function.
49 % |_ 'fmin':         Minimum value of the error
50 %                    function.
51 % |_ 'ct':           Number of algorithm iterations
52 %                    to find the minimum.
53
54 % 1) Set variables.
55 % ------------------------------------------------------
56 global gyro_thigh_y_g; gyro_thigh_y_g = gyro_thigh_y;
57 global gyro_shank_y_g; gyro_shank_y_g = gyro_shank_y;
58 global acc_thigh_x_g; acc_thigh_x_g = acc_thigh_x;
59 global acc_thigh_z_g; acc_thigh_z_g = acc_thigh_z;
60 global acc_shank_x_g; acc_shank_x_g = acc_shank_x;
61 global acc_shank_z_g; acc_shank_z_g = acc_shank_z;
62 global fs_g; fs_g = fs;
63 global l1_g; l1_g = l1;
64 global l2_g; l2_g = l2;
65 global true_angles;    true_angles = ref_angles;
66 global rmse_offset;   rmse_offset = rmse_off;
67
68 % 2) Call the minimisation routine.
```

```
% ------------------------------------------------------
disp('Optimising parameters of extended Kalman Filter...');

% Set tolerance limit between the minimum values found
% in subsequent iterations of the algorithm.
tol = 10^-6;

% Set maximum number of evaluations of the error
% function.
max_feval = 5000;

% Call ANMS algorithm which minimises the error function.
[x_min, f_min, ct] = ANMS(@eofEKF, p0, tol, max_feval);

end
```

Listing 3: MATLAB® code file 'eofEKF.m'

```
function F = eofEKF(p)

% FUNCTION EOFKALMAN is the error function to be
% minimised: That is, the sum of the RMSE between the
% actual angle and the estiamted orientation angle
% computed with the extended Kalman filter.
%
% Input arguments:
% |_ 'p':   Vector of initial value of the parameters
%           to be optimized.
%
% Output:
% |_ 'F':   Value of the error function.

% 1) Set variables.
% ------------------------------------------------------
global gyro_thigh_y_g;
global gyro_shank_y_g;
global acc_thigh_x_g;
global acc_thigh_z_g;
global acc_shank_x_g;
global acc_shank_z_g;
global fs_g;
global l1_g;
global l2_g;
global true_angles;
global rmse_offset;

% 3) Estimate the orientation angle using the extended
%    Kalman Filter.
[theta1, theta2, ~, ~, ~] = fusion_EKF(...
                  gyro_thigh_y_g, gyro_shank_y_g, ...
                  acc_thigh_x_g, acc_thigh_z_g, ...
                  acc_shank_x_g, acc_shank_z_g, ...
```

```
                    fs_g, l1_g, l2_g, p);

thigh_angle_EKF = theta1;
shank_angle_EKF = theta1 + theta2;

% 4) Compute the error function.
F1 = sqrt(mean((true_angles(1, rmse_offset : end) -  ...
     thigh_angle_EKF(rmse_offset : end)) .^ 2));
F2 = sqrt(mean((true_angles(2, rmse_offset : end) -  ...
     shank_angle_EKF(rmse_offset : end)) .^ 2));

F = F1 + F2;
```

Listing 4: MATLAB® code file 'EKF_experiments_1.m'

```
%% 0) Initialisation \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
% ------------------------------------------------------

clear all; close all; clc;

% Generate Tikz-pictures: Yes (1), or no (0).
tikz = 1;

% Load existing signals of the GaitWatch and the
% Qualisys motion capture system.
load('GaitWatch_data_1.mat');
load('Qualisys_data_1.mat');

% Import GaitWatch and WaGyroMag functions library.
% All existing functions have to be called using
% either 'gw.functionName' or 'wag.functionName'.
gw = gwLibrary;
wag = wagLibrary;

% Initialise number of figure.
n = 4;

% Set value of the magnitude of the gravity vector.
g = 9.81;

% Set the first and the last sample, that is, the
% interval of the signal that is used for the following
% computations.
n1 = 1;
n2 = 24 * f;

%% 1) Optimise filter parameters \\\\\\\\\\\\\\\\\\\\\\
% ------------------------------------------------------

% Set RMSE offset.
rmse_offset = 1;

```

```
% ---KALMAN FILTER THIGH-----------------------------

% Set initial value of parameters;
p0_KF = [1000 0.001];

% Call the optimization routine.
[xmin, fmin, ct] = gw.optimize_KF( ...
            g_Y_right_thigh_1_C(n1:n2)', ...
            pitch_acc_right_thigh(n1:n2)', ...
            f, var(pitch_acc_right_thigh(n1:n2)), ...
            var(pitch_acc_right_thigh(n1:n2)), ...
            var(g_Y_right_thigh_1_C(n1:n2)), ...
            pitch_acc_right_thigh(1), ...
            pitch_QS_right_thigh(n1:n2), ...
            p0_KF, rmse_offset);

fprintf('------------KF OPTIMISATION-----------\n');
fprintf(['The optimisation process finished in %d ', ...
         'iterations.\n'], ct);
fprintf('The minimum RMSE found is: %0.4f\n', fmin);
fprintf(['Optimal parameters are: \n -Alpha: %0.4f', ...
        '\n -Beta: %0.4f\n'], xmin(1), xmin(2))
fprintf('-------------------------------------\n')

% Extract optimal parameters.
opt_alpha_KF = xmin(1);
opt_beta_KF = xmin(2);

% Compute thigh angle.
pitch_KF_right_thigh = gw.fusion_KF( ...
        g_Y_right_thigh_1_C, pitch_acc_right_thigh, ...
        f, var(pitch_acc_right_thigh), ...
        var(pitch_acc_right_thigh),...
        var(g_Y_right_thigh_1_C), opt_alpha_KF, ...
        opt_beta_KF, pitch_acc_right_thigh(1));

% ---KALMAN FILTER SHANK-----------------------------

% Set initial value of parameters;
p0_KF = [1000 0.001];

% Call the optimization routine.
[xmin, fmin, ct] = gw.optimize_KF( ...
                g_Y_right_shank_1_C(n1:n2)', ...
                pitch_acc_right_shank(n1:n2)', f, ...
                var(pitch_acc_right_shank(n1:n2)), ...
                var(pitch_acc_right_shank(n1:n2)),...
                var(g_Y_right_shank_1_C(n1:n2)), ...
                pitch_acc_right_shank(1), ...
                pitch_QS_right_shank(n1:n2), ...
                p0_KF, rmse_offset);

```

```
fprintf('----------KF OPTIMISATION----------\n');
fprintf(['The optimisation process finished in %d ', ...
         'iterations.\n'], ct);
fprintf('The minimum RMSE found is: %0.4f\n', fmin);
fprintf(['Optimal parameters are: \n -Alpha: %0.4f', ...
        '\n -Beta: %0.4f\n'], xmin(1), xmin(2))
fprintf('-----------------------------------------\n')

% Extract optimal parameters.
opt_alpha_KF = xmin(1);
opt_beta_KF = xmin(2);

% Compute shank angle.
pitch_KF_right_shank = gw.fusion_KF( ...
        g_Y_right_shank_1_C, pitch_acc_right_shank, ...
        f, var(pitch_acc_right_shank), ...
        var(pitch_acc_right_shank),...
        var(g_Y_right_shank_1_C), opt_alpha_KF, ...
        opt_beta_KF, pitch_acc_right_shank(1));

% ---EXTENDED KALMAN FILTER--------------------------

% Set initial value of parameters.
p0 = [3.5, 30, 30, 300, 0.001, 0.05];

 % Call the optimization routine.
[pmin, fmin, ct] = gw.optimize_EKF( ...
                    g_Y_right_thigh_1_C(n1:n2)', ...
                    g_Y_right_shank_1_C(n1:n2)', ...
                    a_X_right_thigh_1_C(n1:n2)', ...
                    a_Z_right_thigh_1_C(n1:n2)', ...
                    a_X_right_shank_1_C(n1:n2)', ...
                    a_Z_right_shank_1_C(n1:n2)', ...
                    f, 0.35, 0.25, ...
                    [pitch_QS_right_thigh(n1:n2); ...
                     pitch_QS_right_shank(n1:n2)] ...
                     - 90, p0, rmse_offset);

fprintf('----------EKF OPTIMISATION----------\n');
fprintf(['The optimisation process finished in %d ', ...
         'iterations.\n'], ct);
fprintf('The minimum RMSE found is: %0.4f\n', fmin);
fprintf(['Optimal parameters are: \n -sigma_t1: ', ...
         '%0.4f\n -sigma_t2: %0.4f\n -sigma_b: ', ...
         '%0.4f\n -sigma_s_1: %0.4f\n -sigma_s_2:', ...
         ' %0.4f'], pmin);
fprintf('-----------------------------------------\n')
%%
% Compute pitch angles with extended Kalman filter.
% Additionally, store the internal state vector at each
% time step in x, the motion based acceleration in a_m,
% and the angle estimate theta_1 + theta_2, based on
```

```
% the corrected acceleration signal in theta12_c.
[pitch_EKF_right_thigh, pitch_EKF_right_shank, ...
 theta12_c, a_m, x, par] = fusion_EKF( ...
                            g_Y_right_thigh_1_C', ...
                            g_Y_right_shank_1_C', ...
                            a_X_right_thigh_1_C', ...
                            a_Z_right_thigh_1_C', ...
                            a_X_right_shank_1_C', ...
                            a_Z_right_shank_1_C', ...
                            f, 0.35, 0.25, pmin);

save ('Data_1/pmin_1', 'pmin')
save ('Data_1/po_1', 'po')
save ('Data_1/parameters_1', 'par')

%% 3) Plot results \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
% ------------------------------------------------------

% Plot: Thigh angle estimate - acceleration-based,
%       KF, and EKF.
figure(n);
hold on;
plot(time(n1:n2), pitch_QS_right_thigh(n1:n2) - 90, ...
     'linewidth', 1);
plot(time(n1:n2), pitch_acc_right_thigh(n1:n2) - 90);
plot(time(n1:n2), pitch_KF_right_thigh(n1:n2) - 90, ...
     time(n1:n2), pitch_EKF_right_thigh(n1:n2), ...
     'linewidth', 1);

xlabel('Time $t$ in s', 'interpreter', 'latex');
ylabel(['Pitch angle $\theta_1$ in ', ...
        '$^{\circ}$'], 'interpreter','latex');
legend('Reference', 'Accelerometer–based', ...
       'Kalman filter', 'Extended Kalman filter', ...
       'Location','northwest');
legend('boxoff');

if tikz
matlab2tikz(['../tikz/experiment_', num2str(n), ...
             '.tikz'], 'height', '\figureheight', ...
             'width', '\figurewidth');
end

n = n + 1;

% Plot: Shank angle estimate - acceleration-based,
%       KF, and EKF.
figure(n);
hold on;
plot(time(n1:n2), pitch_QS_right_shank(n1:n2) - 90, ...
     'linewidth', 1);
plot(time(n1:n2), pitch_acc_right_shank(n1:n2)-90);
```

```
194 plot(time(n1:n2), pitch_KF_right_shank(n1:n2)-90, ...
195      time(n1:n2), pitch_EKF_right_thigh(n1:n2) ...
196      + pitch_EKF_right_shank(n1:n2), 'linewidth', 1);
197
198 xlabel('Time $t$ in s', 'interpreter', 'latex');
199 ylabel(['Pitch angle $\theta_1 + \theta_2$ in ', ...
200         '$^{\circ}$'], 'interpreter','latex');
201 legend('Reference', 'Accelerometer-based', ...
202        'Kalman filter', 'Extended Kalman filter', ...
203        'Location','southwest');
204 legend('boxoff');
205
206 if tikz
207 matlab2tikz(['../tikz/experiment_', num2str(n), ...
208              '.tikz'], 'height', '\figureheight', ...
209              'width', '\figurewidth');
210 end
211
212 n = n + 1;
213
214 % Compute root-mean-square error.
215 % -Thigh
216 RMSE_acc = sqrt(mean((pitch_QS_right_thigh(n1:n2) ...
217                 - pitch_acc_right_thigh(n1:n2)').^2));
218 RMSE_KF = sqrt(mean((pitch_QS_right_thigh(n1:n2) ...
219                - pitch_KF_right_thigh(n1:n2)').^2));
220 RMSE_EKF = sqrt(mean((pitch_QS_right_thigh(n1:n2) ...
221             - 90 - pitch_EKF_right_thigh(n1:n2)).^2));
222 RMSE = [RMSE_acc, RMSE_KF, RMSE_EKF];
223 % -Shank
224 RMSE_acc = sqrt(mean((pitch_QS_right_shank(n1:n2) ...
225                 - pitch_acc_right_shank(n1:n2)').^2));
226 RMSE_KF = sqrt(mean((pitch_QS_right_shank(n1:n2) ...
227                - pitch_KF_right_shank(n1:n2)').^2));
228 RMSE_EKF = sqrt(mean((pitch_QS_right_shank(n1:n2) ...
229             - 90 - pitch_EKF_right_thigh(n1:n2) ...
230             - pitch_EKF_right_shank(n1:n2)).^2));
231 RMSE = [RMSE; RMSE_acc, RMSE_KF, RMSE_EKF;];
232
233 % Compute sum of seperate RMSEs.
234 RMSE = [RMSE; RMSE(1, 1) + RMSE(2, 1), RMSE(1, 2) + ...
235         RMSE(2, 2), RMSE(1, 3) + RMSE(2, 3)];
236
237 % Plot bar graph and values on top of the bars.
238 figure(n);
239 b = bar(RMSE, 0.3);
240 offset = 0.8;
241 yb = cat(1, b.YData);
242 xb = bsxfun(@plus, b(1).XData, [b.XOffset]');
243 hold on;
244
245 for i = 1:3
```

```
246     for j = 1:3
247         text(xb(j, i), yb(j, i) + offset, ...
248             ['\scriptsize ', num2str(RMSE(i, j), ...
249             '$%0.2f$')], 'rotation', 0, ...
250             'interpreter', 'latex', ...
251             'HorizontalAlignment','center');
252     end
253 end
254
255 b(1).FaceColor = [0.8500    0.3250    0.0980];
256 b(2).FaceColor = [0.9290    0.6940    0.1250];
257 b(3).FaceColor = [0.4940    0.1840    0.5560];
258
259 ylim([0, max(max(RMSE)) + 2]);
260 ylabel('Root-mean-square error in $^{\circ}$', ...
261        'interpreter','latex');
262
263 labels = {'Thigh', 'Shank', 'Thigh + Shank'};
264 format_ticks(gca, labels, [], [], [], 0);
265
266 legend('Acceleration-based', 'Kalman filter', ...
267        'Extended Kalman filter', ...
268        'Location','northwest');
269 legend('boxoff');
270
271 if tikz
272 matlab2tikz(['../tikz/experiment_', num2str(n), ...
273              '.tikz'], 'height', '\figureheight', ...
274              'width', '\figurewidth');
275 end
276
277 n = n + 1;
278
279 % Plot: Acceleration-based pitch angle shank - corrected.
280 n1 = 4 * f + 1;
281 n2 = 20 * f;
282 figure(n);
283 hold on;
284 plot(time(n1:n2), pitch_QS_right_shank(n1:n2) - 90, ...
285      'linewidth', 1);
286 plot(time(n1:n2), pitch_acc_right_shank(n1:n2) - 90);
287 plot(time(n1:n2), theta12_c(n1:n2), 'linewidth', 1);
288
289 xlabel('Time $t$ in s', 'interpreter','latex');
290 ylabel(['Pitch angle $\theta_1 + \theta_2$ in ', ...
291         '$^{\circ}$'], 'interpreter', 'latex');
292 legend('Reference', 'Accelerometer-based', ...
293        'Accelerometer based - corrected', ...
294        'Location','southwest');
295 legend('boxoff');
296
297 % Compute root-mean-square error.
```

```
298 RMSE_acc = sqrt(mean((pitch_QS_right_shank(n1:n2) ...
299                 - pitch_acc_right_shank(n1:n2)').^2));
300 RMSE_acc_corr = sqrt(mean((pitch_QS_right_shank(n1:n2)...
301                 - 90 - theta12_c(n1:n2)).^2));
302 RMSE_acc = [RMSE_acc, RMSE_acc_corr];
303
304 if tikz
305 matlab2tikz(['../tikz/experiment_', num2str(n), ...
306              '.tikz'], 'height', '\figureheight', ...
307              'width', '\figurewidth');
308 end
309
310 %% 3) Display results \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
311 % ------------------------------------------------------
312
313 fprintf('------------------Results----------------\n');
314 fprintf(['AB:\n RMSE_thigh: %0.4f\n RMSE_shank: ', ...
315          '%0.4f\n RMSE_sum:   %0.4f\n\n'], RMSE(:, 1));
316 fprintf(['KF:\n RMSE_thigh: %0.4f\n RMSE_shank: ', ...
317          '%0.4f\n RMSE_sum:   %0.4f\n\n'], RMSE(:, 2));
318 fprintf(['EKF:\n RMSE_thigh: %0.4f\n RMSE_shank: ', ...
319          '%0.4f\n RMSE_sum:   %0.4f\n\n'], RMSE(:, 3));
320 fprintf(['Improvement:\n', ...
321          'RMSE_SUM_EKF / RMSE_SUM_KF:  %0.4f\n\n'], ...
322          RMSE(3, 3) / RMSE(3, 2));
323 fprintf(['Acceleration correction:\n', ...
324          'RMSE_Acc / RMSE_Acc_corr:  %0.4f\n\n'], ...
325          RMSE_acc(2) / RMSE_acc(1));
326 fprintf(['EKF parameters: \n -sigma_d_sq:  ', ...
327          '%0.4f\n -sigma_t1_sq: ', ...
328          '%0.4f\n -sigma_t2_sq: ', ...
329          '%0.4f\n -sigma_b_sq:  ', ...
330          '%0.4f\n -sigma_1_sq:  ', ...
331          '%0.4f\n -sigma_2_sq:  ', ...
332          '%0.4f\n -sigma_s3_sq: ', ...
333          '%0.4f\n -sigma_f3_sq: ', ...
334          '%0.4f\n -sigma_s4_sq: ', ...
335          '%0.4f\n -sigma_f4_sq: ', ...
336          '%0.4f\n'], par);
337 fprintf('---------------------------------------\n');
```

BIBLIOGRAPHY

[1] Weijun Tao, Tao Liu, Rencheng Zheng and Hutian Feng. „Gait Analysis Using Wearable Sensors“. en. In: *Sensors* 12.2 (Feb. 2012), pp. 2255–2283. DOI: 10.3390/s120202255. URL: http://www.mdpi.com/1424-8220/12/2/2255.

[2] Paolo Bonato. „Advances in wearable technology and applications in physical medicine and rehabilitation“. In: *Journal of NeuroEngineering and Rehabilitation* 2 (2005), pp. 2–2. ISSN: 1743-0003. DOI: 10.1186/1743-0003-2-2. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC552335/.

[3] Wee-Soon Yeoh, I. Pek, Y.-H. Yong, Xiang C. and A.B. Waluyo. „Ambulatory monitoring of human posture and walking speed using wearable accelerometer sensors“. In: *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. 2008, pp. 5184–5187. DOI: 10.1109/IEMBS.2008.4650382.

[4] A. Godfrey, R. Conway, D. Meagher and G. ÓLaighin. „Direct measurement of human movement by accelerometry“. In: *Medical Engineering & Physics*. Special issue to commemorate the 30th anniversary of Medical Engineering & Physics 30th Anniversary Issue 30.10 (Dec. 2008), pp. 1364–1386. ISSN: 1350-4533. DOI: 10.1016/j.medengphy.2008.09.005.

[5] Terrell Bennett, Roozbeh Jafari and Nicholas Raphael Gans. „An extended Kalman filter to estimate human gait parameters and walking distance“. English (US). In: Sept. 2013. URL: http://www.essp.utdallas.edu/uploads/Main/Publications/ACC13.pdf.

[6] Wai Yin Wong, Man Sang Wong and Kam Ho Lo. „Clinical Applications of Sensors for Human Posture and Movement Analysis: A Review“. en. In: *Prosthetics and Orthotics International* 31.1 (Mar. 2007), pp. 62–75. ISSN: 0309-3646, 1746-1553. DOI: 10.1080/03093640600983949. URL: http://poi.sagepub.com/content/31/1/62.

[7] T.R. Bennett, R. Jafari and N. Gans. „Motion Based Acceleration Correction for Improved Sensor Orientation Estimates“. In: *2014 11th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. June 2014, pp. 109–114. DOI: 10.1109/BSN.2014.17.

[8] H.J. Luinge, P.H. Veltink and C.T.M. Baten. „Estimation of orientation with gyroscopes and accelerometers“. In: *Engineering in Medicine and Biology, 1999. BMES/EMBS Conference, 1999. Proceedings of the First Joint*. Vol. 2. 1999, 844 vol.2–. DOI: 10.1109/IEMBS.1999.803999.

[9] Alberto Olivares Vicente. „Signal Processing of Magnetic and Inertial Sensor's Signals applied to Human Body Motion Monitoring“. English. PhD thesis. Granada: University of Granada, Jan. 2013. URL: http://hera.ugr.es/tesisugr/21910947.pdf.

[10] P.H. Veltink, H.B.J. Bussmann, W. de Vries, Wim L.J. Martens and R.C. van Lummel. „Detection of static and dynamic activities using uniaxial accelerometers“. In: *Rehabilitation Engineering, IEEE Transactions on* 4.4 (1996), pp. 375–385. ISSN: 1063-6528. DOI: 10.1109/86.547939.

[11] B. Najafi, K. Aminian, A. Paraschiv-Ionescu, F. Loew, C.J. Bula and P. Robert. „Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly“. In: *Biomedical Engineering, IEEE Transactions on* 50.6 (2003), pp. 711–723. ISSN: 0018-9294. DOI: 10.1109/TBME.2003.812189.

[12] M. Ermes, J. Parkka, J. Mantyjarvi and I. Korhonen. „Detection of Daily Activities and Sports With Wearable Sensors in Controlled and Uncontrolled Conditions“. In: *Information Technology in Biomedicine, IEEE Transactions on* 12.1 (2008), pp. 20–26.

[13] O. Giggins, D. Kelly and B. Caulfield. „Evaluating rehabilitation exercise performance using a single inertial measurement unit“. In: *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on*. 2013, pp. 49–56.

[14] J. Lupinski, V. Menon, P. Roh, S. Yuen, A. Valdevit and J. Andrish. „Measuring knee compliance to facilitate post-op ligament rehabilitation“. In: *Bioengineering Conference (NEBEC), 2011 IEEE 37th Annual Northeast*. 2011, pp. 1–2. DOI: 10.1109/NEBC.2011.5778529.

[15] V. Bonnet, C. Mazza, P. Fraisse and A. Cappozzo. „Real-time Estimate of Body Kinematics During a Planar Squat Task Using a Single Inertial Measurement Unit“. In: *Biomedical Engineering, IEEE Transactions on* 60.7 (2013), pp. 1920–1926. ISSN: 0018-9294. DOI: 10.1109/TBME.2013.2245131.

[16] Rui Zhang, F. Hoflinger and L. Reindl. „Inertial Sensor Based Indoor Localization and Monitoring System for Emergency Responders“. In: *Sensors Journal, IEEE* 13.2 (2013), pp. 838–848. ISSN: 1530-437X. DOI: 10.1109/JSEN.2012.2227593.

[17] T. Bennett, R. Jafari and N. Gans. „An extended Kalman filter to estimate human gait parameters and walking distance“. In: *American Control Conference (ACC), 2013*. 2013, pp. 752–757. DOI: 10.1109/ACC.2013.6579926.

[18] A. K. Bourke and G. M. Lyons. „A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor“. In: *Medical Engineering & Physics* 30.1 (Jan. 2008), pp. 84–90. ISSN: 1350-4533. DOI: 10.1016/j.medengphy.2006.12.001. URL: http://www.sciencedirect.com/science/article/pii/S1350453306002657.

[19] A.K. Bourke, P. van de Ven, M. Gamble, R. O'Connor, K. Murphy, E. Bogan, E. McQuade, P. Finucane, G. OLaighin and J. Nelson. „Assessment of waist-worn tri-axial accelerometer based fall-detection algorithms using continuous unsupervised activities“. In: *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. 2010, pp. 2782–2785. DOI: 10.1109/IEMBS.2010.5626364.

[20] A.K. Bourke, P.W.J. van de Ven, A. Chaya, G. Olaighin and J. Nelson. „Design and test of a long-term fall detection system incorporated into a custom vest for the elderly“. In: *Signals and Systems Conference, 208. (ISSC 2008). IET Irish*. 2008, pp. 307–312.

[21] M. Mancini, C. Zampieri, P. Carlson-Kuhta, L. Chiari and F. B. Horak. „Anticipatory postural adjustments prior to step initiation are hypometric in untreated Parkinson's disease“. eng. In: *European Journal of Neurology: The Official Journal of the European Federation of Neurological Societies* 16.9 (Sept. 2009), pp. 1028–1034. ISSN: 1468-1331. DOI: 10.1111/j.1468-1331.2009.02641.x.

[22] L. Palmerini, L. Rocchi, S. Mellone, F. Valzania and L. Chiari. „Feature Selection for Accelerometer-Based Posture Analysis in Parkinson's Disease“. In: *IEEE Transactions on Information Technology in Biomedicine* 15.3 (May 2011), pp. 481–490. ISSN: 1089-7771. DOI: 10.1109/TITB.2011.2107916.

[23] D. G. M. de Klerk, J. P. P. van Vugt, J. a. G. Geelen and T. Heida. „A Long-Term Monitor Including Activity Classification for Motor Assessment in Parkinson's Disease Patients“. en. In: *4th European Conference of the International Federation for Medical and Biological Engineering*. Ed. by Jos Vander Sloten, Pascal Verdonck, Marc Nyssen and Jens Haueisen. IFMBE Proceedings 22. Springer Berlin Heidelberg, Jan. 2009, pp. 1706–1709. ISBN: 978-3-540-89207-6, 978-3-540-89208-3. URL: http://link.springer.com/chapter/10.1007/978-3-540-89208-3_406.

[24] Kaveh Saremi, Jon Marehbian, Xiaohong Yan, Jean-Philippe Regnaux, Robert Elashoff, Bernard Bussel and Bruce H. Dobkin. „Reliability and Validity of Bilateral Thigh and Foot Accelerometry Measures of Walking in Healthy and Hemiparetic Sub-

jects“. en. In: *Neurorehabilitation and Neural Repair* 20.2 (June 2006), pp. 297–305. ISSN: 1545-9683, 1552-6844. DOI: 10.1177/1545968306287171. URL: http://nnr.sagepub.com/content/20/2/297.

[25] Joe Verghese, Richard B. Lipton, Charles B. Hall, Gail Kuslansky, Mindy J. Katz and Herman Buschke. „Abnormality of Gait as a Predictor of Non-Alzheimer's Dementia“. In: *New England Journal of Medicine* 347.22 (2002). PMID: 12456852, pp. 1761–1768. DOI: 10.1056/NEJMoa020441. eprint: http://dx.doi.org/10.1056/NEJMoa020441. URL: http://dx.doi.org/10.1056/NEJMoa020441.

[26] Jussi Collin, Pavel Davidson, Martti Kirkko-Jaakkola and Helena Leppäkoski. „Inertial Sensors and Their Applications“. English. In: *Handbook of Signal Processing Systems*. Ed. by Shuvra S. Bhattacharyya, Ed F. Deprettere, Rainer Leupers and Jarmo Takala. Springer New York, 2013, pp. 69–96. ISBN: 978-1-4614-6858-5. DOI: 10.1007/978-1-4614-6859-2_3. URL: http://dx.doi.org/10.1007/978-1-4614-6859-2_3.

[27] M.N. Armenise, C. Ciminelli, F. Dell'Olio and V.M.N. Passaro. *Advances in Gyroscope Technologies*. Springer, 2010. URL: https://books.google.de/books?id=lJUiyigJRBgC.

[28] J. Lenz and Alan S. Edelstein. „Magnetic sensors and their applications“. In: *Sensors Journal, IEEE* 6.3 (2006), pp. 631–649. ISSN: 1530-437X. DOI: 10.1109/JSEN.2006.874493.

[29] M.J. Thompson, M. Li and D.A. Horsley. „Low power 3-axis Lorentz force navigation magnetometer“. In: *Micro Electro Mechanical Systems (MEMS), 2011 IEEE 24th International Conference on*. 2011, pp. 593–596. DOI: 10.1109/MEMSYS.2011.5734494.

[30] Juansempere. *Wikimedia Commons File: Taitbrianzyx.svg*. [Accessed 31 March, 2015]. URL: http://commons.wikimedia.org/wiki/File:Taitbrianzyx.svg.

[31] James Diebel. *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors*. Tech. rep. Stanford University, 2006, pp. 5–6.

[32] J.R. Raol. *Multi-Sensor Data Fusion with MATLAB®*. CRC Press, 2009. ISBN: 9781439800058. URL: https://books.google.de/books?id=7s8xpR-5rOUC.

[33] Simon Haykin. *Adaptive filter theory*. Upper Saddle River, N.J: Prentice Hall, 2002. ISBN: 9780130484345.

[34] Peter S. Maybeck. „Chapter 1 Introduction“. In: *Stochastic Models, Estimation and Control: Volume 1*. Mathematics in Science and Engineering. Elsevier, 1979, pp. 1 –24. DOI: 10.1016/S0076-5392(08)62166-9. URL: http://www.sciencedirect.com/science/article/pii/S0076539208621669.

[35] Rudolph Emil Kalman. „A New Approach to Linear Filtering and Prediction Problems“. In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[36] Gregory F. Welch. „Kalman Filter“. English. In: *Computer Vision*. Ed. by Katsushi Ikeuchi. Springer US, 2014, pp. 435–437. ISBN: 978-0-387-30771-8. DOI: 10.1007/978-0-387-31439-6_716. URL: http://dx.doi.org/10.1007/978-0-387-31439-6_716.

[37] Paul Zarchan and Howard Musoff. *Fundamentals of Kalman Filtering: A Practical Approach*. Progress in astronautics and aeronautics. Academic Press, 2009, pp. 34–38. ISBN: 9781600867187. URL: https://books.google.es/books?id=LUDKrQEACAAJ.

[38] M.S. Grewal and A.P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB*. Wiley, 2008. ISBN: 9780470377802. URL: http://books.google.es/books?id=J_fqMHOCzB8C.

[39] Alberto Olivares Vicente and Kai Bötzel. *GaitWatch User Manual*. User Manual.

[40] SparkFun Electronics. *IMU Analog Combo Board 5DOF*. [Accessed 17 March 2015]. URL: https://www.sparkfun.com/products/retired/9268.

[41] SparkFun Electronics. *Gyro Breakout Board IDG500*. [Accessed 17 March 2015]. URL: https://www.sparkfun.com/products/retired/9094.

[42] Analog Devices. *ADXL335: Small, Low Power 3-Axis ±3G Accelerometer*. [Accessed 17 March 2015]. URL: http://www.analog.com/en/products/mems/accelerometers/adxl335.html.

[43] Analog Devices. *ADXL345: 3-Axis, ±2g/±4g/±8g/±16g Digital Accelerometer*. [Accessed 17 March 2015]. URL: http://www.analog.com/en/products/mems/accelerometers/adxl345.html.

[44] InvenSense. *IMU-3000 Triple Axis MotionProcessor™Gyroscope*. [Accessed 17 March 2015]. URL: http://www.invensense.com/mems/gyro/imu3000.html.

[45] SparkFun Electronics. *MicroMag 3-Axis Magnetometer*. [Accessed 17 March 2015]. URL: https://www.sparkfun.com/products/retired/244.

[46] ALVIDI. *AVR ATxmega – Development Module*. [Accessed 17 March 2015]. URL: http://www.alvidi.de/avr_xmodul_en.html.

[47] M.W. Spong and S. Hutchinson. *Robot Modeling and Control*. Wiley, 2005. ISBN: 9780471649908. URL: http://books.google.de/books?id=wGapQAAACAAJ.

[48] Derek Rowell. *State-Space Representation of LTI Systems*. Tech. rep. 2002. URL: http://web.mit.edu/2.14/www/Handouts/StateSpace.pdf.

[49] Alonzo Kelly. *A 3D State Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles*. Tech. rep. CMU-RI-TR-94-19. Pittsburgh, PA: Robotics Institute, 1994.