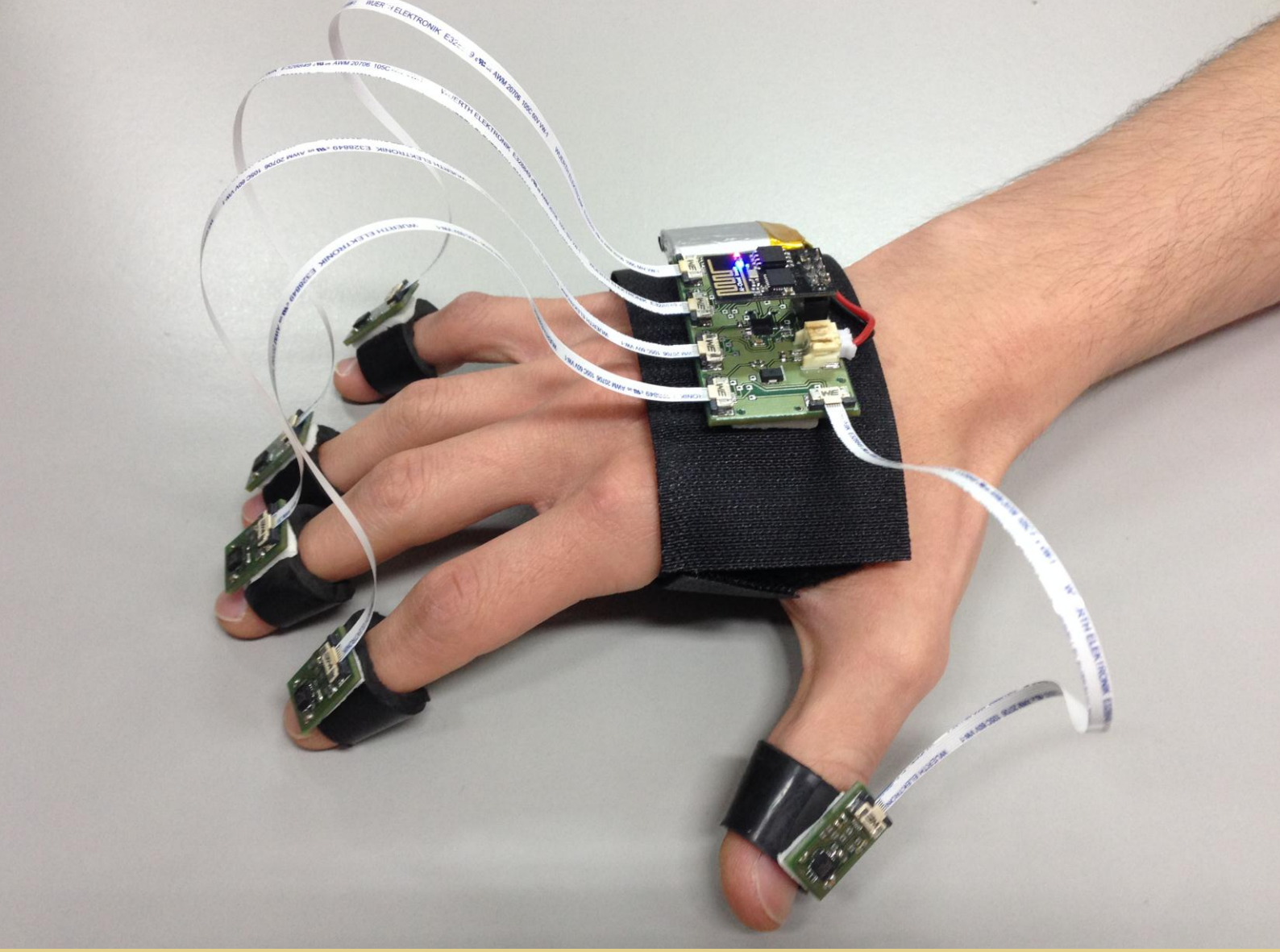

Tiago Filipe Rodrigues Catarino

Development of a Hand-Tracker:
Wireless Solution based on Inertial Sensors

September 2016

· U C ·

UNIVERSIDADE DE COIMBRA

UNIVERSITY OF COIMBRA
FACULTY OF SCIENCES AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Development of a Hand-Tracker

Wireless Solution based on Inertial Sensors

Tiago Filipe Rodrigues Catarino

Jury:
Prof. João Pedro de Almeida Barreto
Prof. João Filipe de Castro Cardoso Ferreira
Prof. Paulo Jorge Carvalho Menezes

Advisors:
Prof. Paulo Jorge Carvalho Menezes
Master Bruno André Santos Patrão

Dissertation submitted to the Electrical and Computer Engineering Department of the Faculty of Science and Technology of the University of Coimbra in partial fulfilment of the requirements for the Degree of Master of Science.

Coimbra, September of 2016

Acknowledgements

The accomplishment of this work would not have been possible without the contribution of a group of people who have been beside me along the way. First and foremost, I would like to express my gratitude to my advisers Professor Paulo Jorge Carvalho Menezes and Bruno André Santos Patrão for their permanent teaching and scientific rigor. I would also like to acknowledge my colleagues at the Mobile Robotics Laboratory in ISR, particularly Samuel Pedro and José Francisco, for their constant contributions and involvement in every step throughout the process.

Getting through my dissertation required not only academic support but also a great deal of friendship and companionship, in this way for many memorable evenings out and in, I must thank all my colleagues and friends at the Department of Electrical and Computer Engineering. Furthermore, a big thanks to two special friends, with his own brand of humor, to my long term friend and already a brother in life João Cascalho, and to Marta Barroqueiro for all the encouragement and constant understanding.

Most importantly, none of this could have happened without my family, my deepest gratitude to my parents, brother, aunt Odete Costa and uncle João Catarino for their endless love and support.

Abstract

The success of virtual reality comes from the sense of immersion it offers, providing sensory experiences that replicate the real world. This feeling is further extended when we are able to simulate a user's physical presence by providing intuitive ways of interaction. Our hands being a natural tool for real world interaction, interest in solutions that explore the tracking and recognition of hand motions grows. Using a hand tracking solution as an interactive tool can provide a multitude of advantages, not only on virtual reality but on any kind of human-computer interaction system.

However, most of the existing systems are high-cost solutions or exhibit limitations that either hinder the natural hand movements or have limited range of action, constraining the set of possible interactions.

This dissertation proposes a low-cost, ergonomic and wireless solution based on MEMS inertial sensors. This system fuses orientation data from gyroscopes, accelerometers and magnetometers in a developed low-complexity filter, known as Complementary Filter. To guarantee the performance of the low-cost sensors, a study was performed describing the motion sensors disturbances and calibration solutions to mitigate these effects. On our approach, the sensors are placed on specific places of the hand, to obtain hand and finger pose tracking relative to a reference frame on the wrist. This solution aims to be used in conjunction with Microsoft's Kinect in order to complement its skeleton pose tracking capabilities.

The proposed filter is compared to classical algorithms and the performance evaluation compared to a well-known commercial solution. The results demonstrate that the implementation provides an acceptable accuracy and is even comparable to more complex filters. This implementation is kept with low computational requirements, making it possible to develop a battery-operated/wearable device where computational power is limited.

With a fully working sensor fusion solution, the developed work during this dissertation culminated in a comfortable-to-use prototype solution and intends to provide a supportive tool extending the set of possible interactions within a virtual reality scenario. This way, we expect to improve a user's experience in a virtual experience by adding the feeling of full virtual hand ownership.

Keywords: Virtual Reality, Hand-tracker, Inertial Sensors, Complementary Filter, Wearable Device, Virtual Hand.

Resumo

O sucesso da realidade virtual advém da sensação de realidade que esta nos proporciona, oferecendo experiências inovadoras e imersivas. Esta sensação de imersão é ampliada quando somos capazes de simular a presença física de um utilizador, oferecendo formas intuitivas de interacção. Uma vez que as mãos são a nossa ferramenta natural para interagir com o mundo exterior, o seguimento e reconhecimento de movimentos das mãos oferece uma solução de interacção que nos é intuitiva e natural. A utilização de uma ferramenta que permita replicar fielmente os movimentos das mãos apresenta vantagens notórias, não só em realidade virtual mas em qualquer sistema de interacção homem-computador.

Contudo, os sistemas existentes de seguimento de mãos têm um preço elevado ou apresentam restrições que limitam a quantidade de possíveis interacções no ambiente virtual. Normalmente estes sistemas, se forem dispositivos vestíveis, apresentam formatos pouco ergonómicos que acabam por constringir movimentos naturais das mãos, ou se forem dispositivos que façam o seguimento à distância, acabam por estar limitados no alcance do seguimento.

Neste trabalho propomos uma solução de baixo custo, ergonómica e sem fios utilizando sensores inerciais baseados em sistemas Microelectromecânicos. A nossa solução obtém dados de orientação de sensores como giroscópios, acelerómetros e magnetómetros, utilizando um filtro de fusão sensorial de baixa complexidade, conhecido como Filtro Complementar. Para garantir o desempenho de sensores inerciais de baixo custo, é feito um estudo sobre o ruído associado a estes sensores, de forma a compor métodos de calibração que reduzam estes efeitos. Na nossa abordagem colocamos estes sensores em pontos específicos da mão, de forma a obter informação da sua pose relativo a um sistema de referência local no pulso. Esta solução tem o intuito de ser utilizada juntamente com o sensor de movimentos da Microsoft, Kinect, e complementar as suas capacidades de seguimento do corpo.

O filtro de baixa complexidade desenvolvido é comparado com algoritmos clássicos de fusão sensorial e o seu desempenho avaliado em relação a um produto comercial de referência na área. Os resultados obtidos demonstram um bom comportamento por parte do filtro, sendo até equiparáveis a filtros de maior complexidade. Com a nossa implementação é

possível o desenvolvimento de plataformas onde o poder computacional é limitado, devido aos requisitos computacionais baixos exigidos pelo filtro.

Sendo o objectivo desta solução melhorar as formas de interacção num cenário virtual, com trabalho produzido ao longo desta dissertação, foi desenvolvido um protótipo de seguimento da mão confortável de usar. Com esta nova ferramenta é esperado um aumento da sensação de imersão atingido num ambiente de realidade virtual.

Palavras-Chave: Realidade Virtual, Seguimento da Mão, Sensores Inerciais, Filtro Complementar, Dispositivo Vestível.

Table of contents

List of figures

List of symbols

Cartesian Coordinate Frames

${}^{A}, {}^{B}$	Generic coordinate frames
^{E}	Earth Frame (Reference Frame)
^{S}	Sensor Frame (Body Frame)

Scalars

K_i	Mahony gyroscope bias drift control gain (Integral gain)
K_p	Mahony filter cutoff frequency gain (Proportional gain)
q_s	Quaternion scalar component
q_x, q_y, q_z	Quaternion vector components
t	Step
ς	Static detector operator

Vectors

a	Measured acceleration
b	Ideal Earth's magnetic field
g	Gravity acceleration
h	Earth's magnetic field from orientation estimate
l	Generic vector
m	Measured magnetic field

q Orientation quaternion

v Noise measures

ω Measured angular velocity

ω_e Angular error

Ω Data Fusion

Matrices

A Accelerometer calibration parameters

E Transformation Matrix representing an end factor

G Gyroscope calibration parameters

K Scale factor

R Rotation matrix

T Transformation Matrix

T_a, T_g Accelerometer and Gyroscope Misalignment correction

X Calibration parameters

W Average of collected data

Y Known data

Other symbols

$\otimes$ Quaternion product

$\times$ Cross product

$\hat{}$ Normalized quantity

$\tilde{}$ Estimated quantity

$\cdot$ Derivative

$\bar{}$ Average

List of acronyms

2D	Two-dimensional
3D	Three-dimensional
AHRS	Attitude and Heading Reference System
CMC	Carpometacarpal
CMOS	Complementary Metal–Oxide Semiconductor
DIP	Distal Interphalanges
DMP	Digital Motion Processing
DoF	Degrees of Freedom
DP	Distal Phalanges
ECF	Explicit Complementary Filter
FFC	Flexible flat cable
FPS	Frames per Second
GDCF	Gradient Descent Complementary Filter
HMD	Head-Mounted Display
HU	Hardware Units
IMC	Intermetacarpal
IMU	Inertial Measurement Unit
INS	Inertial Navigation System

IP Interphalangeal

IR Infrared

LiPo Lithium Polymer

LED Light-Emitting Diode

LSB Least significant bit

MC Metacarpal

MCP Metacarpophalangeal

MEMS Microelectromechanical System

MP Medial Phalanges

OpenCV Open Source Computer Vision

OpenGL Open Graphics Library

PCB Printed Circuit Board

PI Proportional-Integral

PIP Proximal Interphalangeal

PP Proximal Phalanges

SDK Software Development Kit

SMT Surface-Mount Technology

TV Television

UDP User Datagram Protocol

VR Virtual Reality

Chapter 1

Introduction

1.1 Motivation, Goals and Contributions

Nowadays, with the constant development and improvement in technology, new ways for people to interact with virtual environments have been proposed. Even though virtual reality has been around since the 1960s, it regained popularity after the release of the Oculus Rift[1] Head-Mounted Display (HMD), receiving recently a large focus from researchers and developers alike.

Virtual reality is the term used to describe a 3D, computer generated environment which can be explored by a person, using realistic sensations to replicate the real world. Virtual realities artificially create sensory experiences (including sight, touch, hearing, and, less commonly, smell) and simulate a user's physical presence to enable interactions with this space, resulting in a sense of immersion that feels authentic. This sense of authenticity is further extended when the actions made by a user are faithfully replicated in this virtual world, specially hand movements and interactions.

Humans are skilful users of their hands, being our primary sensory organ, we use them to grasp and manipulate objects to complete daily tasks, as well as to communicate with, more or less, explicit gestural language. Analogously, the hands are our natural tools to explore the world around us, making them perfect candidates to use as interfaces between the real and virtual world. The tracking and recognition of detailed hand motions offer the possibility of building a natural and intuitive controller solution that target not only virtual reality but any kind of human-computer interaction system.

Hand tracking solutions are getting to a point where the accuracy is such that a user can start to feel like the avatar hand is their real hand. This illusion of body ownership was

[1] Product details available on: www.oculus.com

studied by Ehrsson et al. [1] on the "rubber hand illusion", their experiments showed that, with adequate synchronism and enough stimuli between a subject hand and a fake rubber hand, the subjects would begin to perceive the rubber hand as their own.

With this in mind, the aim of this work is to develop a hand tracking solution to enable a new way of interaction with 3D content (targeting virtual reality) and improve the user experience. More in depth, to develop an ergonomic solution, that does not hinder the natural hand movements, with finger motion tracking capabilities and as inexpensive as possible.

Succeeding previous research in the use of virtual reality as an immersive interface which explores this technology potential in many contexts, such as training simulators [2], for remote operation of robots [3][4] and as support in psychological therapies [5][6]. The Hand-Tracker developed in this dissertation intends to provide a supportive tool to further extend the available experimental scenarios and the sense of immersion they offer.

1.2 State-of-the-art

In virtual reality HMDs, the 3D environment is displayed to a user in a stereoscopic display. The perspective which the software shows the user depends on the position of his/her gaze, obtained by tracking the HMD orientation. This need to know the user's head position resulted in the research and development of different tracker technologies. The first HMD which included tracking capabilities was "The Sword of Democles", created in 1968 [7]. This was still a primitive and mechanical device, in order to track the head movements the HMD needed to be attached to a mechanical arm suspended from the ceiling of the laboratory, using servo or stepping motors to acquire the measurements.

Since then many tracking devices emerged in different areas, using various technologies. Nowadays, there are technologies such as the Vicon Motion Capture Systems[2], generally used in film-making and video game development, these make use of optical-passive techniques where retro-reflective markers are tracked by infrared cameras, recording the movement of objects or people. On the other hand, systems like the Optotrak Certus[3] use optical-active techniques that triangulate positions using markers that emit their own light instead, widely used by researchers in the fields of medicine and biomechanic sciences. Another reference example are the Xsens products, which allow for motion capture using miniature MEMS based motion trackers (IMU, AHRS and GPS/INS).

[2]Further details available on: www.vicon.com

[3]Available on: certus.ndigital.com

1.2.1 Tracking Devices for Consumer Electronics

Even though these technologies provide body motion capture, they are not really suitable to be used for hand and finger tracking. However, there already some available technologies for consumers that are relevant in the context of this work:

Glove-Based Systems

Numerous glove designs were proposed over the past 30 years, typically these systems are composed of a cloth glove made of Lycra with sensors sewn into it. A survey of glove-based systems is presented by Dipietro et al. [8], these glove designs use various technologies but all present the same basic concepts, to measure the finger joints bending and all use a cloth for supporting the sensors. Earlier gloves used mechanical trackers to measure the joint bending, whereas more recent models include non-contact position measurement devices (typically magnetic, ultrasonic, or optical). Some typical examples of these technologies include: hall-effect sensors in the Humanglove, commercialized by Humanware Srl; optical-fiber flex sensors in the 5DT Data Glove, commercialized by Fifth Dimension Technologies; flex and flexi-force sensors like the Smart Glove present in [9]; goniometric sensors like in the PERCRO data-glove [10]; 9-DoF orientation sensors (inertial sensors with magnetometers) and vibro-tactile actuators like VMG data gloves[4], commercialized by Virtual Motion Labs.

Major limitations originated from the cloth support, which not only acts as a constraint on the user's hand, but also affect measurements repeatability. Not only that, but these devices usually need tedious user-specific calibration procedures and the overall measurement performance is influenced by the sensor support and the quality of its fit to the user's hand.

Game Controllers

Game controllers were not developed with the intention to replicate the human hand in a virtual environment but they were the first kind of controllers that offered the possibility of motion tracking and gesture identification presented to common consumers. Game controllers like the Nintendo Wii remote, PlayStation Move or Razer Hydra allow position tracking and navigation through multimedia content, presenting technology that can easily be used for hand tracking.

The Nintendo Wii remote is equipped with a 3-axes accelerometer (providing the tilt of the remote and the amount of force applied to it in respect to gravity) and an Infrared (IR) camera sensor that, using a sensor bar (with 4 IR light sources) as reference point, allows to obtain position tracking through triangulation, as well as a rough orientation estimate. An

[4] Available on: www.virtualmotionlabs.com

extension was released, the Wii Remote Plus, adding 3 gyroscopes which further increased the precision of the orientation tracking.

The PlayStation Move, developed by Sony, contains a 3-axes accelerometer, two gyroscopes (one with 2 axes and one with a single axis), a temperature sensor and a 3-axes magnetometer. In combination with the PlayStation 3 Eye (a 2.0 USB camera allowing position tracking of the controller) this system allowed for an even more precise tracking than the Nintendo Wii remote.

The Razer Hydra[5] uses a base station which emits a weak magnetic field, the motion tracking is executed by reading this field allowing to detect the absolute position and orientation of the controllers. With a declared resolution of 1 millimetre and 1 degree when closer than 0.5 meters from the base station, this tracker works extremely well in good conditions, however outside this range the tracking becomes unstable.

Microsoft's Kinect

The Kinect sensor incorporates a depth camera, a colour camera and a four-microphone array that provide full-body 3D motion capture, facial recognition, and voice recognition capabilities. The depth sensor consists of an IR laser projector combined with a monochrome CMOS sensor, which captures video data in 3D. Together with the RGB camera and skeletal tracking software, the Kinect performs formidably well in human body tracking and body gesture recognition [11]. Compared to the entire human body, the hand is a smaller object with more complex articulations and more easily affected by segmentation and occlusion errors, nevertheless many researches have demonstrated the capabilities of this technology as a hand tracking solution. Particularly the Microsoft's Handpose system presents an algorithm that can accurately reconstruct complex hand poses and allows for robust tracking. The system still presents some limitations and failed cases due to the previously mentions errors, but also presents solutions where the tracking rapidly recovers from these temporary failures [12].

Leap Motion

The leap motion controller is a small peripheral device which is designed to be placed, facing upward, on a physical desktop or mounted onto a virtual reality headset. Leap supports hand and finger motions as input, delivering positions in Cartesian space of predefined objects like finger tips, pen tip, etc. It uses IR optics and cameras instead of depth sensors, and does not cover as large an area as Microsoft's Kinect sensor (the device only observes a

[5]Product details available on: www.razerzone.com

roughly hemispherical area, to a distance of about 1 meter). Leap does its motion sensing at a fidelity unmatched by any depth camera currently available, tracking all 10 of your fingers simultaneously to within a hundredth of a millimetre [13].

Virtual Reality Controllers

With the recent release of virtual reality HMDs for consumers, VR specific controllers are emerging on the market. The two most relevant system at the moment are the HTC Vive controllers, which are the official input devices for the HTC Vive HMD, and the unreleased Oculus Touch for the Oculus Rift.

HTC Vive controllers use the same tracking system as the Vive HMD[6], this system is called Lighthouse. Lighthouse is defined as a laser-based inside-out positional tracking system, the rotational tracking is achieved with Inertial Measurement Units (IMUs), while positional tracking is accomplish with two lighthouse Base Stations. These Base Stations work as reference points, constantly flooding the room with IR light, the controllers are covered with photo-sensors that recognize this IR light and calculate the appropriate position. This system allows freedom of movement in spaces up to 4.6 meters by 4.6 meters and is optimized for sub-millimetre positional tracking with latency counted in single milliseconds. The main limitation of this technology is that, since the controllers require their users to firmly grip them the whole time they are using them, it is not possible to perform any gestures that involve fingers and thumbs.

Oculus Touch[7] achieves high precision, low latency and 6 DoF tracking through the same Constellation tracking system as the Rift headset, defined as an optical-based outside-in positional tracking system. Similar to Lighthouse, rotational tracking is achieved with IMUs, whereas positional tracking is accomplish with an external camera sensor. Tiny LED markers are placed on the controllers bodies and the camera sensor is able to recognize these markers, tracking their positions. Additionally, this system claims to have hand presence and gesture sensing, the grip of the controller feels natural (similar to shaking someone's hand) and with a matrix of sensors mounted onto the device, it allows to keep track of some fingers and detect hand gestures.

1.2.2 Inertial Sensing and Related Estimation Techniques

Having taken into consideration all these technologies, as well as their advantages and handicaps, in this work we use inertial and magnetic field sensors as the main tracking

[6] HTC and Valve virtual reality systems available at: www.htcvive.com/eu/

[7] Product details available at: www3.oculus.com/en-us/touch/

technology. This choice was taken in order to maintain the balance between the requirements of the hand tracker functionalities and the need for it to be a comfortable-to-use device. Nowadays, these types of sensors are embedded in very small packages (IMUs) allowing for very simplistic and ergonomic designs for any kind of product. Also, taking a look at the state of the art literature in section 1.2, technologies like the ones used for the gloves systems (for example, flex sensors or goniometric sensors) restrain the user's movement and technologies that rely on line of sight (for example, depth cameras or IR laser) present limitations in their range of action and possible occlusion between the fingers.

This approach does not come without its drawbacks, IMUs offer a precise orientation tracking but lack the stability to provide accurate position tracking for more than a few seconds [14]. Given the context of research in which this dissertation is integrated, many of our interactive scenarios already use the Kinect sensor in order to track the users body. In this way, we propose to use the body tracking capabilities of the Kinect as an external auxiliary system to obtain the position estimation of the hand, specifically the kinematic joint[8] associated with the wrist.

In order to mitigate sensor errors and obtain a correct orientation estimate, the sensors data has to be fused through an algorithm, usually referred as a filter. In this regard, we discuss three major approaches developed in the last years, Stochastic Filters (commonly Kalman Filters), Particle Filters and Complementary Filters.

Kalman Filters

The Kalman filter is one of the most celebrated and popular data fusion algorithms in the field of information processing. Due to its applicability to a wide range of fields, extensions and generalizations to the method have also been developed allowing its usage for non-linear problems. Since orientation determination is intrinsically a non-linear problem, Kalman filtering based estimation techniques are usually employed in this regard and are considered a de-facto standard. Most common implementations are the Extended Kalman Filter [15] and Unscented Kalman Filter [16].

The problem associated with Kalman based algorithms is the demand for the computational complexity, due to intensive matrix operations and Jacobian matrices computation, which may be computationally expensive for some applications [17]. Still, few filters can accomplish the same levels of accuracy, making these implementations the preferred solution for professional use in motion tracking applications. In the context of this work, the MTi

[8] Further details on Kinematics presented is in section 4.1)

AHRS[9], employing a proprietary XSens Kalman filter, was used as reference for results validation.

Particle Filters

Particle Filters are recursive implementations of Monte Carlo based statistical signal processing [18]. Its methodology consists in estimating the internal states in dynamical systems when partial observations are made, and random perturbations are present in the sensors as well as in the dynamical system. The particle filters are seen as a serious alternative for real-time applications classically approached by model-based Kalman filter techniques. These methods have the great advantage of not being subject to any linear or Gaussian constrains on the model, and they also have engaging convergence properties. Some works that use these filters can be easily found in the literature, for instance [19][20] present experiments that show a clear improvement in performance compared to the conventional Extended Kalman Filter. However, these filters employed in the regard of orientation estimation are associated with an even more demanding computational burden than the Kalman Filter[21].

Complementary Filters

Complementary Filters work in the frequency domain, where some signals pass through a low-pass filter and others signals through its complementary high-pass filter, combining both signals in the end. The search for simple and still effective filtering algorithm lead to the development of these constant gain based complimentary algorithms. These techniques have been shown to offer efficient performance with little computational cost, the drawback is that, complementary filters do not contemplate adaptability, making parameters like the sensor error lost during the filter process. However, filters such as the explicit complementary filter (ECF) suggested by Mahony et al. [22] and gradient descent based complementary filter (GDCF) authored by Madgwick et al. [23] have been shown to implement ways to dynamically adapt the filter, to estimate sensors bias, and respond to readings deviations. Both ECF and GDCF are effective and novel approaches in this regard, with the power of adjustable gain, these techniques find places in most of the real world applications. Research studies [24] have demonstrated that the evaluation of these filter results in identical outcome, however, ECF has a bit edge over GDCF with slightly higher accuracy and partly because of the two adjustable gains resulting in extra choices (GDCF only has a single adjustable parameter). Moreover, ECF is less sensitive to variation in filter gain in comparison to GDCF.

[9]Commercialized by Xsens, further details on: www.xsens.com

1.3 Approach and Dissertation structure

Even though the Kalman filter has become the accepted basis for the majority of orientation algorithms, being their widespread use a testament to their accuracy and effectiveness, they can be complicated to implement and may demand a large computational load. In embedded systems where portability is critical, the challenges presented by Kalman-based solutions provide a clear motivation for alternative approaches. Particle filters do not provide a viable solution and can in fact be even more demanding. In order to keep computational requirements as low as possible while maximizing available resources, in this work an adaptation of the ECF is implemented and tested.

In this ways the outline of this dissertation is organized as follows: Chapter 2 includes a characterization of the motion sensors used, its disturbances and calibration solutions to mitigate their effects, introduces quaternion representation and presents the implemented sensor fusion filter for orientation estimation; Chapter 3 evaluates the performance of the implemented fusion filter algorithm (ECF); Chapter 4 describes an overview of the Kinematic constrains, depicts the developed Hand-Tracker hardware and illustrates the software implementation that validates the overall system; finally, in Chapter 5 we reflect upon the completion of our objectives and on possible future work.

Chapter 2

Supporting Technologies and Background

This chapter presents the relevant supporting technologies, as well as the relevant background about the implemented orientation filter. In the first section the inertial and magnetic field sensors technologies are introduced together with their respective calibration requirements. The following section presents a brief review of the orientation representation using quaternions, in order to allow a better understanding of the topics ahead. The final section analysis orientation estimation techniques, emphasizing the studied low-complexity Complementary filter.

2.1 Motion Sensors

Motion tracking sensors have had a wide range of applications, mostly industrial such as the automotive or aircraft industry, however, in recent years these sensors received a high miniaturization treatment and are rapidly becoming a common technology in many consumer electronic devices. Consumers are now able to interact in intuitive ways using motions as input commands with their smarthphones, gaming consoles, tablets and even smart TVs. This was made possible due to the introduction of MEMS technology, we now have small chips that combine a 3-axis gyroscope, a 3-axis accelerometer and 3-axis magnetometer all in the same package. These chips, often referred to as IMUs, being extremely low-cost, lightweight and compact (reaching sizes of 3x3x1mm) are not without their flaws, the measurements they give are relatively noisy, can have offset bias, different scale factors, non-orthogonality between axes and, due to the fabrication process, the characteristics of individual sensors may be altered if not properly handled in the soldering process. These problems led to

the development of highly rigorous calibration and assemble procedures, providing IMU solutions which are extremely accurate but suffer from a huge cost increase. According to the application and precision needed, simple calibration methods may suffice providing an affordable and reliable solution. In this section we analyse these three types of sensors as well as their respective calibration procedures.

2.1.1 MEMS Gyroscope

MEMS gyroscopes are sensors capable of measuring the rate of rotation of a body. Based upon simple mechanical principles, these sensors exploit the Coriolis effect of a vibrating structure. When a vibrating structure is rotated, a secondary vibration is induced from which the angular velocity can be calculated [25].

As previously stated the disadvantage of using MEMS technology is that it is far less accurate, when handling these sensors there is a need to examine their error characteristics and the effect they have on the integrated orientation signal, in order to apply a proper calibration procedure. Typically, they present a non-zero offset at rest, in other words, a constant bias is present when the gyroscope is not undergoing any rotation. When integrated, this error causes an angular error which grows linearly with time. Additionally there occurs an offset drift over time, which is caused collectively by non accurate scaling, sensor axis misalignments, cross-axis sensitivities and even temperature effects. Although fusion algorithms (using gyroscopes and other sensors) could estimate this drift and correct it in real-time, these errors should be minimized to obtain higher accuracy. An appropriate calibration provides all the parameters needed to estimate and correct these errors.

2.1.1.1 Minimal usage calibration

In order to obtain meaningful information from cheap MEMS gyroscopes we assume a model for the gyroscope in its own reference frame, where the true angular rate one wants to measure, ${}^{S}\omega$, is given by [26]:

$$ {}^{S}\omega = K_g.({}^{S}\omega_{raw} - {}^{S}\omega_0) \tag{2.1} $$

- ${}^{S}\omega$ - Real angular velocity given in $^\circ/s$;
- K_g - Scale factor (or sensitivity) given in $(^\circ/s)/HU$;
- ${}^{S}\omega_{raw}$ - Raw readings of the gyroscope given in HU;
- ${}^{S}\omega_0$ - Average raw readings at rest given in HU.

This simple calibration allows to compensate for bias instability, this method uses the first few readings of the gyroscope, at least 50 to 100 samples, to subtract from the subsequent data. This has to be done with the sensor at rest, readings with the gyroscope in motion must be discarded. ω_0 is obtained by calculating the readings average for each gyroscope axis. The gyroscope scale factor, which converts hardware units into angular velocity units, differs with each sensor (usually given by the sensor datasheet) and should be calibrated accordingly. This calibration procedure takes little time (varies with chosen sampling rate for the sensor) and should occur after the gyroscope is powered on, so that the subsequent data becomes calibrated. The following figure represents an example of calibration with the available sensor board, for comparison purposes the scale factor K_g was included with the raw data.

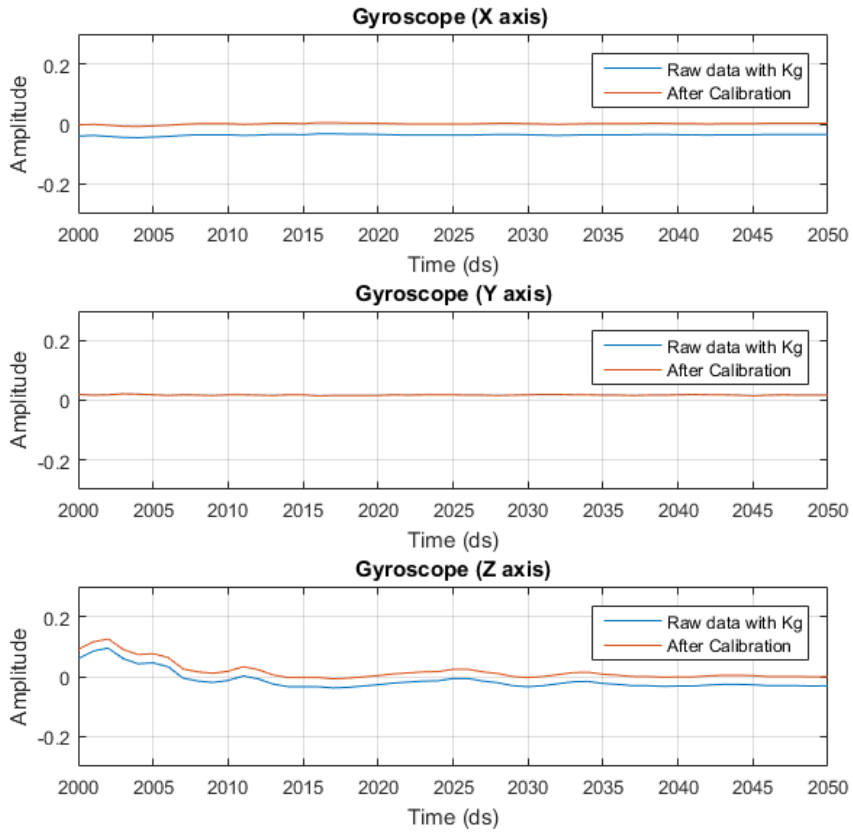

Fig. 2.1 Gyroscope calibration comparison example.

2.1.1.2 Extended Calibration

Further extending the previous calibration, after installing the sensor on the PCB and the PCB on the actual body of the device we want, there may exist axes misalignment between

the device body axes, the PCB and the actual sensor axes. If any misalignment is present, the angular velocity that is applied to one axis of the device will have projection on the other two axis of the device axes. Therefore, to correct misalignment an extended calibration is possible to use, but it is more difficult to implement, requiring the use of an external setup while knowing the real angular velocity during calibration. It is only necessary in highly sensitive applications, where the precise angular velocity is needed. To find the misalignment matrix to compensate the gyroscope measurements, the model presented in equation (2.1) is extended, becoming,

$$^{S}\omega = T_g K_g (^{S}\omega_{raw} - ^{S}\omega_0) \tag{2.2}$$

where T_g is the matrix that defines the misalignment between the sensor axes and the body axes of the device. Unfolding equation (2.2) to equation (2.3):

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \cdot \begin{bmatrix} \omega_{rx} - G_{10} \\ \omega_{ry} - G_{20} \\ \omega_{rz} - G_{30} \end{bmatrix} \tag{2.3}$$

The values of G_{mm} correspond to the calibration parameters needed to adjust the output data from the gyroscope. G_{10}, G_{20} and G_{30} are calculated, as in the previous method, by averaging samples with the sensor at rest. The remaining parameters may be determined by using a single-axis rate table or a step-motor spin table as shown in figure 2.2 [26].

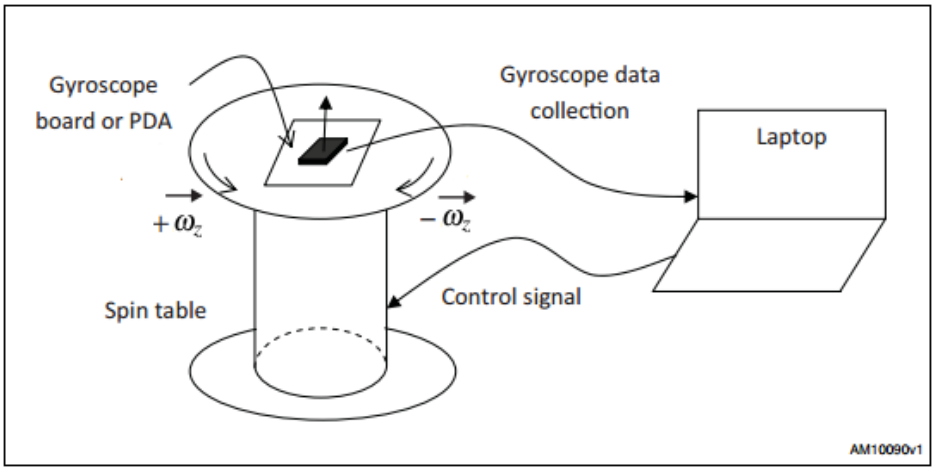

Fig. 2.2 Gyroscope calibration setup.

The objective of this setup is to know and control the real angular velocity while reading the angular velocity values from the gyroscope. Measurements should be done for each axis independently at two different speeds and in opposite directions. In order to apply the

Least Squares method to determine the calibration parameters, we can construct the known applied angular velocity data into a matrix and the average angular velocity obtained from the gyroscops's raw data into another matrix. This way we are able to construct equation (2.4):

$$Y_\omega = W_\omega \cdot X_\omega \tag{2.4}$$

- Y_ω - Know angular rate data in a 12x3 matrix;
- W_ω - Average of the collected data from the gyroscope;
- X_ω - Calibration matrix that needs to be determined.

Considering for example $50\,^\circ/s$ and $100\,^\circ/s$ as our two speeds, we can represent the same equation (2.4) as:

$$\begin{bmatrix} 50 & 0 & 0 \\ -50 & 0 & 0 \\ 100 & 0 & 0 \\ -100 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & -50 & 0 \\ 0 & 100 & 0 \\ 0 & -100 & 0 \\ 0 & 0 & 50 \\ 0 & 0 & -50 \\ 0 & 0 & 100 \\ 0 & 0 & -100 \end{bmatrix} = \begin{bmatrix} \omega_{rx1} & \omega_{ry1} & \omega_{rz1} \\ \omega_{rx2} & \omega_{ry2} & \omega_{rz2} \\ \omega_{rx3} & \omega_{ry3} & \omega_{rz3} \\ \omega_{rx4} & \omega_{ry4} & \omega_{rz4} \\ \omega_{rx5} & \omega_{ry5} & \omega_{rz5} \\ \omega_{rx6} & \omega_{rz6} & \omega_{rz6} \\ \omega_{rx7} & \omega_{ry7} & \omega_{rz8} \\ \omega_{rx8} & \omega_{ry8} & \omega_{rz8} \\ \omega_{rx9} & \omega_{ry9} & \omega_{rz9} \\ \omega_{rx10} & \omega_{ry10} & \omega_{rz10} \\ \omega_{rx11} & \omega_{ry11} & \omega_{rz11} \\ \omega_{rx12} & \omega_{rz12} & \omega_{rz12} \end{bmatrix} \cdot \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \tag{2.5}$$

Using the pseudo-inverse in equation (2.6) we can obtain the intended parameters for calibration.

$$X_\omega = \left[W_\omega^T . W_\omega\right]^{-1} . W_\omega^{-1} . Y_\omega \tag{2.6}$$

2.1.2 MEMS Accelerometer

An accelerometer is an electromechanical device that measures acceleration. These devices are able to sense static accelerations, like the constant force of gravity, as well as dynamic accelerations, caused by moving or vibrating the accelerometer. A typical MEMS accelerometer is composed of a movable proof mass with plates that is attached through a mechanical suspension system to a reference frame. These fixed and movable plates form various capacitors, when the geometry of a capacitor changes the variation in capacitance is detected. By using the capacitance difference we are able to measure the deflection of proof mass and therefore, with proper signal conditioning, translate it to acceleration [27].

These sensors are characterized by very accurate measurements, still for certain applications, calibration is needed to remove systematic measurement bias, which vary both with temperature and in time.

2.1.2.1 Calibration

Similarly to the gyroscope sensor there are calibrations for the accelerometer that can be done with minimal human intervention, simply by ensuring a rest position of the sensor (no external acceleration besides gravity). Like in the gyroscope calibration, assuming a model for the accelerometer in its own reference frame, the acceleration, ${}^{S}a$, one wants to measure, is given by the equation [28]:

$$ {}^{S}a = T_a K_a ({}^{S}a_{raw} - {}^{S}a_0) \tag{2.7} $$

- ${}^{S}a$ - Acceleration given in g ($1g = 9.81 m/s^2$);
- T_a - Misalignment matrix;
- K_a - Scale factor (or sensitivity) given in g/HU;
- ${}^{S}a_{raw}$ - Raw readings of the accelerometer given in HU;
- ${}^{S}a_0$ - Average raw readings at rest given in HU.

Unfolding equation 2.7 results in:

$$ \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} . \begin{bmatrix} a_{rx} - A_{10} \\ a_{ry} - A_{20} \\ a_{rz} - A_{30} \end{bmatrix} \tag{2.8} $$

The values of A_{mm} correspond to the calibration parameters needed to adjust the output data from the accelerometer. The method used for calibration consists in placing the sensor

at six stationary positions, in all three orthogonal directions, while collecting the raw values in order to calculate the calibration matrix. Equation 2.8 can be written as:

$$\begin{bmatrix} a_x & a_y & a_z \end{bmatrix} = \begin{bmatrix} a_{rx} & a_{ry} & a_{rz} & -1 \end{bmatrix} . \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{10} & A_{20} & A_{30} \end{bmatrix} \tag{2.9}$$

Labelling the above matrices:

$$Y_a = W_a . X_a \tag{2.10}$$

- Y_a - Known normalized gravity vector;
- W_a - Collected data from the 6 stationary positions;
- X_a - Calibration matrix that needs to be determined.

Considering the ideal output values for the 6 calibration positions the following equation system is obtained:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} a_{rx1} & a_{ry1} & a_{rz1} & -1 \\ a_{rx2} & a_{ry2} & a_{rz2} & -1 \\ a_{rx3} & a_{ry3} & a_{rz3} & -1 \\ a_{rx4} & a_{ry4} & a_{rz4} & -1 \\ a_{rx5} & a_{ry5} & a_{rz5} & -1 \\ a_{rx6} & a_{rz6} & a_{rz6} & -1 \end{bmatrix} . \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{10} & A_{20} & A_{30} \end{bmatrix} \tag{2.11}$$

Where a_{rxn}, a_{ryn} and a_{rzn}, (n = 1, 2, 3, 4, 5, 6), correspond to the readings average of each axis, in each of the six positions.

To obtain the intended calibration matrix, the calibration parameter X_a, can be determined by the least square method as:

$$X_a = \left[W_a^T . W_a \right]^{-1} . W_a^{-1} . Y_a \tag{2.12}$$

The following figure represents an example of calibration with the available sensor board, for comparison purposes the scale factor K_a was included with the raw data.

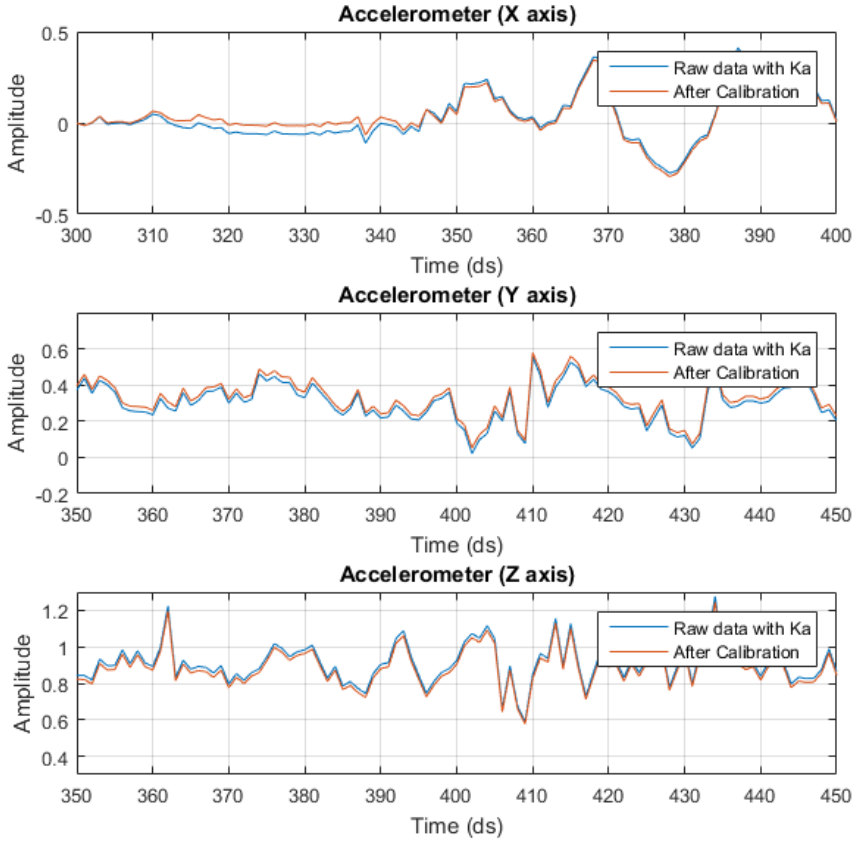

Fig. 2.3 Accelerometer calibration comparison example.

2.1.3 Robust Calibration using both Gyroscope and Accelerometer

Alberto Pretto and Giorgio Grisetti [29] propose a calibration method for IMUs without the use of any external equipment, providing calibration parameters for both the gyroscope and accelerometer. Using features from both sensors they are able to minimize existing errors and estimate calibration parameters between the sensors, such as misalignment and scaling factors as well as sensor biases.

Their procedure for data acquisition employs placing the IMU in multiple static positions between small intervals of time. Starting with a small period with the sensor at rest (for example 30 seconds), the user should move the IMU in different positions (maintaining the sensor still in those positions for a fixed time, between 1 to 4 seconds) as many times as

needed in order to obtain at least 36 distinct static orientations. This protocol can be executed by hand and is described in figure 2.4.

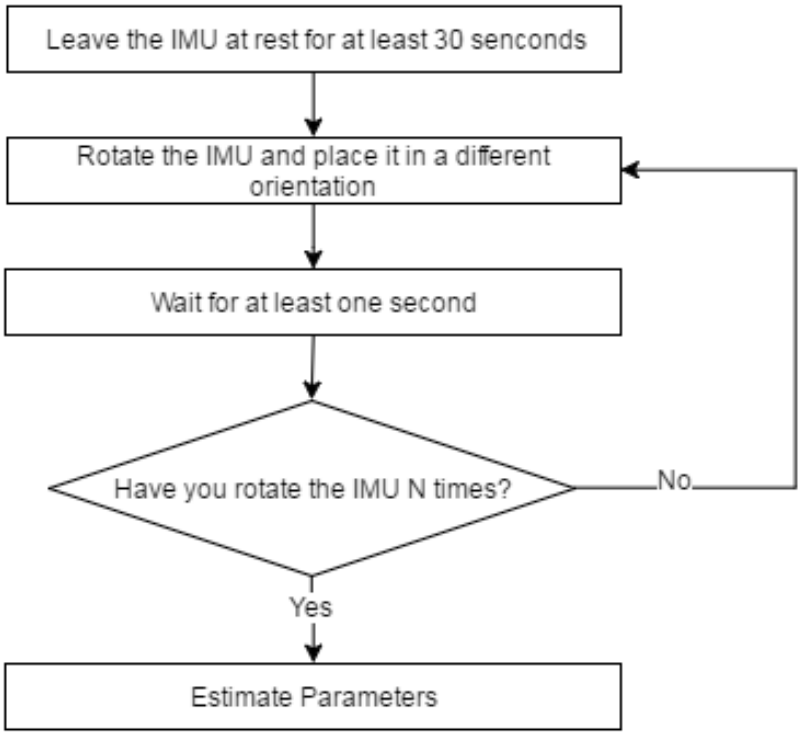

Fig. 2.4 Calibration protocol flowchart.

This calibration explores previous studies on accelerometers calibrations which consider that, in a static position, the norms of the measured accelerations is equal to the magnitudes of the gravity plus multiple error factors and, with a set of multiple static attitudes, these factors can be estimated through minimization.

For this calibration, misalignments between the accelerometer and gyroscope frames (due to assembly inaccuracy) is considered and compensated by means of two matrices, T_a and T_g respectively. The accelerometer frame is used as frame of reference for the gyroscope misalignment matrix. As in previous calibration methods the scaling matrices K_a and K_g, and the bias vectors ${}^S a_0$ and ${}^S \omega_0$ are estimated. The complete error models for the sensors are:

$$ {}^S a = T_a K_a ({}^S a_{raw} - {}^S a_0 - {}^S v_a) \tag{2.13} $$

$$ {}^S \omega = T_g K_g ({}^S \omega_{raw} - {}^S \omega_0 - {}^S v_g) \tag{2.14} $$

where ${}^S v_a$ and ${}^S v_g$ are the accelerometer measurement noise and the gyroscope measurement noise, respectively.

Classifying correctly static and motion intervals heavily influences the accuracy of this calibration, since the accelerometer only considers static intervals for calibration whereas, for the gyroscope, motion intervals are also included. A variance based static detector operator using the accelerometer signals is proposed. For each accelerometer sample (a_x, a_y, a_z) at time t, the magnitude of the variance is computed as:

$$\varsigma_t = \sqrt{[var_{t_w}(a_x)]^2 + [var_{t_w}(a_y)]^2 + [var_{t_w}(a_z)]^2} \tag{2.15}$$

where $var_{t_w}(a)$ is an operator that computes the variance of the accelerometer signal in a time interval of length t_w seconds centered in time t. Using this operator, a static interval is considered if the variance magnitude is lower than a certain threshold (computed during the initialization period).

To estimate the accelerometers parameters a cost function is used:

$$L(\theta_a) = \sum_{k=1}^{M} (||g||^2 - ||h({}^S a_k, \theta_a)||^2)^2 \tag{2.16}$$

where M is the number of static intervals, θ_a is the vector containing the unknown parameters, $||g||$ is the actual magnitude of the local gravity vector (can easily be recovered from specific public tables) and h is a function that corrects the readings applying the current parameter vector. To minimize equation 2.16 the *Levenberg-Marquardt* algorithm is employed [30].

With the accelerometer calibrated in this way, and using the same set of multiple attitudes, the gyroscope is calibrated using the corrected gravity vector (obtained from the accelerometer) as a reference between the different orientations. The angular velocities from the gyroscope are integrated to estimate the gravity vector orientation, and by minimizing the errors between the two gravity estimates it is possible to estimate the wanted calibration parameters, θ_g. In this case, the cost function becomes:

$$L(\theta_g) = \sum_{k=2}^{M} ||u_{a,k} - u_{g,k}||^2 \tag{2.17}$$

where $u_{a,k}$ is the acceleration versor measured averaging in a temporal window the calibrated accelerometer readings in the *k-th* static interval and $u_{g,k}$ is the acceleration versor computed integrating the angular velocities between the $k-1$-*th* and the *k-th* static intervals, as described in [31]. Once again the *Levenberg-Marquardt* algorithm is employed to minimize equation 2.17.

This calibration procedure takes advantage of the combination of gyroscopes and accelerometers on IMUs, providing a reliable and more robust calibration solution.

2.1.4 MEMS Magnetometer

The most common MEMS magnetometers are Hall-effect transducers with a magnetic concentrator. Ideally, a magnetometer would provide information about the vector pointing North, nevertheless the earth magnetic field does not point North, as a matter of fact it is not even constant over time (changing over the years). Depending on where we are on the planet, the magnitude of the field over the surface of the Earth varies as well as the declination angle between the magnetic North and the geographic North. Be that as it may, these characteristics of the Earth's magnetic field do not represent a problem for indoor navigation, as the objective of using a magnetometer is to support and prevent deviations on the navigation provided by other sensors. Problems in indoor navigation arise on the measurements of the local magnetic field, which is usually composed by the sum of multiple magnetic fields, that exist on the local frame of the sensor, plus the Earth's magnetic field.

Furthermore, the measurements of the magnetic field obtained with low cost sensors are corrupted by several errors including sensor fabrication issues and the magnetic deviations induced by the host platform.

2.1.4.1 Calibration - Magnetic Distortions Compensation

Proper calibration of the magnetometers is required to achieve high accuracy measurements, generally performed by means of experimentation and calibration parameters estimation techniques. Several procedures and algorithms have been proposed to perform the calibration [32]. However their performances often rely on assumptions that constraint the type of the errors in the measurements and ignore some critical components.

We propose a simplified model of calibration to deal only with unwanted or interfering magnetic fields. This way we do not take into consideration some characteristic on the error modeling of a magnetometer readings, characteristic that should be factory calibrated, such as the corruption of the output of magnetometers by wide band measurement noise, stochastic biases due to sensor imperfections or installation errors.

The unwanted or interfering magnetic fields can be classified into two distinct groups. The first group consists of constant or slowly time-varying fields generated by ferromagnetic structural materials in the proximity of the magnetometers. The field measurement errors resulting from such interferences are referred to as hard iron biases. The second group of interfering magnetic fields result from materials that generate their own magnetic field in response to an externally applied field. This generated field is affected by both the magnitude and direction of the externally applied magnetic field. Such materials are called soft irons and the error they generate is referred to as a soft iron bias. In summary, although these

disturbances are not completely decoupled, it is possible to somewhat relate the hard iron distortions with the device and the soft iron distortions with the environment where the device is located [33].

The calibration method implemented implies the rotation of the sensor platform in complete circles around it self, in a way that allows us to gather many points of an ellipsoid. Figure 2.5 shows a 3 Dimensions (3D) representation of the effects in the magnetic field measurements, for these types of distortions. The combination of both distortions in 3D cause the expected perfect and origin-located sphere to be distorted into a displaced ellipsoid.

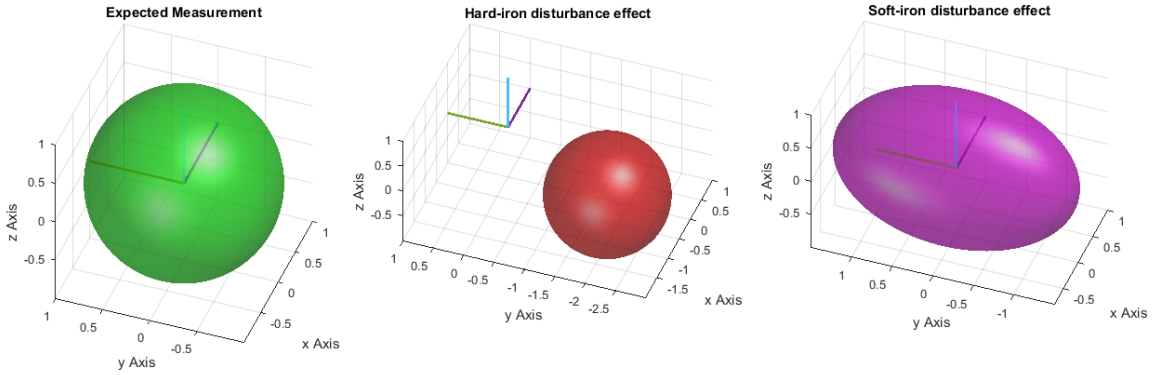

Fig. 2.5 Disturbances to Earth's magnetic field readings.

Soft and hard-iron distortions, are compensated using a simple method, taking into consideration the following model of the sensor:

$$^{S}m = R_m K_m (^{S}m_{raw} - ^{S}m_0) \tag{2.18}$$

- ^{S}m - Corrected measurement of the magnetic field;
- $^{S}m_{raw}$ - Raw data from magnetometer;
- R_m - Orientation of axes correction;
- K_m - Parameters that compensate soft-iron effects.
- $^{S}m_0$ - Parameters that compensate hard-iron effects.

Where R_m is a rotation matrix used to align the axes of the magnetometer regarding the axes of both the gyroscope and accelerometer, most IMUs contain different orientation for the sensitivity axes of the magnetometer.

Compensating for hard-iron distortion is straightforward, recording some of the magnetometer data as the sensor is moved (for example in a figure eight pattern) and keep track of

the minimum and maximum field measured in each of the six principal directions. Once the minimum and maximum values along the three axes are known, the average can be subtracted from the subsequent data which amounts to re-centering the response surface on the origin.

$$ {}^{S}m_0 = \begin{bmatrix} \frac{max(m_x)+min(m_x)}{2} \\ \frac{max(m_y)+min(m_y)}{2} \\ \frac{max(m_z)+min(m_z)}{2} \end{bmatrix} \tag{2.19} $$

Compensating for soft-iron distortion is more compute-intensive than compensating for hard-iron distortion, and it may be more effective from a cost and efficiency perspective, particularly if designing or implementing an embedded system, to eliminate the soft-iron material(s) from the proximity of the sensor.

Still, taking the minimum and maximum values already computed it is possible to use them to rescale the magnetometer data to equalize the response along the three measurement axes. A scale factor can be calculated by taking the ratio of the average $(max - min)$ along each axis and the average of all three axes. This means that an axis where the $(max - min)$ is large has its magnetic field reduced and an axis that under-measures the field with respect to the other axes has its magnetic field values increased. This is just a simple orthogonal rescaling allowing some additional correction for scale bias.

$$ \alpha = \begin{bmatrix} \frac{max(m_x)-min(m_x)}{2} \\ \frac{max(m_y)-min(m_y)}{2} \\ \frac{max(m_z)-min(m_z)}{2} \end{bmatrix} \tag{2.20} $$

$$ K_m = \frac{\bar{\alpha}}{\alpha} \tag{2.21} $$

Figure 2.6 shows an example of the calibration of the magnetometer. As explained previously, the data is collected with rotational movements of the magnetometer in such a way, that it allows covering a large number of points on a sphere. The values of the raw sensor are 3D plotted and approximated to an ellipsoid (red in the figure). The expected perfect and origin-located sphere is also plotted to use as reference (green in figure), following the procedure described previously, the calibrated data readings (blue in figure) will tend to that sphere.

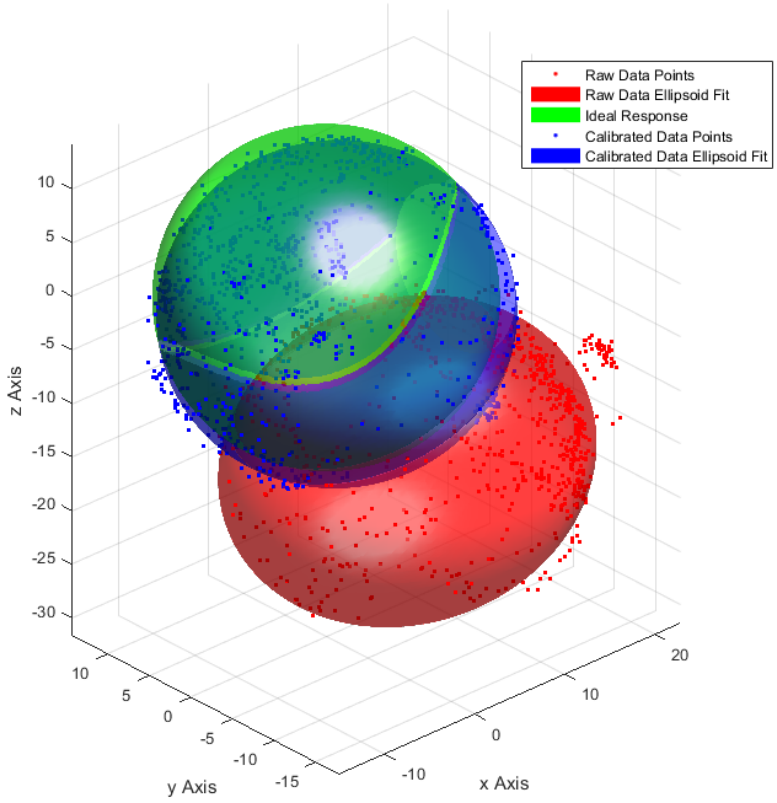

Fig. 2.6 Magnetometer calibration example.

2.2 Quaternions

There are various mathematical formulations to represent the orientation of a rigid body. In this work the quaternion representation is used. A quaternion, generally represented by q, is a four-element vector composed of one real element and three complex elements. When normalized, $\hat{q} = 1$, a quaternion can be used to encode any rotation in a 3D coordinate system. Considering a rigid body in a coordinate frame (B) relatively to any other generic coordinate frame (A), a quaternion ${}^{B}_{A}q$ can be used to represent the orientation of the body as in equation 2.22.

$$ {}^{B}_{A}q = \begin{bmatrix} q_s & q_x & q_y & q_z \end{bmatrix} \tag{2.22} $$

The elements q_x, q_y and q_z of the quaternion can be thought of as a vector about which rotation should be performed. The element q_s represents a scalar component that specifies the amount of rotation that should be performed about the vector part. Specifically, if θ is the angle of rotation and the vector $l = [l_x \quad l_y \quad l_z]$ is a unit vector representing the axes of

rotation, equation 2.22 can be written as:

$$ {}^{B}_{A}q = \begin{bmatrix} cos\left(\frac{\theta}{2}\right) & l_x \cdot sin\left(\frac{\theta}{2}\right) & l_y \cdot sin\left(\frac{\theta}{2}\right) & l_z \cdot sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{2.23} $$

This definition does not need to be used explicitly, but it is included here to provide an intuitive description of the quaternion representation.

The quaternions also have important properties that simplify calculation. As mentioned previously a quaternion only represents a rotation if it is normalized, the normalization procedure is presented in equation 2.24.

$$ \hat{q} = \frac{q}{||q||} = \frac{q}{\sqrt{q_s^2 + q_x^2 + q_y^2 + q_z^2}} \tag{2.24} $$

As long as $||q|| = 1$, the inverse quaternion is equivalent to the quaternion conjugate, a property which can be used to reverse rotations or swap between frames.

$$ q^* = \begin{bmatrix} q_s & -q_x & -q_y & -q_z \end{bmatrix} \tag{2.25} $$

$$ {}^{B}_{A}\hat{q}^{-1} = {}^{B}_{A}\hat{q}^* = {}^{A}_{B}\hat{q} \tag{2.26} $$

Another important property, that can be used to compute successive rotations, is the quaternion multiplication, presented as $\otimes$ in equation 2.27. This is a non-commutative operation, that is $q \otimes p \neq p \otimes q$.

$$ q \otimes p = \begin{bmatrix} q_s p_s - q_x p_x - q_y p_y - q_z p_z \\ q_s p_x + q_x p_s + q_y p_z - q_z p_y \\ q_s p_y - q_x p_z + q_y p_s + q_z p_x \\ q_s p_z + q_x p_y - q_y p_x + q_z p_s \end{bmatrix} \tag{2.27} $$

Also, a vector can be rotated, as in equation 2.28, where ${}^{A}l$ and ${}^{B}l$ are the same vector represented in two different frames.

$$ {}^{A}l = {}^{B}_{A}\hat{q} \otimes {}^{B}l \otimes {}^{B}_{A}\hat{q}^* \tag{2.28} $$

The choice of orientation representation depends on the application in question [34], additional information regarding the other representations, as well as advantages and disadvantages of each one that lead to the choice of quaternion representation over the others can be found in Appendix A.

2.3 Orientation Estimation Techniques

2.3.1 Tracking Orientation in an Inertial Navigation System

2.3.1.1 Strapdown Inertial Navigation

Inertial navigation is a self-contained navigation technique in which measurements provided by accelerometers and gyroscopes are used to track the position and orientation of an object relative to a known starting point, orientation and velocity. IMUs used in this kind of systems typically use sensors capable of obtaining measurements in a single direction and are mounted with their sensitive axes mutually perpendicular, respectively containing three orthogonal rate-gyroscopes and three orthogonal accelerometers.

In strapdown systems the inertial sensors are mounted rigidly onto the device, and therefore output quantities measured in the body frame rather than the global frame. To keep track of orientation, the signals from the rate-gyroscopes are integrated. To track position the three accelerometer signals are resolved into global coordinates using the known orientation, as determined by the integration of the gyroscopes signals. The global acceleration signals are then integrated as in the stable platform algorithm. This procedure is shown in figure 2.7 [35].

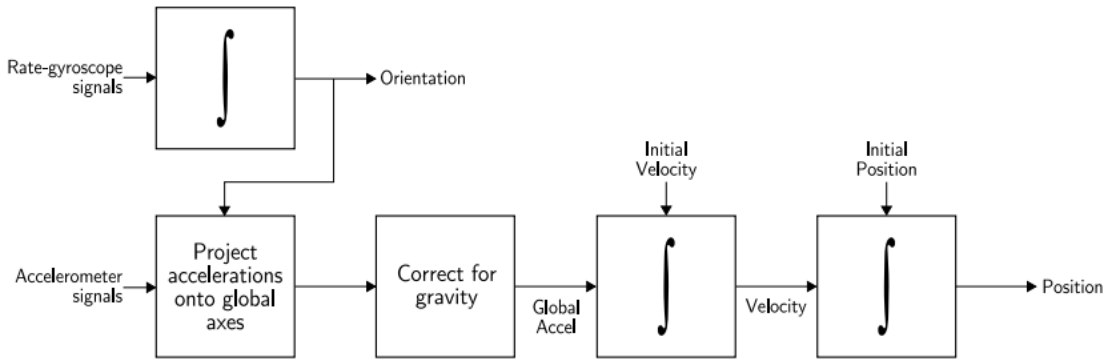

Fig. 2.7 Strapdown Inertial Navigation algorithm.

2.3.1.2 Orientation from Angular Rate

In an Inertial Navigation System (INS) the orientation, or attitude, is tracked by integrating the angular velocity signal ${}^{S}\omega = [\omega_x \quad \omega_y \quad \omega_z]$ obtained from the system's rate-gyroscopes. An IMU provides samples of the angular velocity at a fixed frequency rather than providing a continuous signal. An integration scheme is needed to integrate the sampled signal, for this

application which has a short timespan and low accuracy, the rectangular rule is presented as a sufficient solution.

Using the quaternion attitude representation, for a single period $[t-1,t]$ the solution to obtain the quaternion derivative that describes the orientation change rate of the sensor frame (S) relative to the Earth frame (E), ${}^{S}_{E}\dot{q}_{\omega,t}$, can be written as [36]:

$$ {}^{S}_{E}\dot{q}_{\omega,t} = \frac{1}{2} {}^{S}_{E}\tilde{q}_{t-1} \otimes {}^{S}p_{t} \tag{2.29} $$

where:

$$ {}^{S}p = [0 \quad \omega_x \quad \omega_y \quad \omega_z] \tag{2.30} $$

Let the period between successive angular velocity samples be δt. The orientation estimation ${}^{S}_{E}\tilde{q}_{\omega,t}$ is yielded by integrating the quaternion derivative.

$$ {}^{S}_{E}\tilde{q}_{t} = {}^{S}_{E}\tilde{q}_{t-1} + {}^{S}_{E}\dot{q}_{\omega,t} \cdot \delta t \tag{2.31} $$

Equation 2.31 represents the attitude update equation used to update ${}^{S}_{E}\tilde{q}_{t}$ as each new sample becomes available. This orientation estimate is always relative to the starting orientation. The block diagram for this algorithm is depicted in figure 2.8.

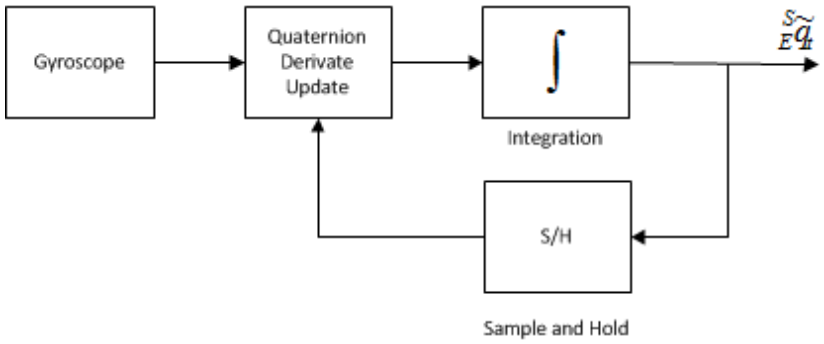

Fig. 2.8 INS orientation estimation block diagram.

Since the angular velocity signals obtained from the gyroscopes are integrated by the INS attitude algorithm, errors in the gyroscope signal will propagate through to the calculated orientation. Such errors lead to the accumulation of an additional drift in the integrated signal. This drift is observable while the device is turning, with its magnitude being proportional to the rate and duration of the motions. This issue leads to the development of a more robust solution to keep track of the appropriate orientation signal.

2.3.2 Complementary Filter

This sensor fusion algorithm combines the best attributes of each sensor type, and tries to mitigate the characteristics that introduce error in the estimates. Basically, it provides a robust trade-off between a good short term precision given by the gyroscopic integration and reliable long term accuracy provided by other sensors. Taking into consideration the use of a PI controller to provide the feedback of the angular error together with the quaternion representation of the data, this algorithm manages to have low computational needs without compromising efficiency.

2.3.2.1 Explicit Complementary Filter

This complementary filter fuses accelerometer and gyroscope data for orientation estimation such that low-pass filtering is applied on accelerometer data and high-pass filtering on gyroscope output [22]. On this filter we still want to compute the quaternion derivative, ${}^{S}_{E}\dot{q}_{\omega,t}$, in equation 2.29:

$$ {}^{S}_{E}\dot{q}_{\omega,t} = \frac{1}{2} {}^{S}_{E}\tilde{q}_{t-1} \otimes {}^{S}p_t \tag{2.29 revisited} $$

But this time considering ${}^{S}p_t$ as:

$$ {}^{S}p_t = [0, {}^{S}\Omega_t] \tag{2.32} $$

where, ${}^{S}\Omega_t$ represents the data fusion, here the gyroscope data is debiased by applying the feedback error. The first step for implementing the explicit complementary algorithm is to measure inertial direction ${}^{S}\hat{a}_t$ (by normalizing the accelerometer data). Then, estimate the direction of gravity from the quaternion output using

$$ {}^{S}\hat{g}_t = {}^{S}_{E}\tilde{q}^{*}_{t-1} \otimes {}^{E}\hat{g} \otimes {}^{S}_{E}\tilde{q}_{t-1} \tag{2.33} $$

This way we can estimate the angular error, being computed by cross multiplying the normalized accelerometer data and estimated direction of gravity as

$$ {}^{S}\omega_{e,t} = {}^{S}\hat{a}_t \times {}^{S}\hat{g}_t \tag{2.34} $$

From here we apply the data fusion in equation 2.35 through the use of a PI controller (this way we account not only, for the error at each time t, but also consider the history of the error).

$$^{S}\Omega_t = {}^{S}\omega_t + K_p \cdot {}^{S}\omega_{e,t} + K_i \cdot \int {}^{S}\omega_{e,t} \tag{2.35}$$

for suitable values of K_P and K_i [37]. To compute the rate of change of quaternion the same principal from the INS orientation tracking algorithm is used. Following the same development, for a single period $[t-1,t]$, we can obtain ${}^{S}_{E}\dot{q}_{\omega,t}$ in equation 2.29. The quaternion output is once again obtained by integrating the quaternion derivative:

$${}^{S}_{E}\tilde{q}_t = {}^{S}_{E}\tilde{q}_{t-1} + {}^{S}_{E}\dot{q}_{\omega,t} \cdot \delta t \tag{2.31 revisited}$$

Finally, for the quaternion to represent rotation correctly it needs to be normalized before the next iteration in the algorithm:

$${}^{S}_{E}\hat{q}_t = \frac{{}^{S}_{E}\tilde{q}_t}{||{}^{S}_{E}\tilde{q}_t||} \tag{2.36}$$

The block diagram for this algorithm is depicted in figure 2.9.

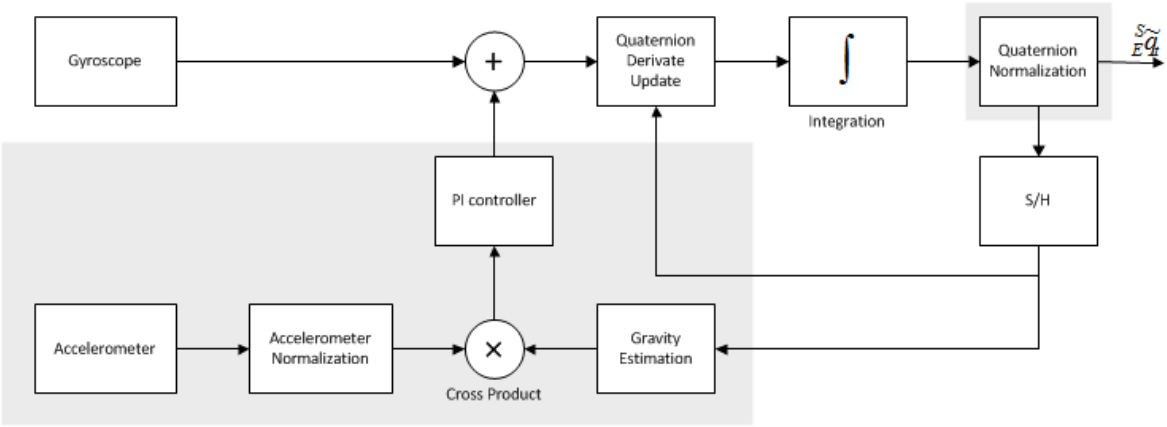

Fig. 2.9 Explicit complementary filter block diagram.

2.3.2.2 Explicit Complementary Filter with magnetometer

This new implementation adds the data of the magnetometer, meaning that the complementary filter will combine the low frequency signals from accelerometer and magnetometer with the high frequency signals from the gyroscope [38].

The angular error, ${}^{S}\omega_{e,t}$, is relative to the difference between the measured orientation and the predicted one. It is obtained with the cross product between the orientation given by magnetometer measures,${}^{S}\hat{m}_t$ and the predicted one, ${}^{S}\hat{b}_t$, plus the gravitational field direction measured by the accelerometer, ${}^{S}\hat{b}_t$, and the predicted one, ${}^{S}\hat{g}_t$.

$$^{S}\omega_{e,t} = {}^{S}\hat{a}_t \times {}^{S}\hat{g}_t + {}^{S}\hat{m}_t \times {}^{S}\hat{b}_t \tag{2.37}$$

$$^{S}\hat{g}_t = {}^{S}_{E}\tilde{q}^{*}_{t-1} \otimes {}^{E}\hat{g} \otimes {}^{S}_{E}\tilde{q}_{t-1} \tag{2.33 revisited}$$

Obtaining the predicted reference direction of the earth's magnetic field is achieved by computing $^{E}\hat{b}_t$ as $^{E}\hat{h}_t$ (measured direction of the earth's magnetic field) normalised to have only components in the earth frame x and z axes. Compensating for magnetic distortions in this way ensures that magnetic disturbances are limited to only affect the estimated heading component of orientation [39].

$$^{E}\hat{h}_t = \begin{bmatrix} 0 & h_x & h_y & h_z \end{bmatrix} = {}^{S}_{E}\tilde{q}_{t-1} \otimes {}^{S}\hat{m}_t \otimes {}^{S}_{E}\tilde{q}^{*}_{t-1} \tag{2.38}$$

$$^{E}\hat{b}_t = \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & 0 & h_z \end{bmatrix} \tag{2.39}$$

$$^{S}\hat{b}_t = {}^{S}_{E}\tilde{q}^{*}_{t-1} \otimes {}^{E}\hat{b} \otimes {}^{S}_{E}\tilde{q}_{t-1} \tag{2.40}$$

The cinematic equation for the orientation of the sensor, $^{S}\Omega_t$, is obtained through a PI gain of the angular error. The gains K_p and K_i correspond to the proportional and integral gains, respectively. The proportional gain controls the frequency value that delimits the importance of the gyroscope sensor information versus the accelerometer and the magnetometer sensors data. The integral gain is adjusted to correct the gyroscope offset drift.

$$^{S}\Omega_t = {}^{S}\omega_t + K_p \cdot {}^{S}\omega_{e,t} + K_i \cdot \int {}^{S}\omega_{e,t} \tag{2.41}$$

To obtain the estimated orientation, it is necessary to compute the rate of change of the quaternion as follows:

$$^{S}p_t = [0, {}^{S}\Omega_t] \tag{2.32 revisited}$$

$${}^{S}_{E}\dot{q}_{\omega,t} = \frac{1}{2} {}^{S}_{E}\tilde{q}_{t-1} \otimes {}^{S}p_t \tag{2.29 revisited}$$

To maintain a proper estimation, the quaternion is integrated (equation 2.31) and normalized (equation 2.36) in the end of each iteration.

The implementation of this filter can be easily understood on the following high level block diagram:

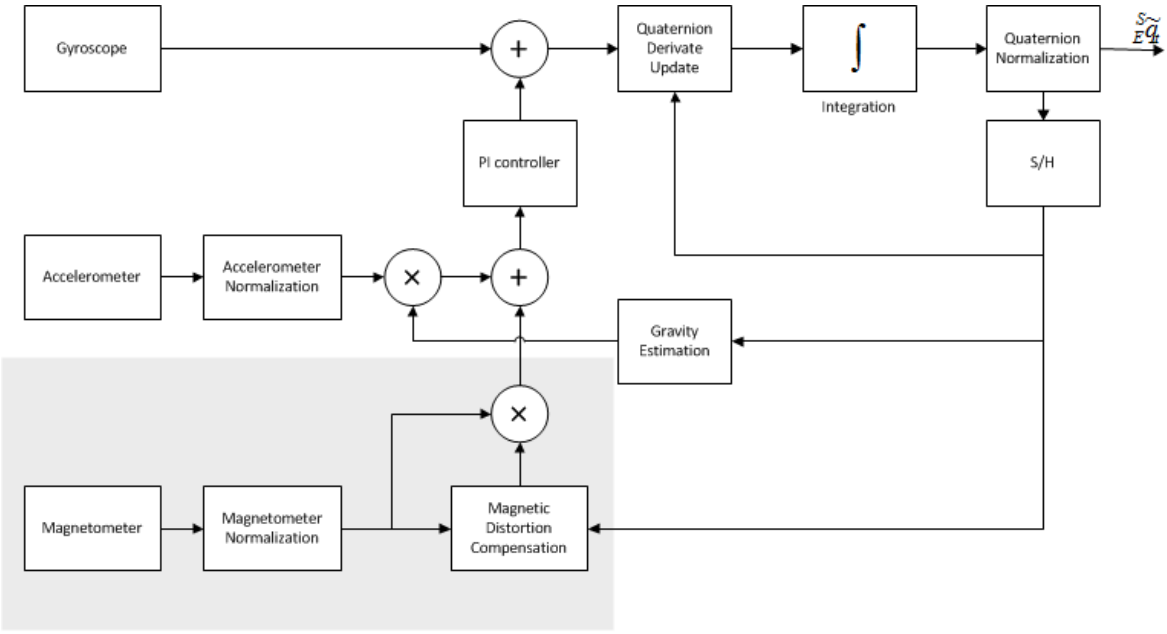

Fig. 2.10 Complementary filter with magnetometer block diagram.

Chapter 3

Filter Results and Discussion

3.1 Orientation Tracking Experiments

In order to access the implemented algorithms on inexpensive sensors, a motion device from Invensense[1] was used for testing, the 9-axis MPU-9250. This sensor combines a 3-axis gyroscope, a 3-axis accelerometer and a 3-axis magnetometer on the same package. Experiments were performed on the MPU-9250 breakout board, giving us easy access to use the integrated circuit and run simulations on Arduino[2] and Matlab platforms. Although the prototype contains MPU-9150 sensors, due to the immediate availability and similar performance, for testing purposes the MPU-9250 also contains an onboard Digital Motion Processor (DMP) capable of processing complex motion fusion algorithms. The DMP acquires data from the sensors, processes the data and outputs directly calculated orientation information. We use this output as our first approach for comparison throughout the experiments. The downside of using the DMP is that the manufacturer did not provide much information on its proprietary inner workings.

Although the DMP provides a fairly reliable source to approve the algorithms capabilities, the MTi sensor from XSens is used as ground-truth for this work. The Xsens has become an industry standard in the production of MEMS AHRSs, their products are often used as replacements for high-grade IMUs, providing high quality components with proven and robust filter design. They use a proprietary Kalman filtering solution that copes with transient accelerations, magnetic disturbances and even vibrations.

[1] Invensense Motion Technologies details present at: www.invensense.com/technology/motion/

[2] Further details www.arduino.cc

3.1.1 Strapdown Inertial Navigation

Tests were performed using an Arduino board, raw data from the sensor and both the INS algorithm and DMP outputs were recorded and processed at 50 Hz for comparison. No specific experimental procedure was used for testing, the tests included rotations along the threes sensor axes with multiple accelerations and durations. As supplementary work, we also present an output of an implementation of the algorithm in Matlab, using the obtain raw data values. An example of such experiments is shown in figure 3.1, which demonstrate a graphical comparison between the orientation estimates, using quaternion representation. The four graphics represent the four components that compose a quaternion. The results were as expected, the INS algorithm follows the same estimation as the DMP with minimal differences during small periods of time. Both the implementations on Arduino and Matlab give the exact same outputs.

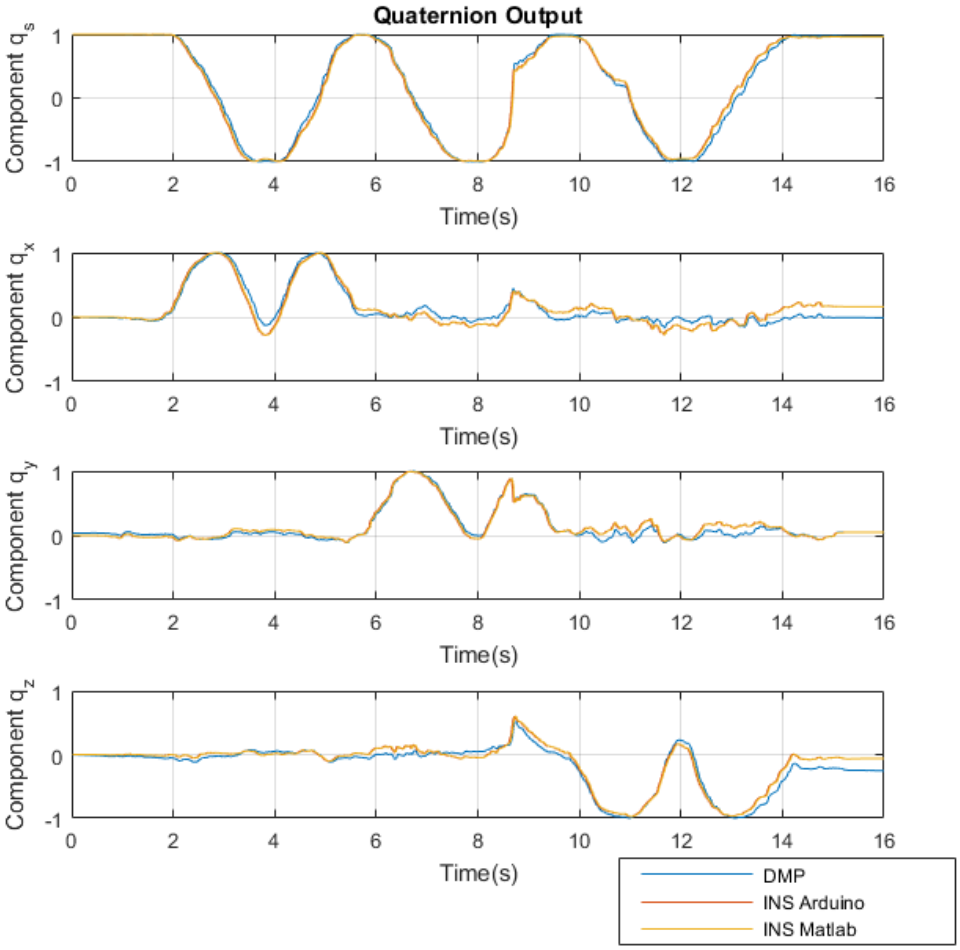

Fig. 3.1 Standard INS algorithm vs. DMP output.

As it was already mentioned in chapter 2.3.1.1, the angular velocity signals obtained from the gyroscopes are integrated, therefore errors in the gyroscope signals propagate through to the calculated orientation causing a drift over time on the calculated quaternion. To confirm this error in orientation, a test was performed where the sensor was left in a stationary position while using the algorithm during a longer period of time (15 minutes). The experiment was made with two independent sensors, in the beginning some rotations were applied and then the sensors were left at rest, the resulting drift can be seen in figure 3.2 (each column of graphics represent the four components of the estimated quaternion, respectively per sensor).

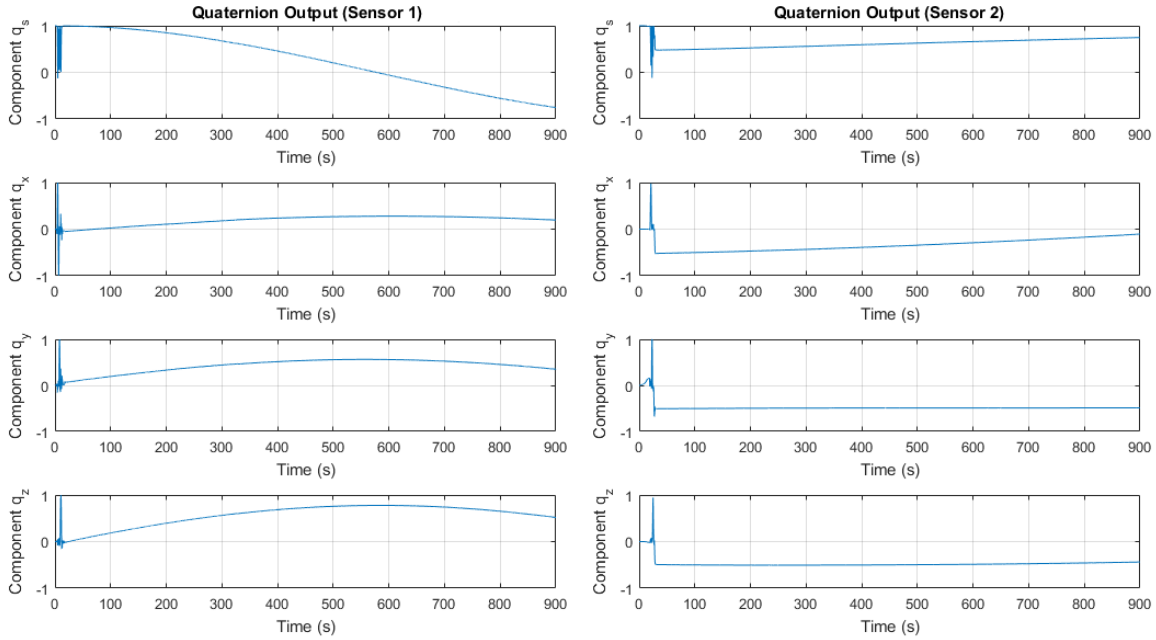

Fig. 3.2 INS algorithm stationary drift (quaternion representation).

As predicted the drift occurs and its magnitude varies differently between sensors. To have a better perception of the amount of drift the quaternion representation was converted to Euler angles (figure 3.3).

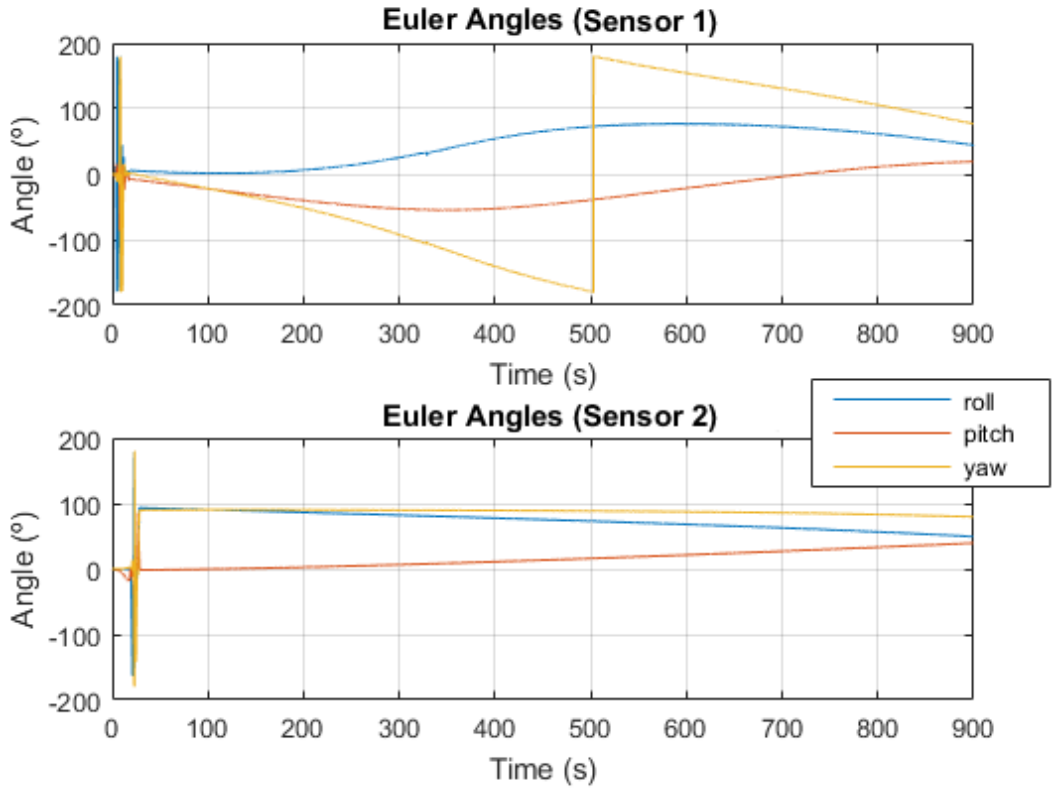

Fig. 3.3 INS algorithm stationary drift (Euler angles representation).

If we compare the initial and final orientation, on sensor 1 (which represents the worst case scenario) we can observe an error up to 247° on the yaw angle, whereas on sensor 2 we observe a maximum difference of 43° on the pitch angle.

These results make the standard INS algorithm unsuitable for our application, only giving a precise orientation tracking during a very small period of time.

3.1.2 Complementary Filter without magnetometer

Following the same experimental procedure as in with the INS algorithm, raw data was obtain to simulate on Matlab, as well as the quaternions from both the DMP and the complementary filter directly implemented on Arduino. Again as expected, the complementary filter algorithm follows the same estimation with minimal differences for short periods of time. The following figure 3.4 presents these results.

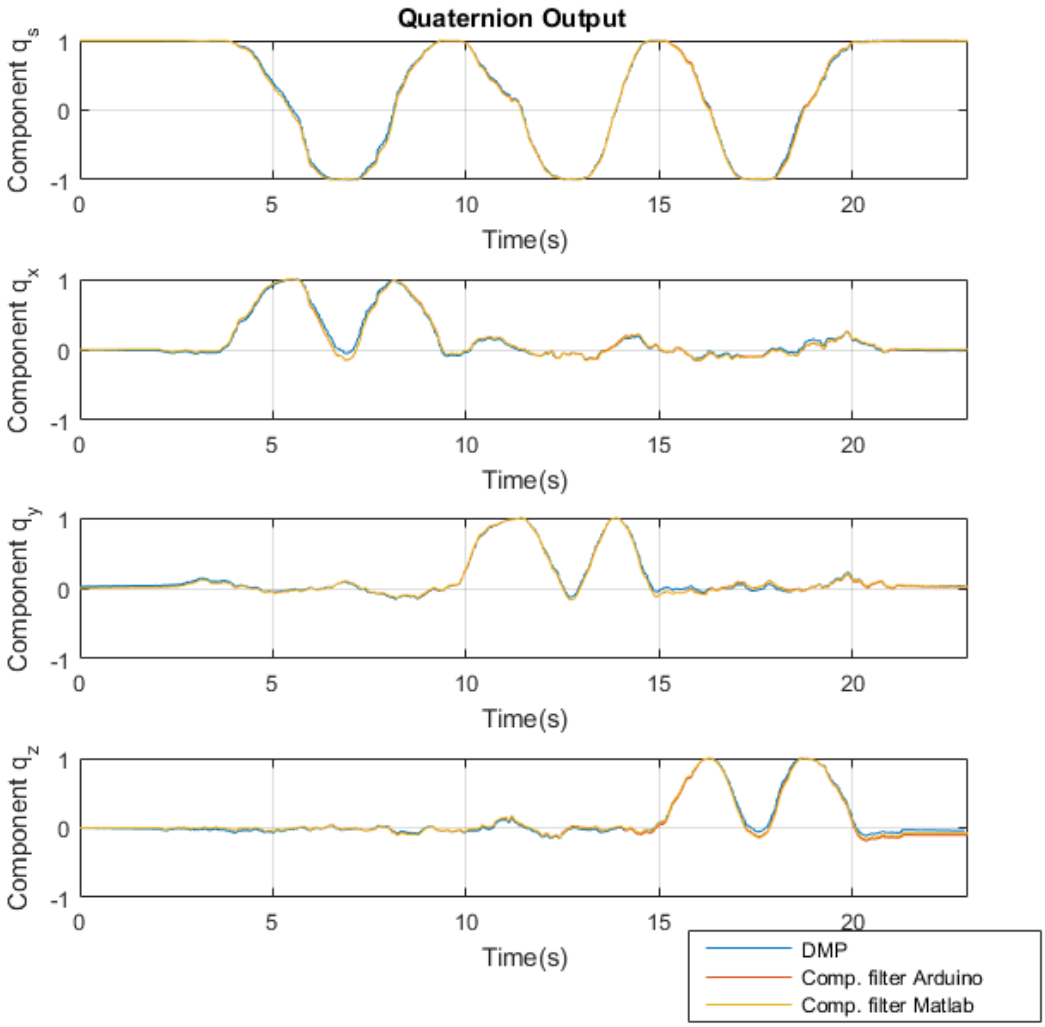

Fig. 3.4 Complementary filter algorithm vs. DMP output.

As mentioned previously, this filter beside using the gyroscope axis output as velocity reference it also uses the angle between the accelerometer output and the body-fixed-frame as attitude reference to obtain a zero bias estimate of the gravitational direction. This results in a much better estimation, the calculated quaternion should have a much smaller drift over time.

Repeating the same experiment as before, leaving the sensors at rest during 15 minutes, we can observe in figure 3.5 and figure 3.6 that the drift is almost gone, or at least it takes much more time to be significant enough to disrupt the intended output. Comparing the initial and final orientation, on sensor 1 we can observe a maximum final drift of only 6° on the yaw angle, whereas on sensor 2 we observe a maximum error of 26° on the pitch angle.

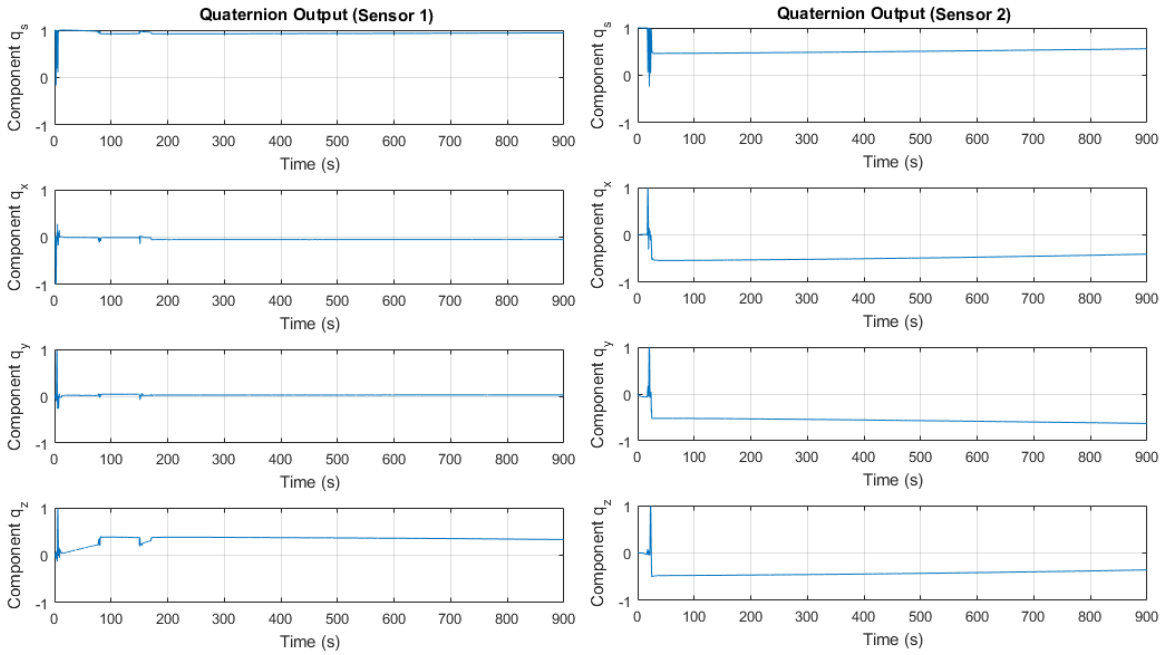

Fig. 3.5 Complementary filter stationary drift (quaternion representation).

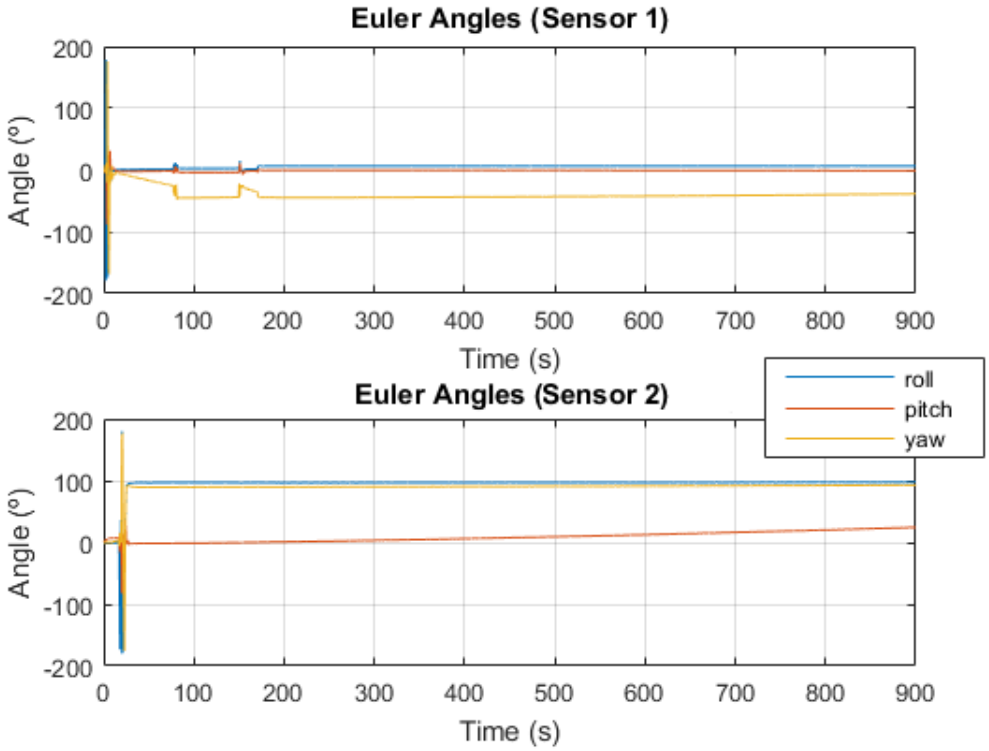

Fig. 3.6 Complementary filter stationary drift (Euler angles representation).

3.1.3 Complementary Filter with magnetometer

Without the heading provided by the magnetometer, with the addition of motion the previous algorithm still has a drift over time on the initial orientation. This drift can be identified on the z and scalar components of the quaternion (or the yaw angle of the Euler angles) since the gravity correction only takes into account drifts that occur on the XY plane.

Some long-term experiments were made in order to access if this is true and that the new implementation corrects this drift. Figure 3.7 and figure 3.8 shows the results of one of these experiments, a ten minutes experiment maintaining the sensor at rest and only moving it after a couple of minutes to really force the orientation to drift. Both implementations were tested and the obtain results were overlapped on the same graphics to have a better perception of the differences.

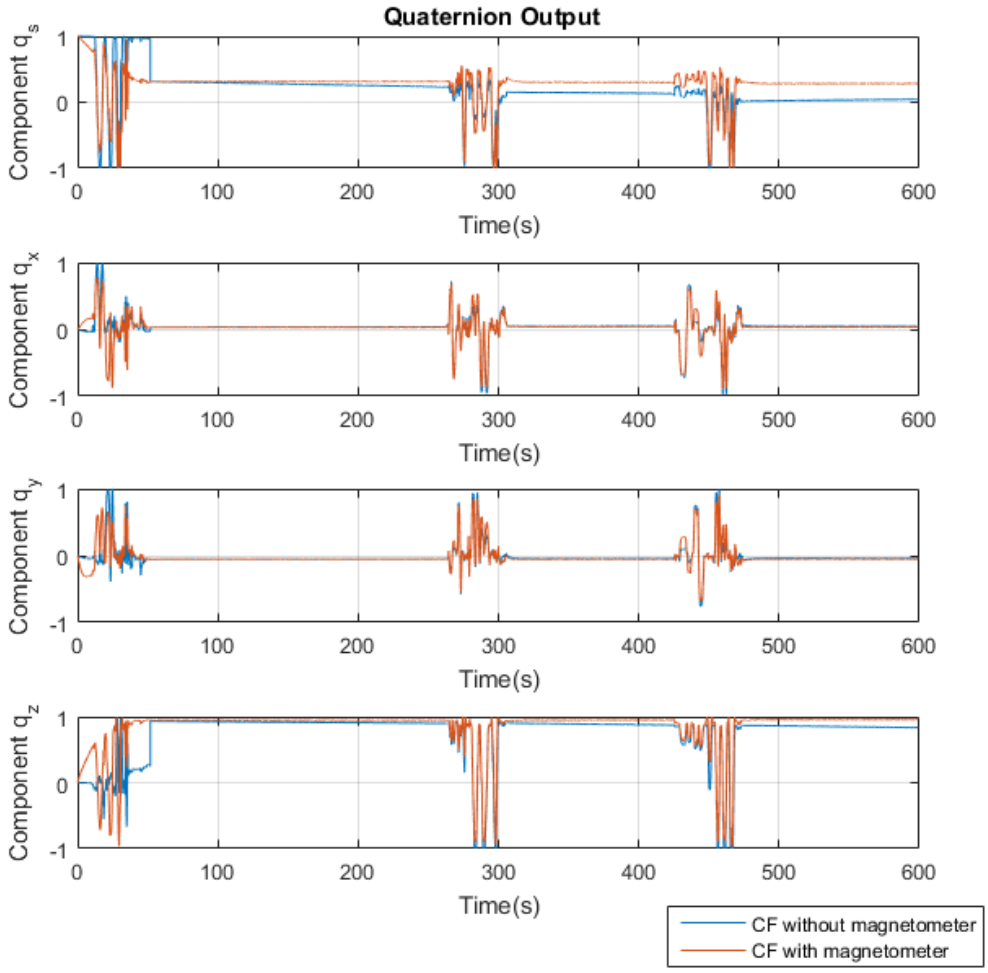

Fig. 3.7 Complementary Filter with and without magnetometer data (quaternion representation).

The results are as expected, the previous algorithm has a small drift over time, it can be seen clearly on the scalar component of the quaternion and slightly on the z component of the quaternion. Keeping the sensor with the same orientation throughout the experiment, with the new implementation the obtained quaternion is the approximately the same, both at the start and ending of the experiment. Without the magnetometer, the orientation has a drift of 30° on the yaw angle, while, using the magnetometer, a 3° drift is observed and only 0.0066° and 0.5022° drifts for the roll and pitch angles, respectfully.

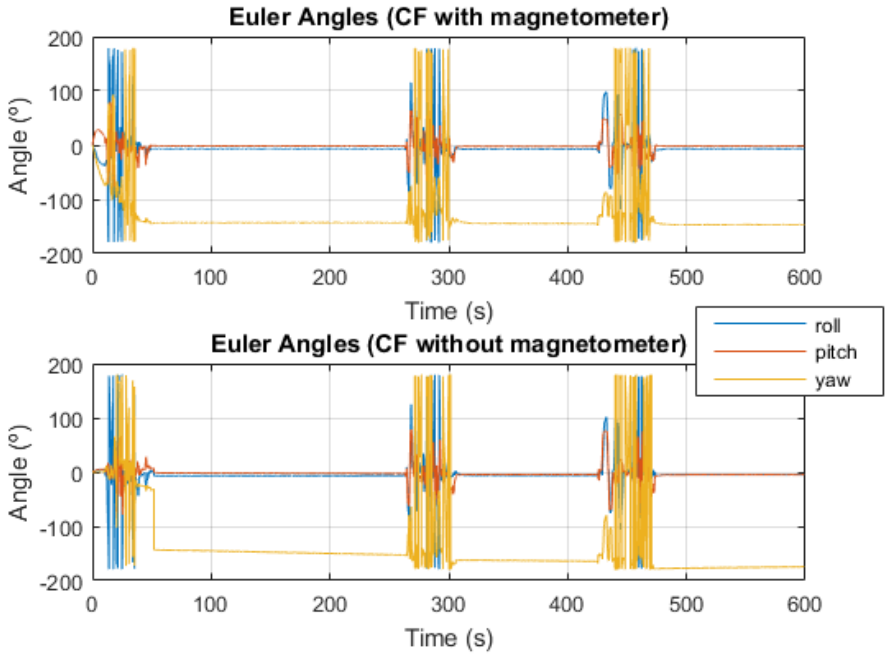

Fig. 3.8 Complementary Filter with and without magnetometer data (Euler angles representation).

3.1.4 Evaluation of the Estimator Quality

The final algorithm was tested using the Xsens MTi sensor. Raw sensor data was logged to a PC at 100 Hz and imported through Xsens software to provide calibrated sensor measurements which were then processed by the proposed orientation estimation algorithm. As both the Kalman-based algorithm and Complementary filter estimates of orientation were computed using identical sensor data, the performance of each algorithm could be evaluated relative to one-another.

Various experiments were conducted and both algorithms compared, figure 3.9 shows a variable scenario with rotations in different velocities on all three axes and a period with the sensor stationary.

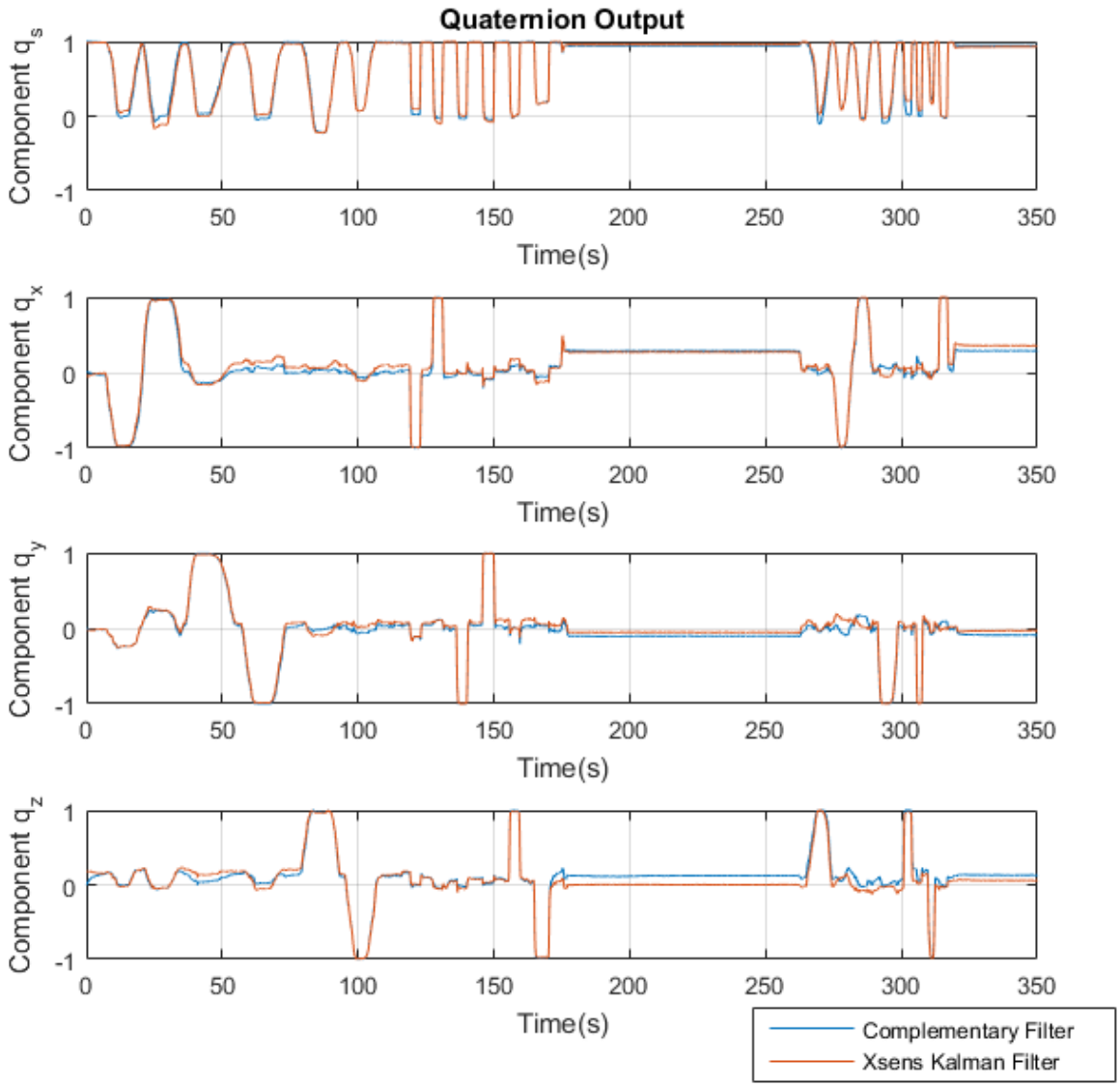

Fig. 3.9 Complementary Filter vs. Xsens Kalman Filter.

It is common [23][40][41] to quantify orientation sensor performance using the Root-Mean-Square Error (RMSE) which measures the differences between values predicted by an estimator and the values actually observed. In order to obtain the best performance possible, the Complementary filter was evaluated for different values of the tunable parameters K_p and K_i. As described in chapter 2.3.2.2 the gain K_i is associated with the bias estimation process of the complementary filter. For such short datasets the slow dynamics associated with the bias estimate do not influence greatly the estimated outputs. Consequently, in the experimental studies undertaken the bias gain is set to a very low value, $K_i = 0.02$. This way by tuning K_p we were able to obtain RMSE values of 0.0586 (in quaternion units) and

0.9585° (Euler angles). Figure 3.10 evaluates this performance of the Complementary filter by showing the RMSE as function of K_p, for constant $K_i = 0.02$.

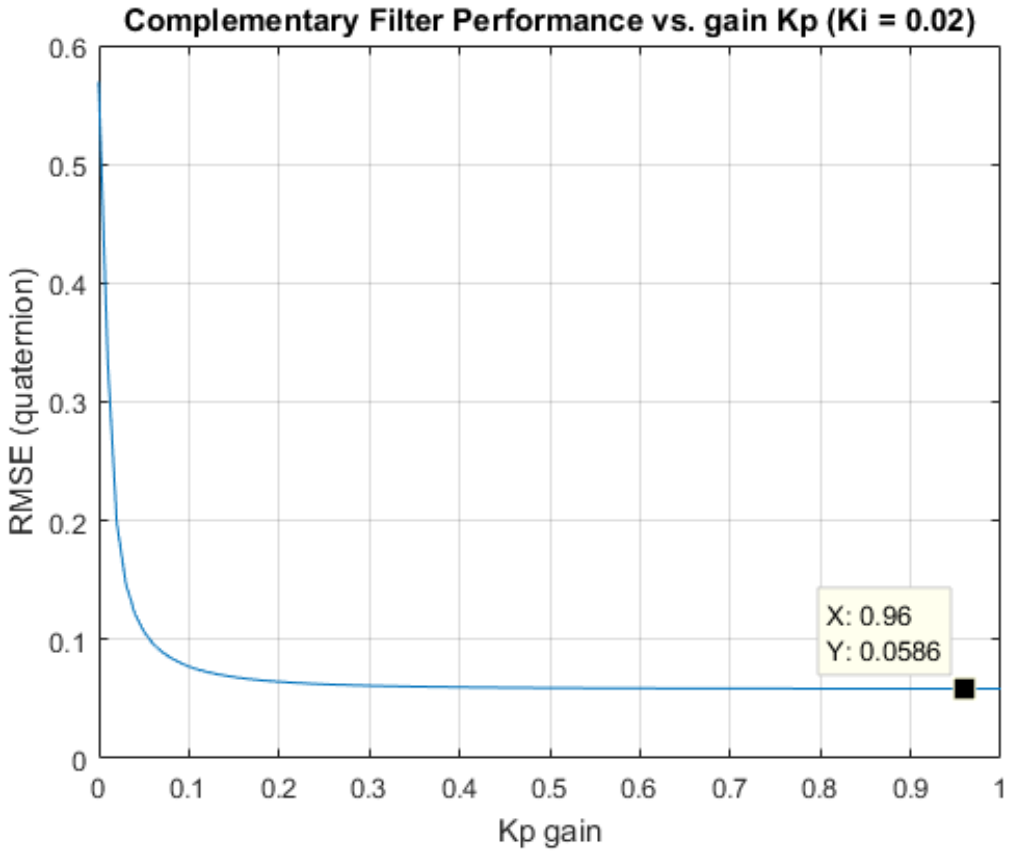

Fig. 3.10 Performance of Complementary Filter as function of K_p.

Chapter 4

Development of a Prototype for the Hand Tracking System

In this chapter we present the prototype produced as a result of this dissertation. We start by giving an overview of the human hand model and the corresponding kinematic approach, as well as the considerations and simplifications implemented in order to provide a simpler analytical solution. The second section describes the hardware implementation detailing the modules and devices used. Finally, the third section show the implemented animation developed in the OpenAR platform.

4.1 Human Hand Kinematics

Kinematics is the branch of classical mechanics which studies the motion of bodies without considering neither the masses nor the forces which cause the motion. In the context of hand modelling, it defines the set of possible motions a hand can do. A kinematic model of a hand is comprised of links that imitate the human bones and joints, which determine the constrains of motion between the links. Taking a look at the human hand bone structure as presented by Gustus *et al.*[42] in figure 4.1 we are able to realize that an accurate modelling may prove difficult due to the complexity of the human hand.

The bones of a human hand consist of the carpal bones, the metacarpal (MC) bones, the proximal phalanges (PP), medial phalanges (MP) and distal phalanges (DP). They are connected by the carpometacarpal (CMC) joints, the intermetacarpal (IMC) joints, the metacarpophalangeal (MCP) joints, the proximal interphalangeal (PIP) joints, the distal interphalanges (DIP) joints and the interphalangeal (IP) joint of the thumb. The fingers are numbered as follows: 1 thumb, 2 index, 3 middle, 4 ring, 5 little.

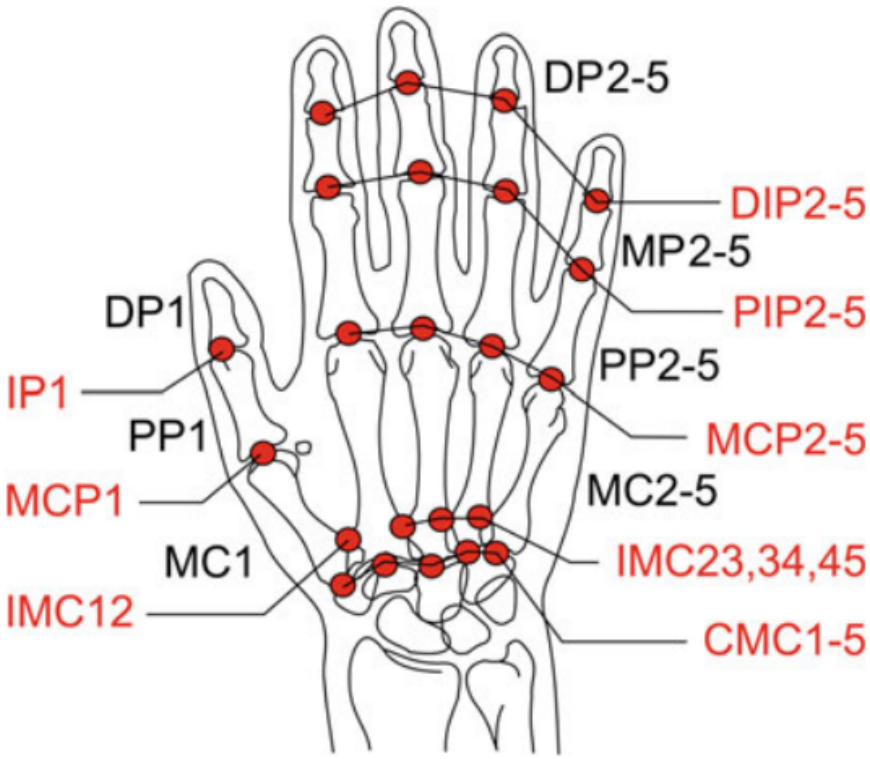

Fig. 4.1 Human hand bone structure and joints layout.

In this work we arrived at a kinematic model where many simplifications are made while preserving the kinematic information for the hand manipulation. This way we provide a suitable solution to ease implementation and improve computational speed for applications in real time. In our implementation we consider three joints for each finger and one joint for the wrist. Comparing with the bone structure in figure 4.1, we maintain the DIP2-5, PIP2-5, MCP2-5, IP1, MCP1 joints while defining one joint for the base of the thumb and one for the wrist, this way we simplify the remaining joints without decreasing the range of motion. Our hand model with the implemented virtual bone structure is depicted in figure 4.2.

All the joints in the hand kinematic model are defined by pure rotations, meaning that they have a maximum of 3 DoF, each describing a rotation around an axis of the attached frame. We propose a hand model with 24 DoF: one DoF (extension/flexion) for DIP2-5, PIP2-5, IP1 and MCP1, two DoF (extension/flexion and adduction/abduction) for MCP2-5 joints and three DoF on the joint on the base of the thumb (allowing the thumb to also rotate longitudinally) and the wrist joint (simulating the twist of the forearm).

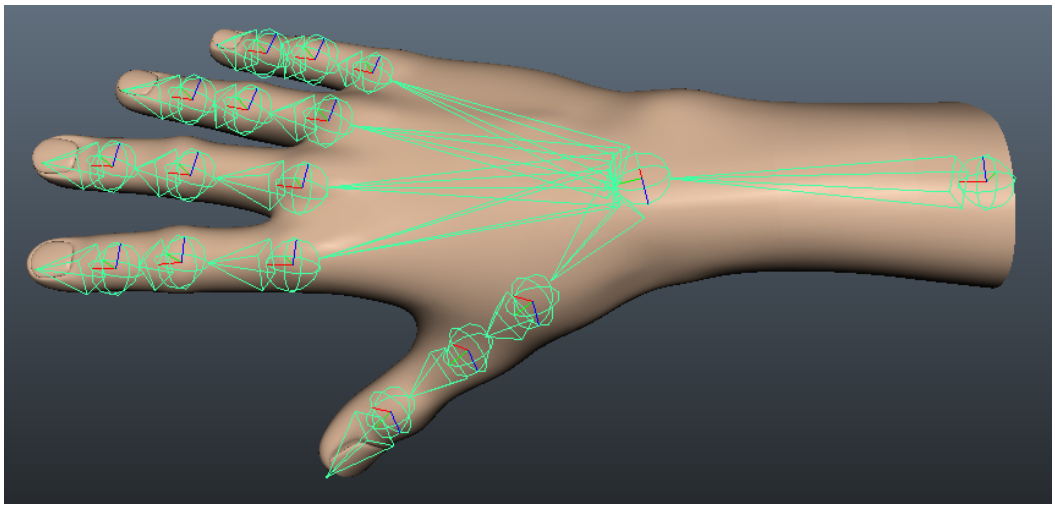

Fig. 4.2 Hand model with virtual bone structure.

As with any model, it is important to understand its limitations to assure that the desired behaviours are produced and that the assumptions of the model are not violated. This way many restrictions supplement the kinematic chain in order to define correct finger configuration according to the model structure and the space of possible postures. Direct kinematic equations are used to provide the position and orientation of each finger tip. Equation 4.1 show the direct kinematics of each finger.

$$E_i = {}^{wrist}_{base}T \cdot {}^{wrist}_{tip}T_i \tag{4.1}$$

- E_i represents a matrix that contains position and orientation of the finger tip ($i = 1,2,3,4,5$; fingers numbered as in figure 4.2);
- ${}^{wrist}_{base}T$ represents the distance between wrist and the base reference frame that connects the hand to the forearm;
- ${}^{wrist}_{tip}T_i$ is a matrix that contains the geometrical transformation between the i-finger reference frame and its corresponding finger tip. This matrix is composed by the concatenation of more simple matrices that represents the contribution of each finger joint.

By expanding equation 4.1 for the thumb we obtain equation 4.2. Anatomical terminology is considered for the joints except for the two simplified joints (*wrist* and *thumb*).

$$E_1 = {}^{wrist}_{base}T \cdot {}^{wrist}_{thumb}T \cdot {}^{thumb}_{MCP1}T \cdot {}^{MCP1}_{IP1}T \tag{4.2}$$

And for each of the remaining finger, where $j = 2,3,4,5$.

$$E_j = {}^{wrist}_{base}T \cdot {}^{wrist}_{MCP_j}T \cdot {}^{MCP_j}_{PIP_j}T \cdot {}^{PIP_j}_{DIP_j}T \tag{4.3}$$

With the proposed hand model, to obtain the orientation of each joint, in terms of hardware, the ideal solution would be to place an IMU sensor in each joint and read the corresponding orientation as the user moves his hands around. This would result in a solution with 16 IMUs, making the final prototype relatively complex and probably not very comfortable to use. Therefore, in order to decrease the amount of sensors placed on the hand, a study was performed [43][44][45] on the most common hand gestures and motions that a user does (for example: grasping motions, okay sign or thumbs up, pointing a finger to give directions and many others) and the impact these movements have on the finger articulation. In this way we determined that, by assuming a constant relation between the joints of the fingers, we are able to use only one sensor for each finger without losing enough gestures and motions that would disrupt the immersion of a user. In this manner, we propose another simplification to the kinematic model, where the orientation of the three joints of each finger are computed in respect to only one joint. This results in a solution with only 6 IMUs, allowing the production of an ergonomic hardware solution that does not disrupt the user and provides the similar performance with subtle limitations.

4.1.1 MATLAB Based Animation

In order to validate the full kinematic model of the hand present in section 4.1 an experiment was performed simulating a hand manipulator in Matlab environment. The toolbox presented by Pajak in [46] was used as it provides the necessary tools and functions to simulate a realistic model of a human upper limb. This toolbox allows to define any manipulator described by Denavit-Hartenberg parameters and connect the mechanisms created into one more complex mechanism, making it possible to simulate any open kinematic chain. Furthermore, it contains default 3D representations for these mechanisms to be shown in a realistic manner, as well as providing a set of basic blocks and functions to manipulate position and orientation of those objects in the 3D space.

The toolbox was also modified to allow a real-time representation of the hand, while reading the data from the sensors. Although this tweak granted an additional perception of the tracking, it only allowed slow frame rates (between 10 to 15 FPS), seeing that the toolbox was never intended to have this feature and was not optimized for it.

An example of the hand manipulator used from the toolbox is depicted in figure 4.3.

The purpose of this simulation was to assess how the sensor placement affected the behaviour of the hand, particularly the finger behaviour while only considering one sensor

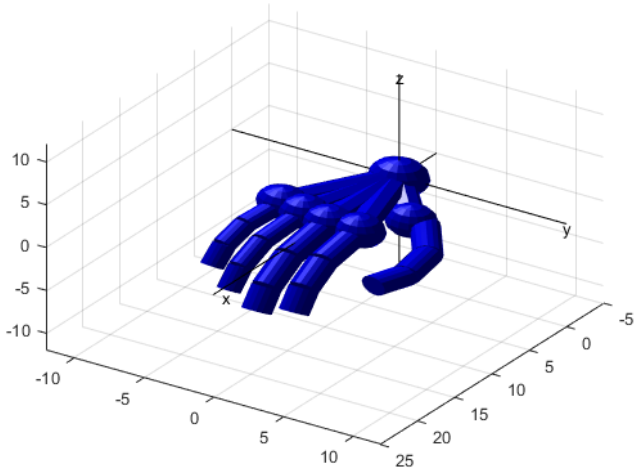

Fig. 4.3 Hand in default 3D view of the toolbox.

for the three joints. The Denavit-Hartenberg convention uses four parameters for attaching reference frames to the links of the spatial kinematic chain, it assumes revolute joints and allows to control the joint angle while the other parameters describe the constrains between the link coordinate frames.

Taking into consideration the common active range of motion of the joints of the human hand, as documented in [47], on our solution we calculate the kinematic chain with the finger sensor on the first joint of the finger, the MCP joint, and assume a proportional relation for the next two joints according to their range of motion (for example, for the finger joints excluding the thumb, the MCP joint as a range of $[0, 100°]$, the PIP joint $[0, 105°]$ and the DIP joint $[0, 85°]$). With this approach, all expected hand configurations were faithfully represented, although we lose independent joint motion, making less common configurations like the ones in figure 4.4 impossible to perform.

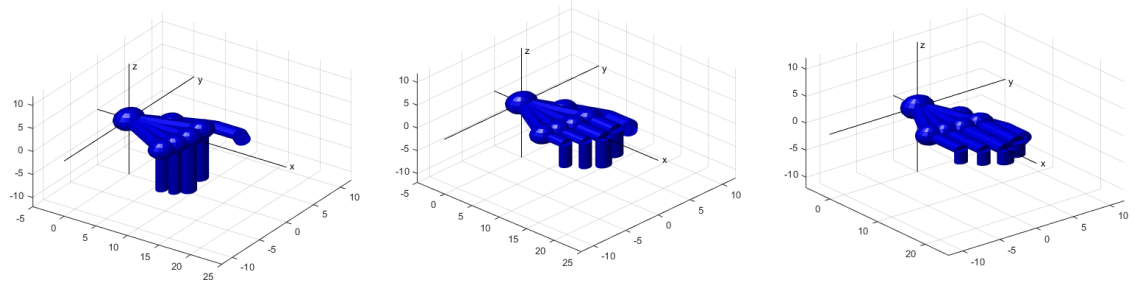

Fig. 4.4 Examples of impossible to perform hand configurations.

Nevertheless, the prototype is not meant to be use for gesture recognition and most of the gestures that result from independently moving these joints are not commonly used in manipulation of objects or in interactions with an interface.

4.2 Hardware Implementation

4.2.1 Proof-of-Concept Prototypes

Before the development of the full prototype, to demonstrate the feasibility of this solution, two simpler prototypes were developed. These first designs were never intended to be early versions of the prototype, but merely proof-of-concept prototypes developed at an early stage of the implementation. The first solution contained one sensor board and an ESP-01 on a wrist band and the second one was extended containing an additional sensor board on one finger, as is demonstrated on figure 4.5.

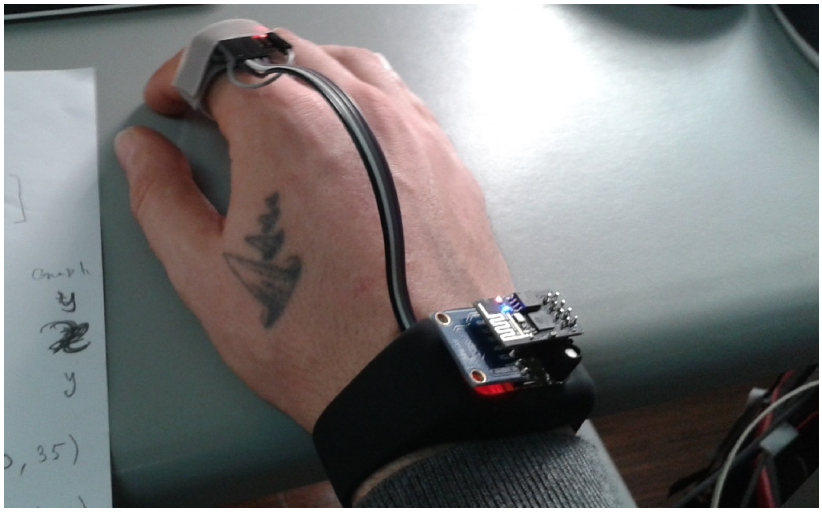

Fig. 4.5 Proof-of-concept Prototype.

In experiments using these prototypes the orientation data from the sensor on the wrist was applied directly to the wrist joint and the data from the other sensor applied to all the MCP joints of the fingers. These experiments only provided very limited finger motion and allowed motions that did not respect the natural hand kinematic, however they proved the capabilities of this approach with very precise tracking of the hand during small periods of time (only using gyroscopes and accelerometers), giving the necessary enticement to further extend and improve the technology.

4.2.2 Hand-Tracker Prototype

To make the Hand-Tracker prototype as comfortable-to-use as possible two types of boards were developed. The full design consists of six PCBs, a primary board containing the main components of the prototype and placed on the back of the hand and five identical boards containing one IMU sensor each, positioned on each respective finger. The five boards are connected to the primary board using FFC cables. These boards were designed to occupy minimal space and fit easily on the dorsal part of the hand. Figure 4.6 shows the complete Hand-Tracker prototype.

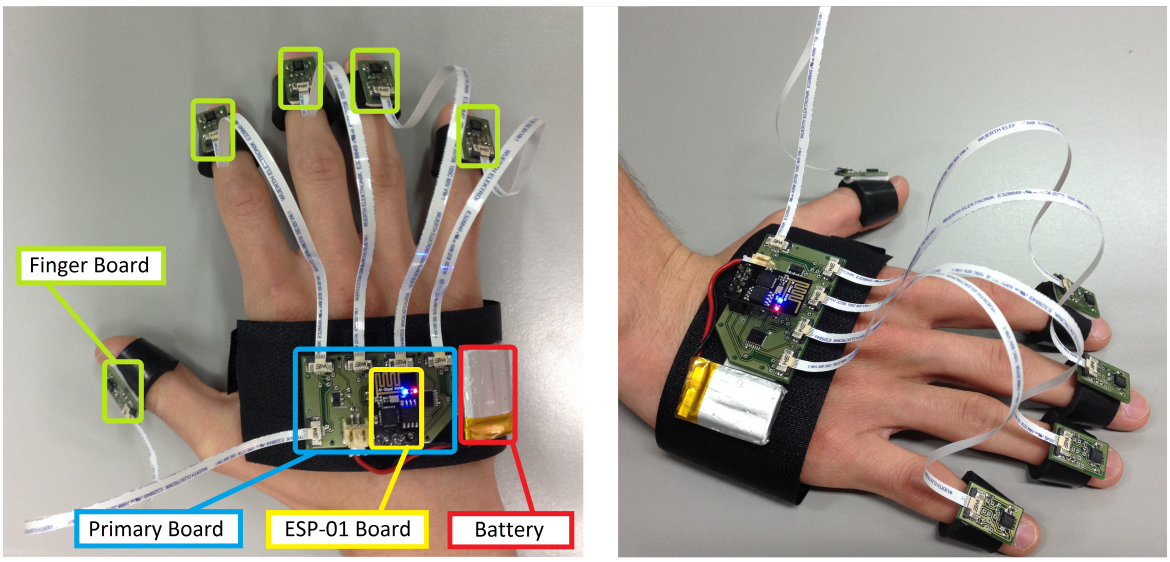

Fig. 4.6 Hand-Tracker prototype.

All these boards were designed and the components assembled at the laboratory with the only exception of the ESP-01 board, depicted in figure 4.7, which is already designed to occupy minimal PBC area and no further miniaturization was needed.

The primary board works as the core unit of the the prototype containing the processing and communication modules, as well as the power input. It incorporate a power regulator, one MPU-9150 sensor, a shift register and an ESP8266 board.

To power the Hand-Tracker a small LiPo battery is used, with a nominal voltage of $3.7V$ regulated down to $3.3V$. The power regulator is a XC6206 positive voltage regulator with low power consumption that provides more than enough output current for our implementation (up to $200mA$) and the appropriate output voltage ($3.3V$).

The board also contains one IMU, the MPU-9150 sensor, which provides the data corresponding to the wrist joint of the kinematic model. Even tough an MPU-9250 breakout board was used for testing the algorithms, the MPU-9150 sensor was already available at the

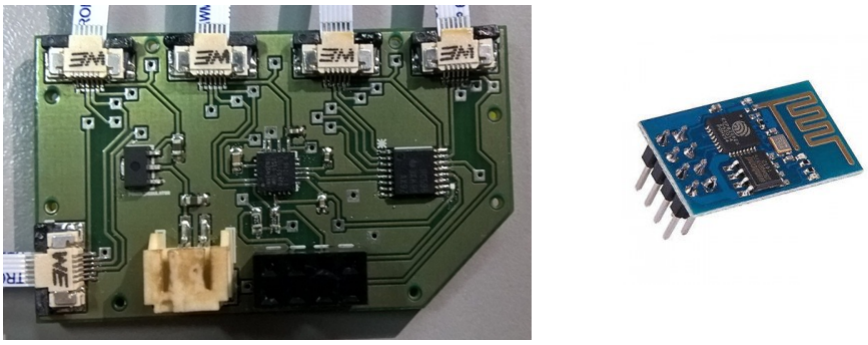

Fig. 4.7 Prototype primary board (left) and ESP8266 ESP-01 board example (right)

laboratory in the appropriate SMT case, ready to be mounted on the PCB. Since these two sensors have very similar characteristics and identical performances, there was no need to wait for the acquisition of new MPU-9250 sensors to assemble the boards.

To be able to read multiple sensors we use the SN74HC595 8-Bit Shift Register. We use this device due to a limitation in the I^2C communication interface of the MPU-9150 sensor. As described in [48], in a generalized I^2C interface implementation, attached devices can be a master or a slave. The master device puts the slave address on the bus, and the slave device with the matching address acknowledges the master. In the case of the MPU-9150, it always operates as a slave device when communicating to the system processor, which thus acts as the master. The slave address of the MPU-9150 is 7 bits long with the LSB bit of the address determined by the logic level on pin AD0. This only allows two MPU-9150s to be connected to the same I^2C bus. For that reason we use a shift register which is a device that allows to shift by one position the bit array stored in it. Basically we only read from one slave address and keep all the remaining sensors with the other address, in other words, every time we want to read a sensor, its AD0 pin has one logic level whereas all remaining AD0 pins from other sensors have the opposite logic level. Using this strategy the shift register allows to read each sensor successively.

To compute all the information provided by the sensors and transmit the data to a computer a Wi-Fi module, the ESP8266 ESP-01, is used and connected to a stackable header on the PCB. This module offers a self-contained Wi-Fi networking solution and capabilities such as on-board processing and storage. Therefore, this module is used to store all the calibration routines and sensor fusion algorithms. The module creates a wireless access point and outputs the final orientation data of each sensor directly to the user's computer.

The smaller boards placed on the fingers simply contain an MPU-9150 and its respective operating circuit, depicted in figure 4.8.

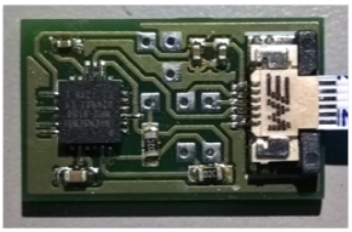

Fig. 4.8 Prototype finger board.

With these board designs the final Hand-Tracker prototype is equipped with 6 IMU sensors, it has an operating voltage of $3.3V$ and an operating current of $100mA$, resulting in 3 hours of use with a $300mAh$ lithium battery. Currently containing wireless data transmission (through Wi-Fi) and achieving a $59Hz$ sampling rate.

The electrical schematic diagram containing the operating circuit of each module and appropriate connections between them, as well as the board design for the PCBs are presented in Appendix B.

4.3 Software Implementation - OpenAR Animation

The final prototype was tested directly in the OpenAR platform, by importing the hand model proposed in section 4.1, an example of the demonstration is depicted in figure 4.9.

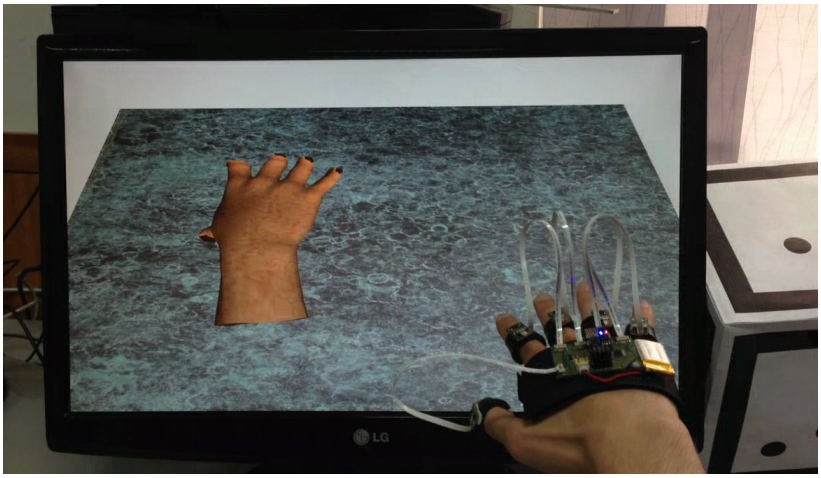

Fig. 4.9 Final Hand-Tracker demonstration.

The OpenAR is an in-house evolving development platform designed to implement Augmented and Virtual Reality applications. It is written in C/C++ language, containing various open source libraries, such OpenCV and OpenGL, as well as interfaces with the most common devices used in the context of these fields.

In terms of software, communication is established through network sockets, using UDP, which allows a user to not only receive the orientation data but also to send commands to access different implemented functions. Functions to stop or reset the data stream, to acquire raw data and recalibrate the sensors if needed. The kinematics were applied to the hand model by expanding and adapting the body kinematics implementation provided by the Kinect SDK, this way maintaining a consistent approach for future applications (using also the Kinect sensor for body tracking).

This demonstration provided a constant and faithful tracking of the hand aside from some model deformations due to the placement of the joints, resulting in some abnormal mesh behaviours in specific positions (most noticeable on the thumb joints). The following figures show examples of two types of control tested on the hand model. Figure 4.10 depicts the intended behaviour of the hand, the model reference joint is placed in a fixed position and the control of the hand starts at the wrist, this way considering that we have our forearm secured (within a full body tracking this joint would be consistent with the far end of the forearm).

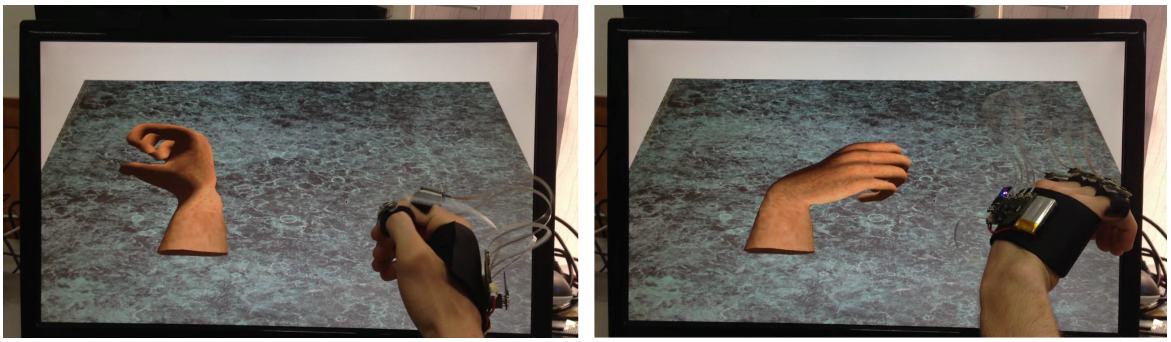

Fig. 4.10 Intended tracking examples, considering a secured arm.

Figure 4.11 depicts a control starting directly at the model reference joint, this approach does not consider rotations on the wrist but provides additional range of motion within this specific demonstration.

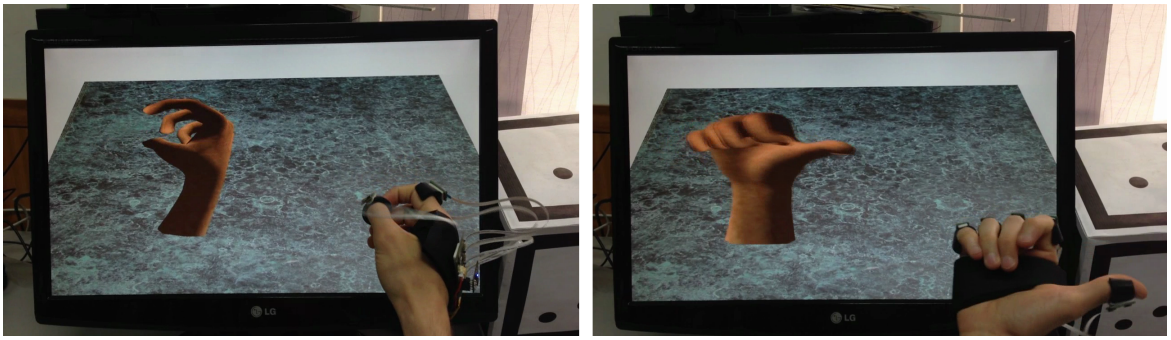

Fig. 4.11 Tracking examples with a different kinematic approach.

Chapter 5

Conclusion and Future Work

The ambition of this dissertation was to find a solution for adding hand-motion capture and hand-interaction in a virtual reality environment. The existing solutions either hinder the hand natural gestures or present limitations that would disrupt the sense of immersion of a user during their virtual tours. In the compromises between the available functionalities and the flexibility of the reached solution, we managed to use low-cost technology, obtaining a final battery-operated prototype with wireless communication and an ergonomic design, which feels comfortable to use and does not constrain the hand motion in any way. This was possible because we analysed an orientation filter, that significantly ameliorates the computational load and parameter tuning burdens associated with more conventional approaches. The algorithm proved effective and even with a performance comparable to high quality commercial Kalman-based systems. In summary, the initially proposed objectives for this dissertation were completed with success, the developed Hand-Tracker prototype achieves a faithful tracking of the hand and paves the way to a new form of interaction.

As future work, there are features that can be added to improve the proposed system. Correctly estimating position using the accelerometers can really improve and support the kinect tracking or even complement it in moments of hand occlusion during body motion capture. Not only that, but with position tracking we can further extend the ways to interact with the 3D virtual environments. Reflecting on the hardware, some features of the prototype can also be improved. Incorporating a battery charger in the prototype or, if not suitable, just adding a status indicator, would allow to check the battery level and help prolong its life. To protect the components, we also recommend an encapsulation of the boards and the use of shorter FFC cables. Last but not least , for the full experience of using our hands in an interactive 3D environment a matching left hand prototype has yet to be assembled.

References

[1] H. Henrik Ehrsson, Charles Spence, and Richard E. Passingham. That's My Hand! Activity in Premotor Cortex Reflects Feeling of Ownershipof a Limb. Technical report, August 2004. Science Vol. 305, Issue 5685.

[2] B. Patrão and P. Menezes. A Virtual Reality System for Training Machinery Operators. *International Journal of Online Engineering, Volume 9, Special Issue 8, pp. 53-55, Wien, Austria*, 2013.

[3] L. Almeida, B. Patrão, P. Menezes, and J. Dias. Be the Robot: Human Embodiment in Tele-Operation Driving Tasks. *Ro-Man 2014: The 23rd IEEE International Symposium on Robot and Human Interactive Communication, Edinburgh, UK*, 2014.

[4] J. Garcia Sanchez, B. Patrão, L. Almeida, J. Perez, P. Menezes, J. Dias, and P. Sanz. Design and Evaluation of a Natural Interface for Remote Operation of Underwater Robots. *IEEE Computer Graphics and Applications, Issue: 99*, 2015.

[5] B. Patrão, J. Seabra, S. Pedro, P. Castiilho, and P. Menezes. Immersive Technologies as a Key Tool in Therapeutic Context. *The Compassionate Mind Foundation's 4th International Conference, Manchester, UK*, 2015.

[6] B. Patrão, P. Castiilho, and P. Menezes. Immersive Mixed Reality Systems for Therapy Support. *Eurographics 2016, the 37th Annual Conference of the European Association for Computer Graphics, Lisbon, Portugal*, 2015.

[7] Ivan E. Sutherland. A head-mounted three dimensional display. *Proceedings of AFIPS 68*, pages 757–764, 1968.

[8] Laura Dipietro, Angelo M. Sabatini, and Paolo Dario. A Survey of Glove-Based Systems and Their Applications. *IEEE Transaction on Systems, Man, and Cybernetics—Part C: Application and Reviews, Vol. 38, No. 4*, pages 461–482, July 2008.

[9] A. M. Mohd Ali, R. Ambar, M. M. Abdul Jamil, A. J. Mohd Wahi, and S. Salim. Artificial Hand Gripper Controller via Smart Glove for Rehabilitation Process. *2012 International Conference on Biomedical Engineering (ICoBE), Penang*, February 2012.

[10] Silvia Pabon, Edoardo Sotgiu, Rosario Leonardi, Cristina Brancolini, Otniel Portillo-Rodriguez, Antonio Frisoli, and Massimo Bergamasco. A data-glove with vibro-tactile stimulators for virtual social interaction and rehabilitation . *PRESENCE, 10th Annual International Workshop on Presence*, pages 345–348, 2007.

[11] Zhengyou Zhang. Microsoft Kinect Sensor and Its Effect. *Published by the IEEE Computer Society in Multimedia at Work, Volume 19, Issue 2*, pages 4–10, February 2012.

[12] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Eyal Krupka, Andrew Fitzgibbon, Shahram Izadi, and Pushmeet Kohli. Accurate, Robust, and Flexible Real-time Hand Tracking. *CHI 2015, Crossings, Seoul, Korea*, 2015.

[13] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the Accuracy and Robustness of the Leap Motion Controller. *Department of Computer Science VII, Technical University Dortmund, Dortmund 44221, Germany*, 2013.

[14] E. Foxlin. *"Motion Tracking Technologies and Requirements," Handbook of Virtual Environment Technologies*. K. Stanney, ed., Lawrence Erlbaum Publishers, 2002, chap.8, p. 182, 2002.

[15] Angelo M. Sabatini. Quaternion-Based Extended Kalman Filter for Determining Orientation by Inertial and Magnetic Sensing. *IEEE Transactions on Bimedical Engineering, Vol. 53, No. 7*, July 2006.

[16] Nima Enayati, Elena De Momi, and Giancarlo Ferrigno. A Quaternion-Based Unscented Kalman Filter for Robust Optical/Inertial Motion Tracking in Computer-Assisted Surgery. *IEEE Transactions on Instrumentation and Measurment, Vol. 64, No. 8*, August 2015.

[17] Tae Suk Yoo, Sung Kyung Hong, Hyok Min Yoon, and Sungsu Park. Gain-Scheduled Complementary Filter Design for a MEMS Based Attitude and Heading Reference System. *Sensors, vol. 11, doi: 10.3390/s110403816*, pages 3816–3830, March 2011.

[18] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *"Sequential Monte Carlo Methods in Practice," Statistics for Engineering and Information Science*. 2001.

[19] Yafei Ren and Xizhen Ke. Particle Filter Data Fusion Enhancements for MEMS-IMU/GPS. *Intelligent Information Management, doi:10.4236/iim.2010.27051*, pages 417–421, June 2010.

[20] Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and Per-Johan Nordlund. Particle Filters for Positioning, Navigation and Tracking. *IEEE Transactions on Signal Processing, Volume 50 Issue 2*, pages 425–437, February 2002.

[21] Dung Duong Quoc1, Jinwei Sun, Van Nhu Le, and Nguyen Ngoc Tan. Sensor Fusion based on Complementary Algorithms using MEMS IMU. *International Journal of Signal Processing, Image Processing and Pattern Recognition Vol. 8, No. 2*, pages 313–324, 2015.

[22] Mark Euston, Paul Coote, Robert Mahony, Jonghyuk Kim, and Tarek Hamel. A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2008.

[23] Sebastian O.H. Madgwick, Andrew J.L. Harrison, and Ravi Vaidyanathan. Estimation of IMU and MARG orientation using a gradient descent algorithm. *IEEE International Conference on Rehabilitation Robotics Rehab Week Zurich, ETH Zurich Science City, Switzerland*, July 2011.

[24] Fakhri Alam, Zhou ZhaiHe, and Hu JiaJia. A Comparative Analysis of Orientation Estimation Filters using MEMS based IMU. *2nd International Conference on Research in Science, Engineering and Technology (ICRSET'2014)*, March 2014.

[25] Joe Seeger, Martin Lim, and Steve Nasiri. Development of high-performance, high-volume consumer MEMS gyroscopes. *In Proc. Tech. Dig. Solid-State Sensors Actuators Microsystems Workshop, 1197 Borregas Avenue, Sunnyvale, CA 94089, USA*, pages 61–64, June 2010.

[26] STMicroeletronics. Everything about STMicroelectronics' 3-axis digital MEMS gyroscopes. Technical report, STMicroeletronics, July 2011. Doc ID 022032 Rev 1.

[27] Matej Andrejasic. MEMS Accelerometers. Technical report, University of Ljubljana, Faculty for mathematics and physics, Department of physics, March 2008.

[28] STMicroeletronics. Using LSM303DLH for a tilt compensated electronic compass. pages 1–34, 2010.

[29] Alberto Pretto and Giorgio Grisetti. Calibration and performance evaluation of low-cost IMUs. *In Proceedings of the 20th IMEKO TC4 International Symposium, Benevento, Italy*, pages 429–434, September 2014.

[30] Donald W. Marquardt. An algorithm for Least-Squares Estimation of Nonlinear Parameters. *Published in: Journal of the Society for Industrial and Applied Mathematics, Vol. 11, No. 2*, pages 431 – 441, June 1963.

[31] A. Pretto, G. Grisetti, and E. Menegatti. A Robust and Easy to Implement Method for IMU Calibration without External Equipments. *In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2014), Hong Kong, China*, pages 3042 – 3049, June 2014.

[32] Valerie Renaudin, Muhammad Haris Afzal, and Gérard Lachapelle. Complete Triaxis Magnetometer Calibration in the Magnetic Domain. *Journal of Sensors*, 2010. Article ID 967245.

[33] Freescale Semiconductor. Calibrating an eCompass in the Presence of Hard- and Soft-Iron Interference. Technical report, November 2015. Document Number AN4246 Rev. 4.0.

[34] CHRobotics. Understanding Euler Angles. [http://www.chrobotics.com/library/understanding-quaternions Online; accessed 24-August-2016].

[35] Oliver J. Woodman. An introduction to inertial navigation. Technical Report UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, August 2007.

[36] David H. Titterton and John L. Weston. *Strapdown Inertial Navigation Technology - 2nd Edition (Radar, sonar, navigations & avionics)*. The Institution of Electrical Engineers, 2004.

[37] Fakhri Alam, Zhou Zhaihe, and Hu Jiajia. A Comparative Analysis of Orientation Estimation Filters using MEMS based IMU. *2nd International Conference on Research in Science, Engineering and Technology (ICRSET'2014)*, pages 86–91, 2014.

[38] Tarek Hamel and Robert Mahony. Attitude estimation on SO (3) based on direct inertial measurements. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 2170–2175, May 2006.

[39] Sebastian O. H. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Technical report, April 2010.

[40] David Jurman, Marko Jankovec, Roman Kamnik, and Marko Topic. Calibration and data fusion solution for the miniature attitude and heading reference system. *Faculty of Electrical Engineering, University of Ljubljana, Trzaska cesta 25, SI-1000, Ljubljana, Slovenia*, May 2007.

[41] Fakhri Alam, Zhou ZhaiHe, and Hu JiaJia. A Comparative Analysis of Orientation Estimation Filters using MEMS based IMU. *2nd International Conference on Research in Science, Engineering and Technology (ICRSET'2014)*, March 2014.

[42] Agneta Gustus, Georg Stillfried, Judith Visser, Henrik Jörntell, and Patrick van der Smagt. Human hand modelling: kinematics, dynamics, applications. *Published in: Journal Biological Cybernetics archive Volume 106 Issue 11-12*, pages 741 – 755, November 2012.

[43] Ian M. Bullock, Julia Borràs, and Aaron M. Dollar. Assessing Assumptions in Kinematic Hand Models: A Review. *The Fourth IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics Roma, Italy.*, June 2012.

[44] Salvador Cobos, Manuel Ferre, M.A. Sanchéz Urán, Javier Ortego, and Cesar Peña. Efficient Human Hand Kinematics for Manipulation Tasks. *IEEE/RSJ International Conference on Intelligent Robots and Systems Acropolis Convention Center Nice, France*, September 2008.

[45] T. Feix, R. Pawlik, H.-B. Schmiedmayer, J. Romero, and D. Kragic. The generation of a comprehensive grasp taxonomy, June 2009.

[46] I. PAJĄK. The Matlab Toolbox for Modeling Complex Mechanims. *International Journal of Applied Mechanics and Engineering. Volume 19, Issue 2*, pages 285 — 301, August 2014.

[47] Mary C. Hume, Harris Gellman, Harry McKellop, and Robert H. Brumfield Jr. Functional range of motion of the joints of the hand. *In: The Journal Of Hand Surgery*, April 1990.

[48] InvenSense Inc. MPU-9150 Product Specification Revision 4.3, September 2013. [https://www.invensense.com/wp-content/uploads/2015/02/MPU-9150-Datasheet.pdf] Accessed on: 26-08-2016.

[49] CHRobotics. Understanding Euler Angles. [http://www.chrobotics.com/library/understanding-euler-angles Online; accessed 24-August-2016].

[50] D. Eberly. Rotation representations and performance issues. Technical report, 2002. Geometric Tools, LLC, Tech. Rep.

Appendix A

Mathematical Background

This Appendix is a summary of representations of rotations by matrices, Euler angles, and quaternions. Although the reader is assumed familiar with these topics, it intends to give a brief description with some mathematical notions and their notations.

A.1 Matrix Representation

A 2D rotation is a transformation of the form:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{A.1}$$

where θ is the angle of rotation. A 3D rotation is a 2D rotation that is applied within a specified plane that contains the origin. Such a rotation can be represented by a 3×3 rotation matrix $R = [R_0 \quad R_1 \quad R_2]$ whose columns R_0, R_1, and R_2 correspond to the final rotated values of the standard basis vectors $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$, in that order. These form a right-handed orthonormal set, that is, $|R_0| = |R_1| = |R_2| = 1$, $R_0 \cdot R_1 = R_0 \cdot R_2 = R_1 \cdot R_2 = 0$, $R_0 \cdot R_1 \times R_2 = 1$. Resulting in very important properties that allow to reduce some calculations, $R^T = R^{-1}$ and $det(R) = 1$.

A rotation about one of the axis of a coordinate system by a given θ angle, usually called elementary rotation, results in the following matrices:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) \\ 0 & -sin(\theta) & cos(\theta) \end{bmatrix} \tag{A.2}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{A.3}$$

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

A.2 Euler Angles Representation

Euler angles provide a way to represent the 3D orientation of an object using a combination of three sequential rotations about the axes a frame. These are given by the right hand rule, following the aeronautical convention as in figure A.1, usually the roll (ϕ) angle is around the x axis (normally this is the axis that points in the direction of the body), the pitch (θ) angle is around the y axis (the other axis in the motion plane of the body), and the yaw angle (ψ) around the z axis.

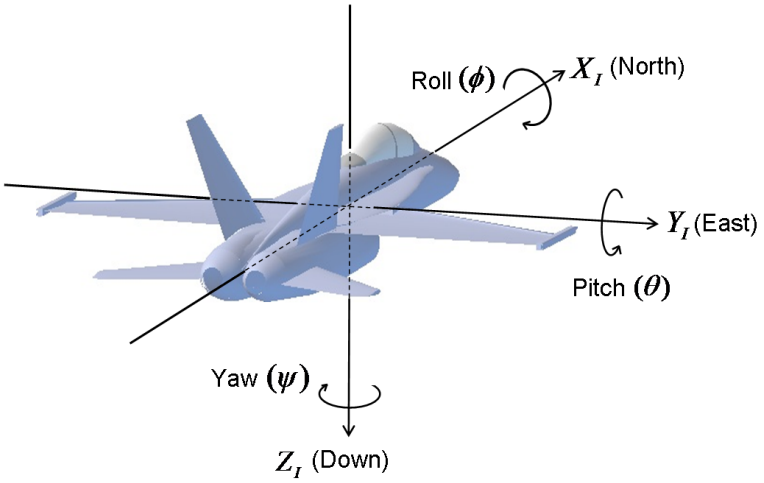

Fig. A.1 Euler angles as described in aeronautical conventional.

Considering that there are no consecutive rotations around the same axis, there are 12 different rotation sequences which can fully describe the final orientation of any body. By multiplying the sequence of elementary rotations, the orientation between two generic frames, A and B, can be described by a rotation matrix. Considering the successive rotations $R_z(\psi)$, $R_y(\theta)$ and $R_x(\phi)$, performing the multiplication, and letting c represent cos and s represent sin, the complete rotation from a frame A to a frame B is given by:

$$ {}^{B}_{A}R(\phi,\theta,\psi) = R_x(\phi)R_y(\theta)R_z(\psi) \tag{A.5} $$

$$ {}^{B}_{A}R(\phi,\theta,\psi) = \begin{bmatrix} c(\psi)c(\theta) & c(\theta)s(\psi) & -s(\theta) \\ c(\psi)s(\phi)s(\theta)-c(\phi)s(\psi) & c(\phi)c(\psi)+s(\phi)s(\psi)s(\theta) & c(\theta)s(\phi) \\ s(\phi)s(\psi)+c(\phi)c(\psi)s(\theta) & c(\phi)s(\psi)s(\theta)-c(\psi)s(\phi) & c(\phi)c(\theta) \end{bmatrix} \tag{A.6} $$

Euler angles suffer from singularities commonly know as gimbal lock, making sensors unable to correctly represent orientation on specific cases. The exact orientation at which gimbal lock occurs depends on the order of rotations used. In the sequence used previously, the order of operations results in gimbal lock when the pitch angle is at 90 degrees. On this situation, yaw and roll cause the sensor to move in exactly the same fashion. This is a fundamental problem of Euler Angles and can only be solved by switching to a different representation method [49].

A.3 Orientation Representations Assessment

As with most computer science topics, there is no answer to which is the best representation, but depending on the application there may be trade-offs to consider.

Quaternions were chosen as the mathematical ground because, in the context of this work, they deliver some advantages over the other representations. In terms of memory usage, a rotation matrix yields a total of 9 parameters, resulting in 9 floats, a quaternion requires 4 floats and the Euler angles only 3 integers. Compared to quaternions and matrices, Euler Angles provide the best choice, they are simple, intuitive and they lend themselves well to simple analysis and control. On the other hand, Euler Angles are limited by the phenomenon known as "gimbal lock", while the others have no problems with these singularities. Comparing performances in terms of low level operation, as done by David Eberly in [50], when composing two rotations the product of two rotation matrices requires 27 multiplications and 18 additions, whereas the product of two quaternions requires only 16 multiplications and 12 additions. Euler angles are not even used because it implies the use trigonometric functions which are is quite expensive, so clearly quaternions present the best solution in an application that requires computational efficiency.

Appendix B

Schematic and Board Diagrams

Figure B.1 depicts the schematic diagram of the designed PCB for the primary board of the Hand-Tracker, with the operating circuit of each module and appropriate connections between them.

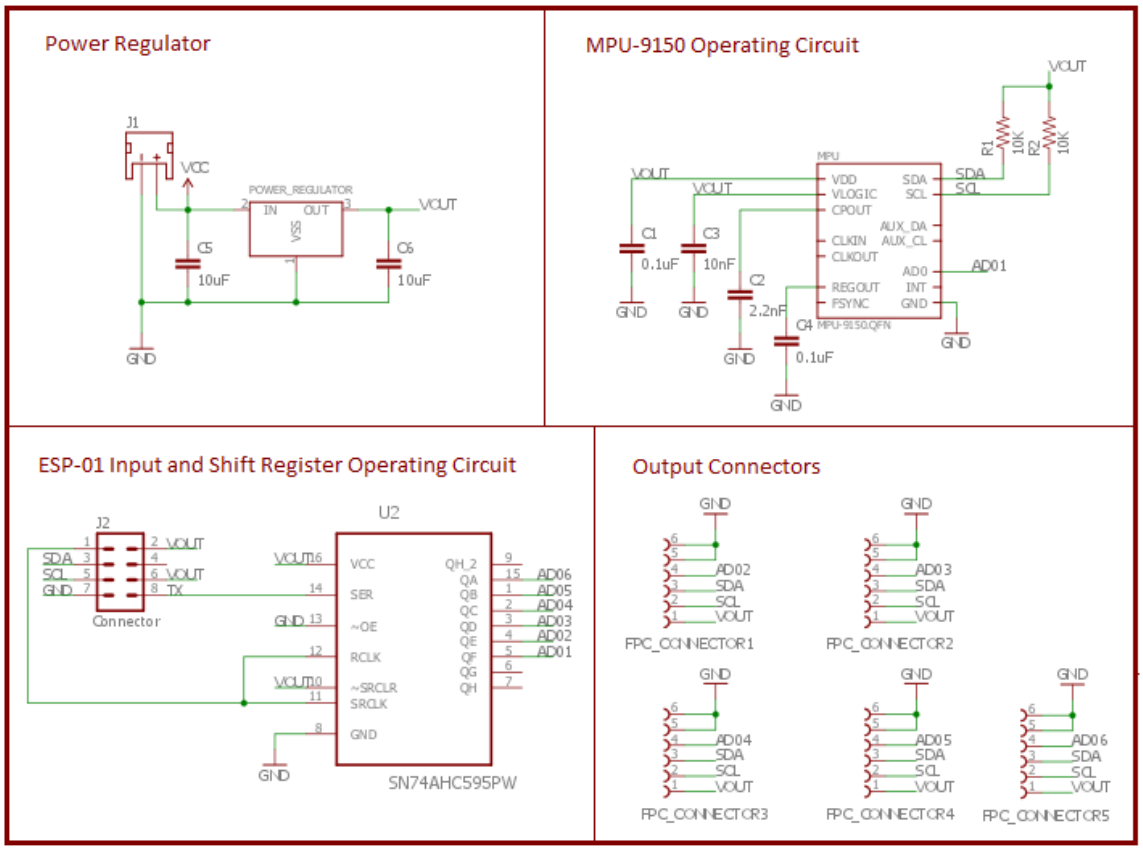

Fig. B.1 Primary board schematic diagram.

Figure B.2 depicts the board design of the PCB for the primary board. The top board design on the figure contains all the connection on both layers of the PCB without the layout of the ground plane (dimensions in millimetres). The bottom left board design contains the first layer of the PCB (top view) with the corresponding ground plane. Similarly, the bottom right board design contains the second layer of the PCB (bottom view) with the corresponding ground plane.

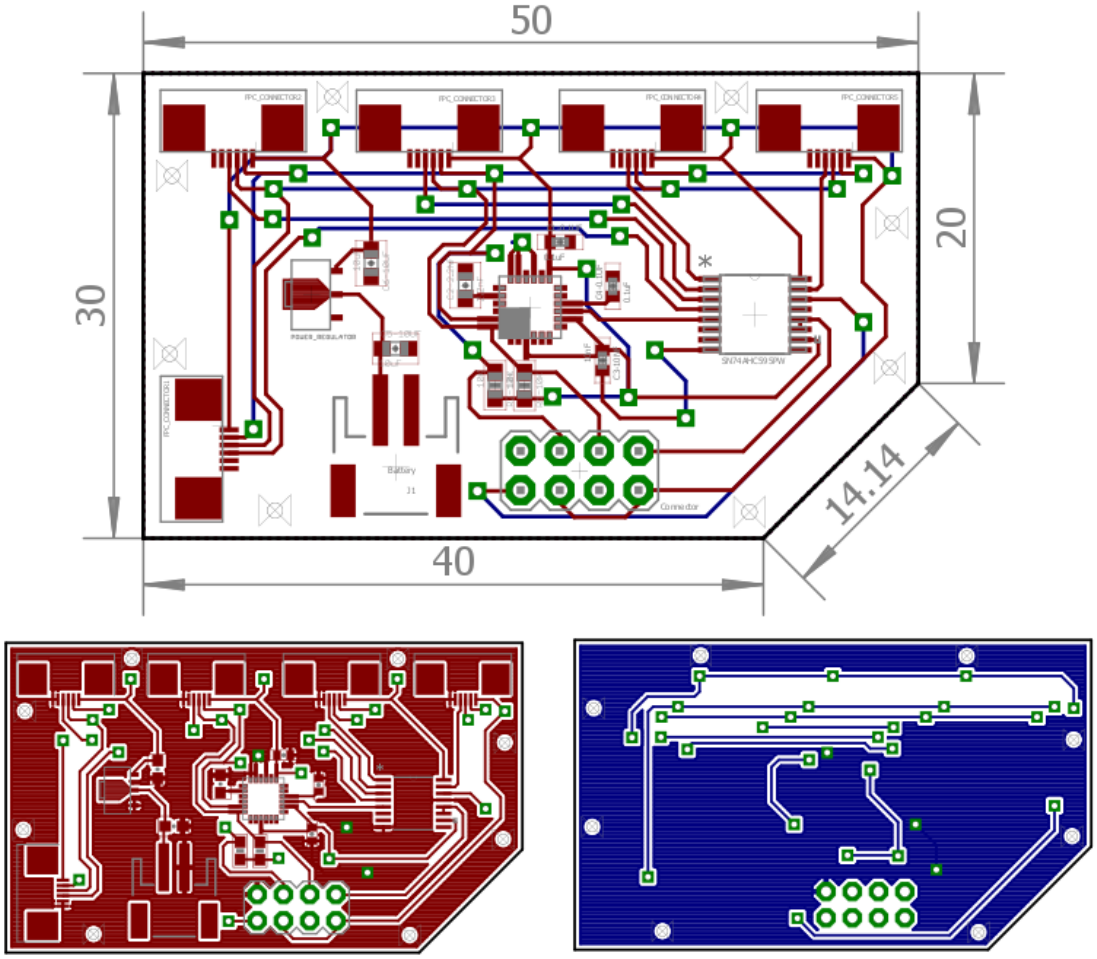

Fig. B.2 Primary board PCB design.

Figure B.3 depicts the schematic diagram of the designed PCB for the finger board of the Hand-Tracker, only containing the operating circuit of the MPU-9150 sensor and appropriate output connector.

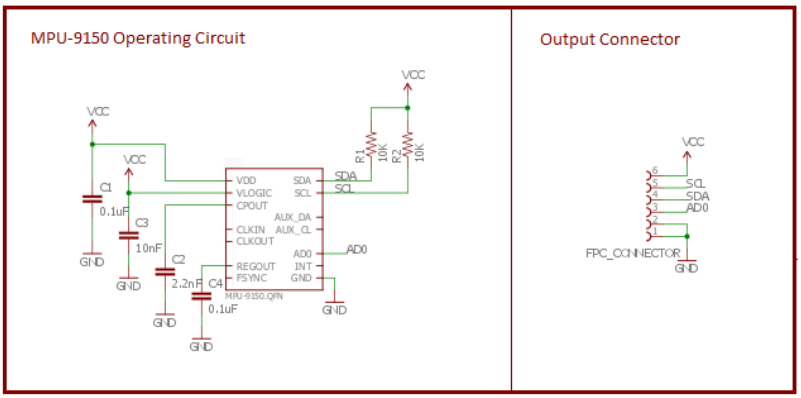

Fig. B.3 Finger board schematic diagram.

Figure B.4 depicts the board design of the PCB for the finger board. In accordance with figure B.2, the left board design contains all connection except the ground plane, the centre board design depicts the first layer of the PCB (top view) and the right board design depicts the second layer (bottom view), both with the respective ground plane

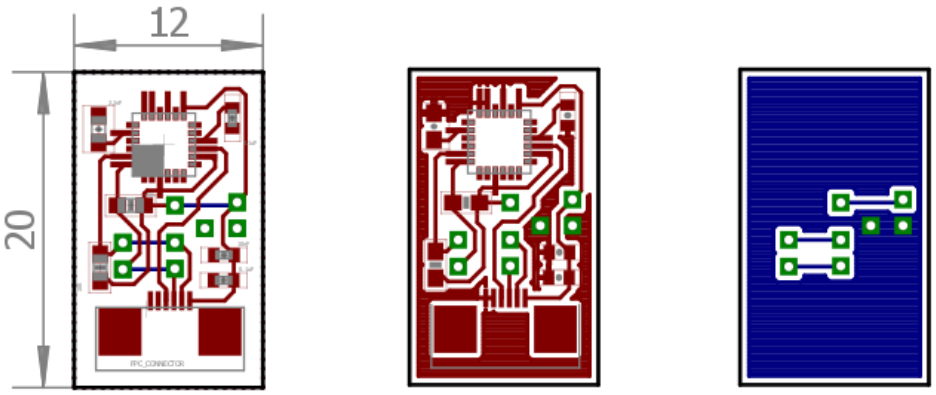

Fig. B.4 Finger board PCB design.