An IMU tracking algorithm for 3D motion reconstruction using OpenSim dynamical models

Implementation on a robot manipulator

P.A.M. de Kanter

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

An IMU tracking algorithm for 3D motion reconstruction using OpenSim dynamical models

Implementation on a robot manipulator

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

P.A.M. de Kanter

February 3, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Inertial Measurement Unit (IMU)-based motion capture has gained interest over the years due to its potential to measure human movement in the clinic and on the sports field at low cost. Still, IMU-based motion reconstruction remains a challenging task as these IMU measurements are corrupted by noise and bias. There have been many filtering and sensor fusion algorithms developed to address this problem, but limited attention has been paid to including the system dynamics of the subject to estimate kinematics and kinetics from multiple IMUs. I propose a novel motion reconstruction algorithm for capturing 3D motions in a markerless and unconstrained environment using gyroscope and accelerometer measurements. The algorithm is based upon an Iterated Extended Kalman Filter for state estimation and the 3D dynamical model of the subject created in OpenSim (IEKF-OS). The IEKF-OS algorithm consists of two stages. The first stage incorporates the dynamics of the subject to predict the motion. The second stage tracks measurements from multiple IMUs to improve model estimates of joint angles and speeds. The goal of this thesis is threefold. Firstly, the IEKF-OS is derived and verified using simulations performed on a six-link robot manipulator including sensor placement errors. Secondly, experiments are performed to validate the algorithm's estimations against the robot's ground truth joint encoder values. Thirdly, the algorithm is compared against an inverse kinematics method to track IMU estimated orientations (OpenSense) provided by OpenSim. The IEKF-OS algorithm showed lower motion tracking errors compared to OpenSense for various motions performed by the robot manipulator. The joint angle estimations as computed by both methods are compared against the gold-standard ground truth robot encoder values. OpenSense joint angle values were in the range of 0.6-6.4 [deg] RMSE, whereas the IEKF-OS algorithm estimated joint angles and speeds in the range of 0.4-2.8 [deg] and 0.3-2 [deg/s] RMSE, respectively. These results highlight accurate 3D motion reconstruction on a six-link robot manipulator. Contrary to OpenSense, the IEKF-OS is able to accurately reconstruct motions regardless of magnetic disturbances because it does not rely on magnetometer data.

Table of Contents

List of Figures

List of Tables

Preface

Prior to applying for the Systems and Control Master's degree program, I did my Bachelor's thesis in the field of Mechanical Engineering. During this thesis, I joined a team of multidisciplinary students with the aim to develop a sensor system for patients with Parkinson's disease. My focus was to estimate spatio-temporal gait characteristics using acceleration data logged by an IMU positioned at the lower back region to predict when a patient would experience a freeze of gait event. Having enjoyed this project thoroughly, I knew that I wanted to continue my studies and noticed that the TU Delft offered a BioMechanical Design Master's program. Despite being intrigued by the combination of biomechanical systems and sensor fusion, I ended up applying for the Systems and Control Master, not knowing that Dr. Manon Kok, an expert in the field of sensor fusion, would eventually become my first supervisor during my thesis project. After our first meeting, I couldn't be happier. Dr. Kok had a joint project in mind for me together with Dr. Ir. Ajay Seth. During Dr. Seth's time at Stanford University, he, together with other people in Biomechanical engineering, created an open-source modeling and simulation software package for biomechanical systems. After the first meeting with the three of us, I became very enthusiastic as I could now focus my thesis project on this exciting combination of sensor fusion and biomechanical modeling.

Acknowledgements

First, I would like to thank my supervisors, Dr. Manon Kok and Dr. Ir. Ajay Seth. You helped me not only to develop myself academically but also personally. Particularly the advice: *"Be the captain of your own ship"*, and *"If it doesn't work, go back and build it again piece by piece."*, are quotes I will always keep in mind for my further career. I am very grateful for the amazing project you gave me, which I really enjoyed.

Apart from my supervisors, I also want to thank Dr. Ir. Luka Peternel for preparing and doing the robot experiments. The fact that I was able to do these experiments made my thesis project not only more exciting and fun but also pushed the value of this work to a higher level. This due to the comparisons I was able to make between the robot joint encoder values and my algorithm's estimations.

My ambition has always been to help people. As such, I was happy to meet two PhD candidates, Bart van Trigt and Ton Leenen, during the CBL meetings that my supervisor Dr. Ir. Seth organized. I hope that we can apply this algorithm in the future to help people recover from and prevent sports injuries by analyzing their reconstructed motions.

Moreover, I would also like to thank Evan, Frida, Thomas, Bart, Stefan, and Janneke for the bi-weekly MSc thesis meetings in which I got the opportunity to learn from your projects and insights.

Next to that, I want to thank all the friends I made during these study years in Delft. Studying together at the university made it much more enjoyable. I never thought that I would meet such a nice group of friends during my Master's degree program.

In particular, I want to thank my roommates Pepijn and Thijs. Even though I faced some difficulties during this thesis project, I am very grateful for the motivation you gave me, and the conversations and discussions we had. Especially, I really value the amazing memories we made.

Finally, I want to thank my amazing parents and brother, who I always was able to count on. During this past year, you always assisted me with good advice. Even if you did not really understand the technical problems I faced, you were always there to listen to me and motivate me to take the next step forward.

Delft, University of Technology — P.A.M. de Kanter
February 3, 2021

"Part of this thesis is *serendipity*: allowing for chance finds, and accepting that what is found is not necessarily what was being looked for."

Glossary

List of Acronyms

OMC	Optical Motion Capture
IMC	Inertial Motion Capture
IMU	Inertial Measurement Unit
IMUs	Inertial Measurement Units
MEMS	Micro Electro-Mechanical System
IK	Inverse Kinematics
ID	Inverse Dynamics
2D	Two-dimensional
3D	Three-dimensional
MATLAB	MATrix LABoratory
API	Application Programming Interface
GUI	Graphical User Interface
KF	Kalman Filter
EKF	Extended Kalman Filter
IEKF	Iterated Extended Kalman Filter
IEKF-OS	Iterated Extended Kalman Filter - OpenSim
XKF3hm	Xsens Kalman Filter for 3D human motion
HOT	Higher Order Terms
DoFs	Degrees of Freedom
jc	Joint center
EoM	Equations of Motion
MBS	Multibody System
rad	Radians
deg	Degrees
RMSE	Root-mean-square error

List of Symbols

α	Scaling factor for the accelerometer calibration
α	Translation perturbation value
β	Rotation perturbation value
β	Gain value in Madgwick orientation estimation algorithm
ϵ	Threshold for relative change between subsequent IEKF-OS iterations
ϵ	Size of the perturbation step size
ε	Error
ϕ	Angle of rotation around the x-axis
θ	Angle of rotation around the y-axis
ψ	Angle of rotation around the z-axis
θ	Angle of rotation
ω	Angular velocity
$\dot{\omega}$	Angular acceleration
μ	Mean of measured sensor signal
Σ	Covariance
σ	Standard deviation
τ	Control input/control torque/joint torque
χ	Vector used for evaluation in numerical Jacobian scheme
a	Acceleration
b	Bias
b	Body frame
D	Calibration matrix
e	Noise
F	Derivative to the state in the Extended Kalman Filter
f	Specific force
G	Global frame / Ground frame
g	Gravity force
H	Output derivative to the state in the Extended Kalman Filter
h	Sample period
I	Inertia tensor
K	Kalman gain
k	Iteration
L	Derivative to the noise in the Extended Kalman Filter
l	Length of rigid body segment
M	Output derivative to the noise in the Extended Kalman Filter
m	Mass of rigid body segment
n	Navigation frame
N	Amount of measurements
N_{LGF}	Norm of gravitational field
n	Order of the system
P	State process covariance matrix
Q	Process noise covariance matrix

q	Quaternion
q	Generalized coordinate
$\dot{q}$	Derivative of the generalized coordinate
R	Measurement noise covariance matrix
R	Rotation matrix
r	Position vector
r	Radius of rigid body segment
$\mathcal{S}$	Set of attached sensors
s	Sensor frame
t	Time instant
Δt	Discretization time step
u	Generalized coordinate velocity
$\dot{u}$	Generalized coordinate acceleration
$\mathcal{V}$	Set of attached virtual sensors
v	Virtual sensor frame
w	Process noise
x	State vector
y	Measurement
x	Cartesian coordinate in the x-direction
y	Cartesian coordinate in the y-direction
z	Cartesian coordinate in the z-direction

$(\cdot)^{\mathrm{c}}$	Quaternion conjugate
$(\cdot)^{\top}$	Transpose
$(\cdot)^{-1}$	Inverse
$\check{x}$	Prediction given the state transition model
$\hat{x}$	Corrected prediction given the measurement
$\bar{x}$	Arbitrary operating point
$\dot{x}$	Derivative with respect to time
exp	Exponential
cos	Cosine
sin	Sine
$\sum$	Summation
$\mathcal{I}_n$	n by n identity matrix
$\mathcal{O}_n$	n by n matrix of zeros
$\mathbb{R}^n$	Set of real numbers in n
Tr	Trace
F	Function used for evaluation in numerical Jacobian scheme
H	Hinge matrix/joint transition matrix
$\dot{H}$	Derivative of the hinge matrix/joint transition matrix
N	Kinematic coupling matrix
$\mathcal{N}(0, \Sigma)$	Zero-mean Gaussian noise with covariance Σ
X^{AB}	Transform defining frame B measured from and expressed in frame A
R^{AB}	Rotation matrix expressing the orientation of frame B in the frame A
p^{AB}	Translation vector from A frame to B frame
det	Determinant

$\cdot$	Dot product
$\times$	Cross product
$[\times]$	Matrix-vector product operator
$\odot$	Quaternion multiplication
$\otimes$	Kronecker product
vec	Vectorization operator
$\lvert \cdot \rvert$	Absolute value
$\lvert\lvert \cdot \rvert\rvert_2$	Magnitude
$\approx$	Approximately
$<$	Less than
$\leq$	Less than or equal to
$=$	Is equal to
$\geq$	Larger than or equal to
$>$	Larger than
$\neq$	Not equal to

Chapter 1

Introduction

In this first chapter, the motivations behind this research are posed. Three research questions will be defined after which the main contributions of this thesis will be presented. Lastly, this introductory chapter is concluded with the outline of this thesis work.

1-1 Research motivations

The field of motion capture, also referred to as motion reconstruction, has gained interest over the years due to its widespread applicability [1]. Motion capture is the process of recording the motion of people or objects [2]. Application areas are very diverse and range from analyses in the domains of robotics and biomechanics, to capturing motions for the game and movie entertainment industry, see e.g. [3], [4]. These areas make it an exciting and attractive topic to research.

In the past years, various methods have been developed for human motion capture and analysis. Cappozzo et al. (2005) [5] formulated human motion analysis as "*the science that aims at gathering quantitative information about the mechanics of the musculoskeletal system during the execution of a motor task*". From these various methods, two main motion reconstruction directions can be recognized. One direction of research focuses on tracking motions by attaching reflective markers on rigid bodies. Optoelectronic camera-based systems are then used to identify the 3D motions of these body segments. Approaches based on these kinds of setups are commonly denoted as Optical Motion Capture (OMC). This capture technique, however, has its drawbacks. It is rather expensive due to the large set of cameras needed, and as such, can mostly only be used indoors. This is a critical downside as it limits the motions possible to be tracked, i.e., sport and daily life activities. For example, when clinicians desire to monitor the rehabilitation process of patients at home and use this information to support their medical diagnoses made [6]. As such, with this aim of eventual unconstrained and environment independent human motion analysis in mind, techniques in the field of OMC seem unsuitable.

An alternative is Inertial Motion Capture (IMC), which relies on Inertial Measurement Units (IMUs). IMUs are tri-axial sensors that are becoming a serious candidate for human motion analysis [7]. Especially due to the advances in Micro-Electro-Mechanical System (MEMS) technology, these IMUs have become small, wearable, inexpensive, and easy in use [8]. These advantages make them suitable for continuous human motion analysis [9]. An IMU, depicted in Figure 1-1, typically consists of two components, being a gyroscope and an accelerometer. The gyroscope measures the sensor's angular velocity around each of its three axes. Integrating these angular velocity measurements yields information about the sensor's 3D orientation. The accelerometer measures the external specific force acting on the sensor. This force consists of both the earth's gravity and the sensor's acceleration. By fusing the measurements made by these two components in a sensor fusion algorithm, the IMU's orientation, as well as its position can be estimated [10].

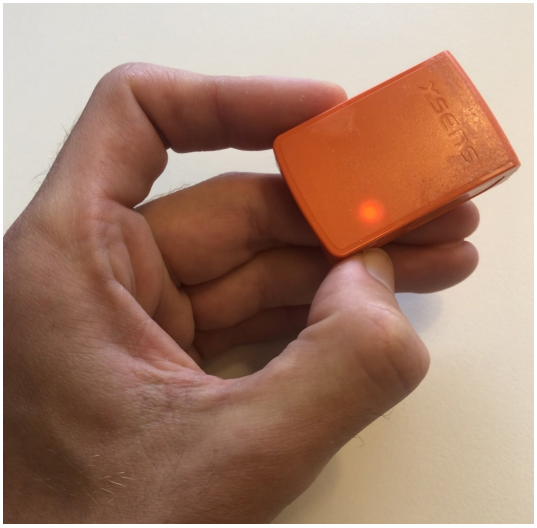


Figure 1-1: The Xsens MTw Awinda wireless IMU sensor [11].

Unfortunately, these measurements are corrupted by noise and bias making estimations erroneous [12]. For orientation estimation, in the sensor fusion algorithm of [10] the gyroscope measurements need to be integrated once. Moreover, for position estimation, the accelerometer measurements need to be integrated twice. As a result, accumulations of error evolve due to the integration of this noise and bias present in these measurements. This leads to so-called integration drift [10], degrading the quality of these estimations made, resulting in unreliable information.

For that reason, researchers have been looking into physics-based approaches using kinematic chain models or musculoskeletal models. A kinematic chain model is an assembly of rigid bodies, connected by joints to constrain its motion. Musculoskeletal models are more advanced. Next to the joints incorporated, this computational model encompasses a skeleton and has muscles spanning between the rigid bodies. These muscles are able to generate forces resulting in the model's movement. Using these models, more information can be taken into account. Together with global optimizations or Extended Kalman filters, these models can be used to constrain the motions possible [7],[13],[14],[15]. However, the kinematic chain models seem to lack the complexity required to estimate human kinetics and do not take into account physical ranges of motion. In contrast, reconstructing motions with a musculoskeletal model

could open the doors to an extensive analysis of kinematic and kinetic related quantities. Kinematics is the branch of mechanics that considers the motion of an object in terms of displacement, velocity, and acceleration. It does, however, not take into account the forces that produce this motion. Kinetics, on the other hand, is the branch of mechanics that studies the relationship between the forces acting on the body and the changes it produces in the body's motion [16].

However, to date, limited attention has been paid to including the system dynamics of the subject to estimate kinematics and kinetics from multiple IMUs [17]. Koning et al. (2015) [7] demonstrated that kinematically driving a musculoskeletal model using IMU orientation estimations solely is feasible. However, they only focused on the kinematics of the model and did not perform inverse dynamic calculations. The limitation is that no forces and torques were estimated. Evaluation of joint torques and muscle forces is important to understand the physiological and mechanical mechanisms of human motion [17].

At the moment, literature seems to be lacking. Only a few papers have looked into combining IMU data with dynamical models, including its inertial properties to estimate human kinematics and kinetics [6][17]. However, the method of [6] only investigated 2D motions. From a research perspective, it is therefore interesting whether including the system's dynamical model combined with multiple IMUs could improve motion reconstruction accuracy. Especially when in addition to kinematic constraints, the system dynamics with its inertial properties are included. The term inertial properties encompasses a rigid body's mass, length, center of mass location, and moment of inertia. From these inertial data-driven models, the kinematics and kinetics can be estimated which are especially useful for clinical studies. As of now, these studies are primarily limited to motion capture laboratories [17].

The recently developed motion reconstruction algorithm by Dorschky et al. (2019) [6] formed the inspiration for this thesis work. Their idea was to create virtual IMUs on the segments of a modeled human body. The outputs of these virtual IMUs then had to match the measurements as logged by the experimental counterpart IMUs which were positioned on the body of the participant. This approach aimed to not only estimate human kinematics but rather also allowed for kinetics estimations. As previously stated, this method is currently only applied to 2D models, disregarding the effects of the unmodeled dynamics in the third dimension. This leaves out important information, from which the authors of [6], considered that this 2D model might have affected the capability to track the IMU data correctly [6]. Therefore, the hypothesis made prior to this thesis is that when incorporating a 3D model together with 3D inertial measurements, motion reconstruction accuracy can be improved. To conclude, the overall goal of this thesis is formulated as

How can the inertial measures of multiple IMUs and the system's dynamical model be used to improve 3D motion reconstruction accuracy?

Therefore, this thesis aims to constitute an important step to boost the accuracy of estimating 3D motions and the torques driving the model by bridging the fields of sensor fusion and biomechanical modeling.

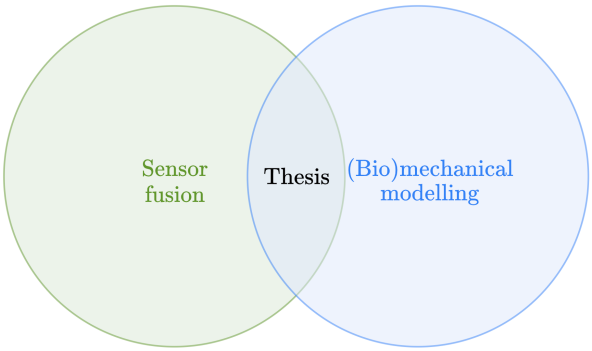


Figure 1-2: This thesis aims to bridge the fields of sensor fusion and biomechanical modelling to improve the accuracy of estimating 3D motions.

Looking at this research question, the idea is thus to develop a novel algorithm, which includes the system's kinematic constraints and its dynamics. It should be noted that the eventual goal of this algorithm is focused on human motion capture. However, it is hard to validate this novel algorithm on human motions. Mainly as there are no ground truth joint angles available. Therefore, the algorithm will be applied to motions performed by the KUKA LBR iiwa 7 R800 robot manipulator. This robot features joint encoders in each link such that the actual joint angles during undergoing movements can be logged. These encoders are considered to be producing ground truth joint angle data allowing for validation. Another reason why no experiments have been performed on human body segments has to do with the placement of IMUs on human skin. This can lead to the violation of assuming that the IMU sensor is positioned fixed to the rigid body segment. This results in so-called soft tissue artifacts (STAs) disturbing the measurements of the IMUs. As such, the algorithm will be tested on the KUKA robot manipulator composed of rigid links. As a robot does not have soft tissue covering its links, no STAs are present. Moreover, it is assumed that it is easier to create a more accurate model of a given robot manipulator with its mechanical joint types. This compared to a model of for instance a participant's upper body with the complex shoulder joint. The latter would require to accurately measure the subject's body segment lengths and estimate its masses together with their moments of inertia. Contrary, information about robot manipulators is specified by the robot manufacturer. With this focus, the overall goal of this thesis can be formulated in more detailed questions presented below.

- ***What method can be developed to exploit the system's nonlinear dynamical model while simultaneously addressing the noise affecting the inertial measurements?***
  Devising an algorithm to combine the information available, will be the cornerstone challenge that initially needs to be addressed. The new method should embrace scalability to systems composed of multiple linked rigid bodies, and be applicable to various systems across different domains. Particularly to facilitate and motivate exploration for future human motion capture research.

- ***How sensitive is the new algorithm to common sensor placement errors when applied to a robot manipulator?***
  With the future goal of applying this method to reconstruct human motions, it is interesting to analyze how accurately these sensors need to be positioned. Still, outcomes should only be seen as an indication of the algorithm's robustness to these errors for the robot manipulator. As such, it must be noted that when answering this question, the results cannot be translated directly to specify how accurately this algorithm requires sensors to be positioned on a human body.

- ***How does the novel algorithm utilizing the system's dynamical model compare to a method which solely uses kinematical constraints in terms of tracking performance for various ranges of motion?***
  To date, there is no comparison showing the added value of a method including these dynamical models compared to a method incorporating only the system's kinematical constraints.

1-2 Contributions of this thesis

Prior to this thesis work, a literature survey was conducted. From this survey, it was hypothesized that tracking experimentally obtained IMU measurements in combination with a (bio)mechanical model, yields a promising method for motion reconstruction. Particularly, as this technique combines the data from different sensors while taking into account the system's dynamics and its kinematical constraints. Therefore, the goal of this thesis is to develop a novel motion reconstruction algorithm incorporating these elements. As previously stated, the initial aim of this algorithm was to reconstruct human motions. However, to verify and validate this novel approach, the method presented, will be applied to the motions performed by the KUKA robot manipulator. The obtained ground truth joint encoder measurements during motion experiments can be used for this validation. The main contributions of this thesis are presented in the following list.

- ***Development of a 3D motion reconstruction algorithm for a generic n-DoF robotic system or human body.***
  In this thesis, a novel motion reconstruction method is derived based on tracking noisy gyroscope and accelerometer data. The algorithm presented is easily applicable to (bio)mechanical systems when the user has a model of this system at hand available. Using the developed algorithm, accurate motion reconstruction is obtained even for systems composed of multiple linked rigid bodies. This work extends the work of [6], who limited their analysis to 2D motions, to reconstruct 3D motions. Moreover, the novel approach derived in this thesis realized a different approach. It combines filtering of measurements and estimation of the control input in one block leading to 3D motion reconstruction.

- ***Verification and validation of the motion reconstruction algorithm on a 7 DoF robotic system.***
  Firstly, the performance of the technique presented will be verified using numerical simulations. Results obtained from these simulations show that high motion reconstruction

accuracy is obtained for the KUKA robotic manipulator. Next to that, for this system, the method developed shows that it can deal with disturbances such as sensor noise and moderate sensor misalignments errors. Simultaneously, the algorithm still converges when the joint angles of the modeled system are initialized differently compared to the original system. Secondly, validity was assessed by conducting experiments on the KUKA robot under varying movement excitations. The algorithm developed, reconstructs the original motion in presence of naturally occurring sources of error like sensor misalignments and misplacements.

- ***Comparison of motion reconstruction performance with respect to an inverse kinematics method incorporating only kinematical constraints.***
  To evaluate the performance of this novel algorithm, a comparison with an inverse kinematics (IK) based approach is presented. Both methods incorporate the same model and use the same positioned and oriented virtual IMUs allowing for a fair comparison. The differences are, however, that this IK method only includes the system's kinematical constraints and does not include the system dynamics. Moreover, the IK method relies on orientation estimations and thus not on tracking noisy inertial sensor data. It rather constrains the orientation estimates to the underlying model. The joint angle estimations as computed by both methods are compared against the gold-standard ground truth robot encoder values and evaluated using the RMSE metric. The results presented, provide evidence that the novel algorithm has lower tracking errors compared to the IK method for the motions performed by the KUKA robot manipulator.

1-3 Outline of the thesis

Having defined the research questions and thesis contributions, Chapter 2, will focus on the related work which inspired this thesis work. Next to that, other relevant sensor fusion methods are outlined giving an overview of the work in this field. As previously stated, this thesis aims to develop a new algorithm extending the work of Dorschky et al. (2019) [6] who limited their reconstruction to 2D motion, to 3D motion tracking. As such, it is essential to list the requirements for this novel motion reconstruction approach. To that intend, this chapter will conclude with these main requirements.

Following up Chapter 2, in Chapter 3, the novel motion reconstruction algorithm is derived for general n-DoF (bio)mechanical systems. The motion reconstruction problem formulation is posed and the algorithm's three main components are illustrated. These three concepts, being experimental IMU measurements, virtual IMU measurements, and the algorithm's filtering and state estimation capabilities will be explained in great detail. Lastly, an algorithmic overview of the method developed is presented.

In Chapter 4, the motion reconstruction performance will be assessed using numerical simulations on the 7-DoF modeled robot manipulator used for this thesis. The task is to reconstruct the robot's six segment link's 3D motions. As this same system will be used for actual experiments conducted later on, a sensitivity analysis to sensor misplacements and sensor misalignments will be performed. These errors will be introduced on the virtual sensors as translational and rotational offsets. This step should yield insights into how its reconstruction performance for this robotic system is affected by these commonly occurring experimental sources of error.

With the robustness assessed, in Chapter 5, the method presented will be validated on the actual robotic manipulator system proposed in Chapter 4. As this robot features joint encoders in each link, the joint angles as estimated by this novel reconstruction method can be validated against these ground truth joint encoder measurements. This validation constitutes an essential step for the aim of applying this novel algorithm in the field for future research. Moreover, the algorithm will be compared to an inverse kinematics based approach which does not take into account the system's dynamics.

Finally, Chapter 6 will conclude the work presented in this thesis. The research questions posed in this introductory chapter will be answered and the suggestions for future work will be stated. Here, the main challenges and the questions that are unanswered will be pointed out.

Chapter 2

Related work

This chapter will acquaint the reader with the related work, giving a clear overview of the current research in the field of motion reconstruction. These methods presented, are adapted from the literature survey, which was conducted prior to this thesis. Next to that, the challenges addressed and limitations left unanswered of each method are stated. Especially the latter should motivate the need for this novel technique developed in this thesis. Finally, the requirements, which this new motion reconstruction algorithm needs to fulfill, are posed.

2-1 Inspiration from current human motion estimation techniques

While developing this thesis work, knowledge and inspiration were accumulated from current motion estimation techniques. Although the combination of IMU measurements with the system's dynamical model has gained limited attention, the literature that did consider this was carefully studied. Moreover, also the state-of-the-art sensor fusion approach of Weygers et al. (2020) [15] was analyzed to gain more insights. In their study, however, the authors did not include the system's dynamics. Still, a brief outline will be given first in Section 2-2, to gain a broader understanding of how connections of body segments can be integrated into a sensor fusion framework.

This thesis, however, aims to include the system's dynamical model and its kinematical constraints to improve 3D motion reconstruction using IMU measurements. The dynamical model is based on the inertial properties of each rigid body. These inertial properties include the body's length and mass parameters, its center of mass location, and its moment of inertia. A large part of this thesis is based on the fundamentals of the developed method by Dorschky et al. (2019) [6]. In their study, the authors formulated an objective function of tracking experimentally obtained IMU data using virtual sensors attached to a musculoskeletal model. The goal was then to find the appropriate set of control inputs and model joint angles to achieve motion tracking. The control inputs were then applied to drive the model such that a dynamically consistent simulation was obtained from which human gait kinematics and kinetics could be estimated. As this idea of tracking experimentally obtained IMU measurements formed the main principle of this thesis, this method will be further detailed in Section 2-3.

2-2 Existing work on sensor fusion based kinematics estimation

Recently, the study of [15], presented a method that estimates the 3D joint kinematics using IMU data and modeled connections of body segments. These connections were included by tightly coupling the rigid body kinematics in a sensor fusion algorithm which allowed for drift-free joint kinematics estimation. As shown in Figure 2-1(a), two linked human body segments are modeled as a system of two adjacent rigid body segments attached by a spherical joint. The 3D joint type allows all rotational motions. Hence, this technique inherently assumes that between two linked bodies, no rotational restrictions are imposed even when the underlying system does feature this constraint. Each rigid body has a sensor with an individual coordinate system S_i modeled, for $i = 1, 2$. The i-th sensor is positioned at a distance $r_i^{S_i}$ from the joint center to its sensor coordinate center S_i. This distance is estimated from inertial measurement data using the technique of [18].

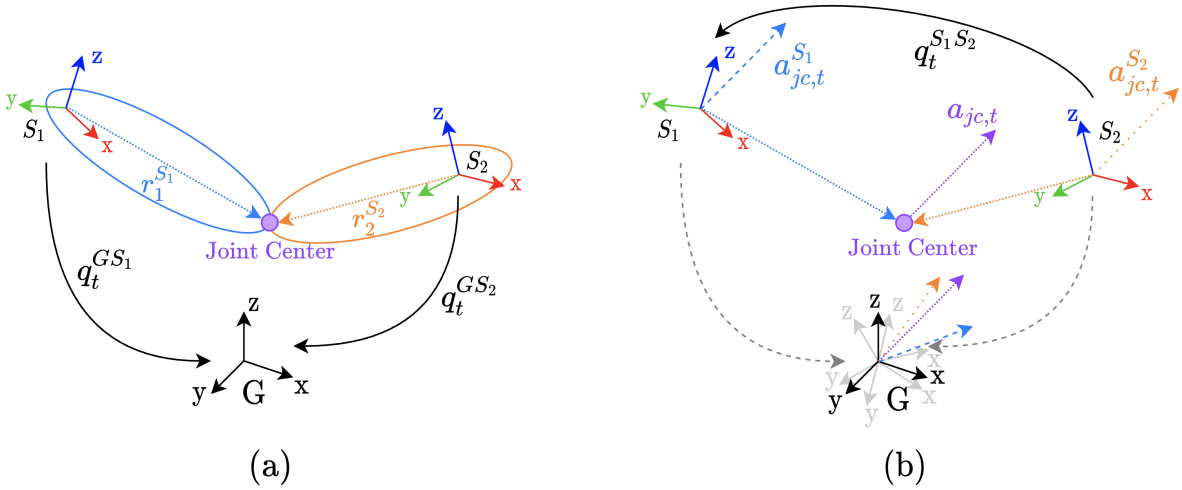


Figure 2-1: (a) Illustration of a kinematic model of two segments, each with an attached IMU, connected by a spherical joint [15]. (b) The joint acceleration $a_{\mathrm{jc},t}$ given in both sensor coordinate frames, $a_{\mathrm{jc},t}^{S_1}, a_{\mathrm{jc},t}^{S_2}$, and both their projections on the common, but moving global coordinate frame G. Figures are adapted from [15].

As described in Chapter 1, due to the integration of gyroscope measurements to obtain orientation estimates, these estimates will drift over time. Commonly, use is made of magnetometers, which yield information about the heading of the sensor by measuring the local magnetic field which compensates for angle drift [10]. This method of [15], however, proposes a magnetometer-free approach. To eliminate drift in the 3D relative IMU orientation $R_t^{S_1S_2}$, the author's idea was to update both the orientations $R_t^{GS_1}$ and $R_t^{GS_2}$ simultaneously from the adjacent IMUs. Here, $R^{S_1S_2}$ denotes the orientation of S_2 with respect to S_1. Similarly, $R_t^{GS_i}$ denotes the orientation of the i-th IMU with respect to the global coordinate frame G. This update is done by making use of the gyroscope and acceleration measurements, and exploiting knowledge from rigid body kinematics. The joint center, being the point of interest here, can only have one unique acceleration $a_{jc,t}^{S_i}$ in a common reference frame. This is illustrated in Figure 2-1(b) and can be expressed as

$$R_t^{GS_1} a_{jc,t}^{S_1} = R_t^{GS_2} a_{jc,t}^{S_2} + e_{\mathrm{link},t}. \tag{2-1}$$

Assuming that $e_{\text{link},t}$ is zero-mean Gaussian noise with covariance Σ_{link} denoted as $e_{\text{link},t} \sim \mathcal{N}(0, \Sigma_{\text{link}})$. The acceleration of the joint, expressed in both the sensor coordinate frames $a_{\text{jc},t}^{S_1}$ and $a_{\text{jc},t}^{S_2}$ are then approximated by evaluating the obtained accelerations $y_{a,t}^{S_i}$ at a distance $r_i^{S_i}$ from the joint center as follows [15][19][20][21],

$$a_{\text{jc},t}^{S_i} = y_{a,t}^{S_i} - \mathcal{C}_t^{S_i} r_i^{S_i}, \tag{2-2a}$$

$$\mathcal{C}_t^{S_i} = \left[y_{\omega,t}^{S_i}\times\right]^2 + \left[\dot{y}_{\omega,t}^{S_i}\times\right]. \tag{2-2b}$$

Here, $y_{\omega,t}^{S_i}$ denotes the angular velocity, $\dot{y}_{\omega,t}^{S_i}$ denotes the angular acceleration and $[x\times]$ describes the cross product matrix operator as given in [10]. These models were then implemented in an optimization approach or an Extended Kalman Filter (EKF). The desired relative sensor orientations were then obtained using

$$\hat{q}_t^{S_1S_2} = \left(\hat{q}_t^{GS_1}\right)^{\text{c}} \odot \hat{q}_t^{GS_2}, \tag{2-3}$$

where c and $\odot$ denote the quaternion conjugate and the quaternion multiplication operator respectively. Readers unfamiliar with quaternions and these operators can refer to [10].

The authors applied both an optimization and filtering approach to gait analysis on eleven subjects to evaluate the IMU-based joint kinematic estimation method against a 3D optical motion capture reference system. The mean RMSE knee joint angle values over all subjects were 2.14 [deg], 1.85 [deg] and 3.66 [deg] for the optimization-based smoothing approach. In the filtering-based implementation, these mean RMSE values were 3.08 [deg], 2.42 [deg], and 4.47 [deg]. From these RMSE values, it can be concluded that the optimization-based approach yielded more accurate results. However, the filtering method opens up for long term patient monitoring [15].

When the related work methods have been discussed, in Section 2-6, a summary will be given of each discussed method's challenges being addressed and its remaining limitations. For that reason, attention will now be turned to the method of Dorschky et al. (2019) which mainly inspired this thesis work. Their method did not only take into account this notion of modeled connected body segments but also included human anatomy and more model complexity. This allowed the authors to estimate not only the human kinematics like the method of Weygers et al. (2020) discussed here, but also opens the door to estimating human kinetics.

2-3 Existing work on sensor fusion and dynamical model based kinematics and kinetics estimation

The work presented by Dorschky et al. (2019) made use of a biomechanical model with more model complexity compared to the previously discussed method. For each segment, its mass, length, center of mass location, and the moments of inertia were incorporated. This extra information allowed the authors to fulfill their objective to estimate the kinematics as well as the kinetics during human gait. To that aim, they modeled a planar (2D) musculoskeletal model, as can be seen in Figure 2-2(a), conforming to the body of the participant. Virtual sensors were positioned on the segments of this model. These virtual sensors produced comparable gyroscope and accelerometer signals as would be measured by the real IMUs being

attached to the various body segments of the participant. One such virtual sensor attached to the shank is shown in Figure 2-2(b). Subsequently, they proposed an optimal control problem to find the model states x and control inputs u, such that these virtual sensors tracked experimentally logged raw IMU data gathered from each individual participant. From the dynamically consistent simulations obtained, human kinematic and kinetic quantities could be estimated [6].

The focus of [6] was aimed at human gait. As such, only the lower extremities were modeled together with the trunk segment using nine generalized coordinates q, also shown in Figure 2-2. The first two states contained the X and Y-position of the trunk denoted p_{Trunk}, with respect to the ground frame G. The other seven states were the various joint angles θ, between linked rigid body segments defined relative to each other. The time derivatives of q were defined as the generalized velocities v. The model itself was actuated by 16 muscles, where the state of each muscle was characterized by its contractile element L_{CE}, and its activation function α as defined in [22]. Besides, foot-contact interactions were also taken into account. The model's state vector x consisted of all these defined variables. Their method was based on formulating the system dynamics implicitly as a function of $x, \dot{x}$ and u as $f(x, \dot{x}, u) = 0$. Here, u denotes the control vector that drives the model consisting of the neural excitations of the muscles. Due to the system dynamics being twice differentiable with respect to $x, \dot{x}$, and u, allowed the authors to use a gradient-based optimal control method.

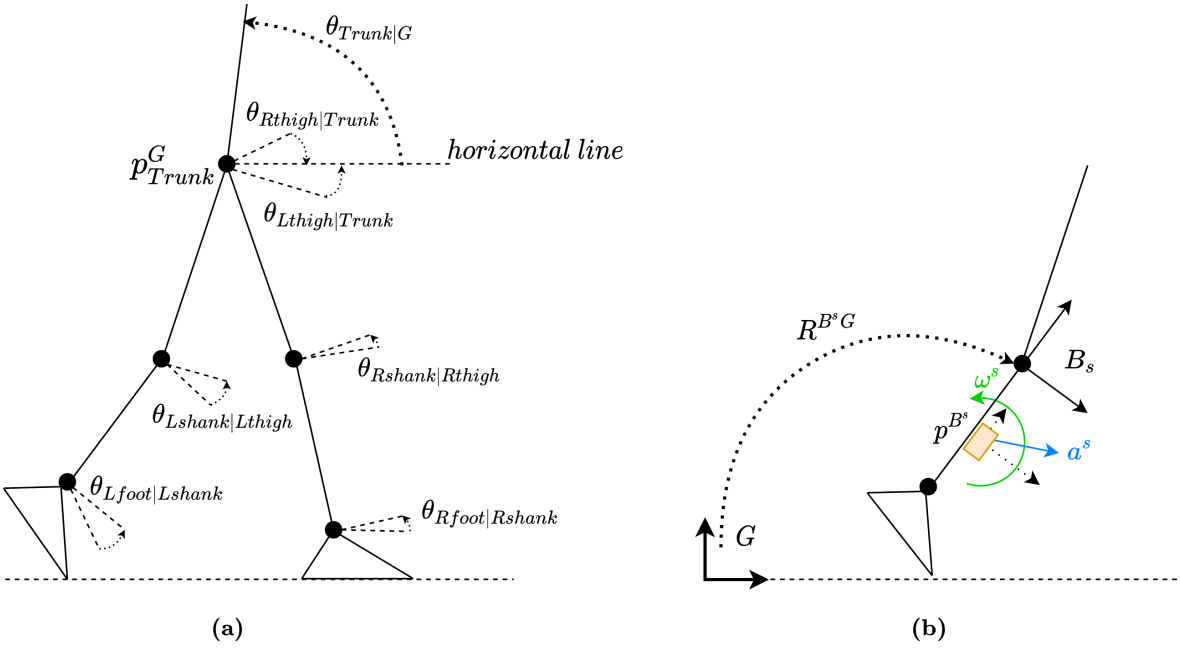


Figure 2-2: (a) The musculoskeletal model with its generalized coordinates, muscles, and rigid segments. (Muscle have been omitted for model clarity). (b) Placement of a virtual sensor at position p^{B^s} in the body-segment coordinate system B^s. The acceleration a^s and angular velocity ω^s are measured in B_s with respect to the global frame G using the transformation matrix R^{B^sG}. Both figures are adapted from [6].

It was assumed that the axes of the sensor were aligned with the body segment coordinate frame B^s and that the positions p^{B^s} were known in this B^s frame. The virtual sensor's gyroscope signal ω^s was equal to the angular velocity of its corresponding body segment with

respect to the global frame G. The virtual accelerations measured at a body segment a^s were estimated as

$$a^s = R^{B^s G}\left(a^G_{G|B^s} - g\right) + \begin{bmatrix} -(\omega^s)^2 & -\dot{\omega}^s \\ \dot{\omega}^s & -(\omega^s)^2 \end{bmatrix} p^{B^s}. \tag{2-4}$$

Here $a^G_{G|B_s}$ is the segment's origin acceleration relative to G, and $R^{B^s G}$ denotes the transformation matrix from the global frame G to the respective body-segment frame B^s. The gravity vector was set equal to $g = [0, -9.81]^\top$ [m/s^2]. To have these virtual sensors track experimentally obtained raw IMU measurements, an optimal control problem was formulated for which the cost function J was defined as follows

$$\begin{aligned} \underset{x(t),u(t)}{\text{minimize}}\ J(x(t),u(t)) = \frac{1}{|\mathcal{S}|N}\sum_{k=1}^{N}\sum_{s\in\mathcal{S}}\left(\left(\frac{a^s_{x,k}-\mu_{a^s_{x,k}}}{\sigma_{a_{x,k}}}\right)^2 + \left(\frac{a^s_{y,k}-\mu_{a^s_{y,k}}}{\sigma_{a^s_{y,k}}}\right)^2 + \right. \\ \left.\left(\frac{\omega^s_k-\mu_{\omega^s_k}}{\sigma_{\omega^s_k}}\right)^2\right) + \frac{W_{\text{effort}}}{16NV^2}\sum_{k=1}^{N}\sum_{m=1}^{16}u^2_{mk} + J_{\text{reg}}. \end{aligned} \tag{2-5}$$

Here $\mathcal{S}$ is the set of IMUs attached to the various body segments, N the number of collocation nodes, and μ denotes the mean for multiple strides of the measured sensor signals. The differences were normalized to the measured standard deviation σ of multiple strides. This guaranteed that noise and movement artifacts were not tracked. Moreover, W_{effort} was a weighting term found empirically and V was the model's speed. J_{reg} denotes the term proportional to the integral of the sum of squares of all state and control variables' time derivatives. The problem at hand was a large scale nonlinear optimization problem which was then solved to compute the trunk position p_{Trunk}, joint angles θ and control inputs u. Although this proposed method constitutes a necessary step towards more advanced human movement research, the limitation remains that a 2D model was used. This 2D model restricted all motions to a plane. Hence, the authors propose future research to investigate motions using 3D models for more advanced motion analysis.

Another method, proposed by Karatsidis et al. (2019) [17], made use of the commercially available software suite Xsens MVN Link, developed by Xsens Technologies B.V. In their approach, 17 IMUs were attached, using dedicated clothing, to the hands, forearms, upper arms, shoulders, feet, lower legs, upper legs, pelvis, sternum, and head. The Xsens MVN Link software was then used to obtain the orientations of the IMUs with respect to an earth-based coordinate frame and to compute the kinematics by constrained optimization. The exact sensor positions on the respective body segment were determined using the accompanied manufacturer guidelines described in the Xsens MVN manual. By formulating an optimization problem, they obtained the muscle reaction forces for the performed motion trajectory by minimizing the muscle activity which was subject to dynamic equilibrium constraints. For validation purposes, the authors validated their approach against an Optical Motion Capture (OMC) system. Eight infrared high-speed cameras were used to track the trajectories of the 53 attached reflective markers on the human body. Lastly, the ground reaction forces (GRFs) and joint moments were determined using inverse dynamics for walking motions. The results obtained from both Dorschky et al. (2019) and Karatsidis et al. (2019) are shown in Table 2-1.

The use of the Xsens MVN suite, however, limits the applicability of this procedure to these commercial systems. Moreover, it is still vague from their paper [17] what kind of algorithms

Table 2-1: The results of Dorschky et al. (2019) [6] and Karatsidis et al. (2019) [17]. The lower limb kinematics and kinetics in the sagittal-plane are compared between the IMC method and the OMC method. For each motion scenario, the root-mean-squared error (RMSE) mean and (standard deviation) in degrees, BodyWeightBodyHeight% and BodyWeight% respectively are shown. GRF in the table stands for Ground Reaction Force. Table is adapted from [6].

Quantity	Walking [6]		Running [6]		Walking [17]	
	RMSE		RMSE		RMSE	
Hip angle	8.2	(3.3)	8.7	(3.2)	5.7	(2.1)
Knee angle	5.5	(2.8)	5.3	(3.0)	4.4	(2.0)
Ankle angle	4.3	(1.5)	4.6	(1.7)	4.1	(1.3)
Hip moment	1.5	(0.4)	3.2	(1.0)	2.2	(0.6)
Knee moment	1.5	(0.4)	3.4	(1.2)	1.9	(0.5)
Ankle moment	1.6	(0.8)	3.2	(2.1)	1.6	(0.6)
Anterior-Posterior GRF	4.1	(1.2)	10.7	(3.9)	1.6	(0.6)
Vertical GRF	11.1	(3.4)	32.0	(7.9)	9.3	(3.0)

and methods are implemented to obtain these orientation estimates such that motions can be tracked. To that aim, for this thesis work, use will be made of an open-source and freely available modeling and simulation software package to circumvent this drawback.

2-4 OpenSim: A modeling and simulation environment

The discussed kinematic model for sensor fusion purposes developed by [15] lacks the human anatomy. This results in the fact that the system's dynamics and the type of joint connecting two rigid bodies cannot be taken into account. From that perspective, this thesis will aim to investigate the advantages of including more information about the system. Knowledge about the system such as the type of joint between two segments, the rigid body's mass and length, its center of mass location, and moments of inertia will be incorporated.

For a double pendulum, one can derive the equations of motion (EoM) by hand that govern the dynamics of this system using for example the Euler-Lagrange equations. However, for systems composed of multiple links, this becomes a tedious and error-prone task as one has to properly take care of for instance all the respective lengths, masses, and velocities. Hence, formulating and solving the EoM that describe the dynamics of a musculoskeletal system is troublesome and even more error-prone. Besides, especially in biomechanical systems, the joints between two rigid body systems are not comparable to the classical type of joints, like revolute or ball-and-socket joints, found in mechanical systems.

For that reason, the OpenSim software suite showed to be a promising modeling and simulation framework. Mainly, as it automatically generates a system with these governing EoM for a user-created model [23]. Using the built-in OpenSim solvers, a forward dynamics simulation is generated such that one can study these models with their motion. From this simulated model, more human motion-related metrics can then be obtained. Moreover, OpenSim was chosen as these human-like joints can be modeled and scaled to the dimensions of the participant's body. This is essential for the accurateness of in-field participant experiments. Another

important benefit of this OpenSim framework is that it is a free and open-source modeling platform. Next to that, the community of OpenSim users is growing, diverse, worldwide, and has surpassed 370,000 downloads at the time of writing. This shows that developing a new motion reconstruction algorithm based on OpenSim can, when tested and validated, be implemented and used by this large community of users. The OpenSim API is available via different programming languages such as `C++`, `Java`, `MATLAB` and `Python` or via the desktop application comprising a GUI and visualizer. Especially, accessing the OpenSim functionality via `MATLAB` or `Python` opens the door to researchers with less experience in coding, but having experience in scripting. This moreover extends the functionality compared to running OpenSim in the native desktop application.

Certainly, there are also other simulators available, for instance, Simscape Multibody from `MATLAB`. This environment yet lacks the ability to create accurate custom joints analogous to human joints and include muscles that deform when the body attached to the joint is extended. As the aim of this thesis is to develop an algorithm that generalizes well to both mechanical as well as biomechanical systems, this further motivated the choice for OpenSim. In Figure 2-3, two different systems modeled in the OpenSim environment are shown. The left biomechanical model depicts the scapulothoracic joint of [24], and the right mechanical model shows the KUKA LBR iiwa R800 7 DoF robot manipulator.

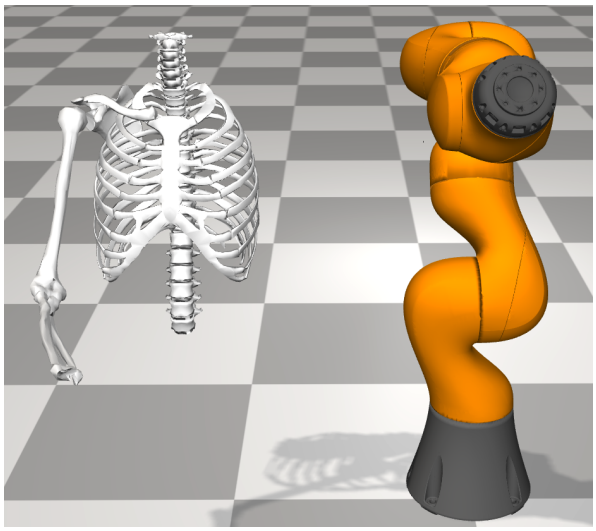


Figure 2-3: The biomechanical model of the scapulothoracic joint [24] (left), and the mechanical model of the KUKA LBR iiwa 7 R800 robot manipulator (right), in the modeling and simulation environment OpenSim.

As previously explained in Chapter 1, the aim is to validate the algorithm developed on the physical KUKA robot manipulator system. The corresponding OpenSim model of the KUKA is shown in Figure 2-3 on the right. The main reason for choosing this physical robot system was that this robot features quality joint encoders from which the ground truth joint angles can be obtained. The estimates of the novel algorithm can then be validated against these encoder values. With this validated generally applicable algorithm, future work could focus on reconstructing human motions. Particularly, using the human shoulder joint [24] modeled in OpenSim as shown in Figure 2-3 on the left. OpenSim then automatically determines the EoM governing this complex shoulder joint's motion.

Recently, a new method as proposed by the creators of OpenSim has been created which also allows to reconstruct human motions. This technique is based on tracking IMU orientations which also uses a (bio)mechanical model and will be described next.

2-5 Opensense: An OpenSim software tool for reconstructing motion

In the new 4.1 version of OpenSim, a workflow for analyzing movements obtained from IMU data is made available to the research community [25]. This method relies on the creation of virtual IMUs as orthogonal XYZ-frames on the segments of the user-created OpenSim model. The participant's motion is then reconstructed using an inverse kinematics method. This approach, at each time of the motion, computes the set of joint angles that minimizes the errors between the experimental IMU orientations and the orientations of the virtual model XYZ-sensor frames.

The user is required to input the orientations of an IMU. Estimating the orientation of a sensor can be accomplished using various techniques. For example, one can use either the Extended Kalman Filter algorithm with quaternions as states [10], Xsens' native orientation estimation algorithm provided with their MTw Awinda IMUs [11] or Madgwick's orientation estimation algorithm [26].

The OpenSense procedure will first start with the placement of the sensors on the desired body segments whose motion should be tracked. Calibration data must then be collected by having the participant standing in a known pose which should, as close as possible, match the default pose of the OpenSim model. OpenSense assumes pre-processed data, such as time syncing and data interpolation for missing entries. This pre-processing will be handled by Xsens' sensor system. However, when users work with different manufactured IMUs, they have to perform these pre-processing steps themselves. OpenSense then requires the orientations parametrized as rotation matrices as input. Upon import, a time-synced storage file for these logged orientations is created where the rotation matrices are converted into quaternions. By reading in the data, each IMU sensor is represented as an orthogonal XYZ coordinate frame in the OpenSim model [25].

Lastly, the calibration step uses the gathered calibration data and the loaded OpenSim model as input. For each constructed IMU frame, it finds its orientation with respect to the body segments of the OpenSim model. Note that, this calibration procedure assumes that the subject's pose obtained from the calibration data matches the default pose of the model. This then results in virtual XYZ-frames onto the segments of the biomechanical OpenSim model. These virtual XYZ-frames represent the corresponding experimental IMUs positioned on the participant's body.

The limitation of this OpenSense workflow lies in the fact that these orientation estimates, computed from the IMU measurements, are assumed to be 100% accurate. While in fact, these orientation estimates are always slightly off. Especially, when the sensor is in the presence of magnetic material, which disturbs the magnetometer measurements, leading to incorrect orientation estimations [27]. Therefore, it is assumed that including more information together with tracking raw inertial measurements, instead of orientations derived thereof, will increase the accuracy of reconstructions obtained.

2-6 Requirements for the motion reconstruction algorithm

With these current methods presented, various challenges addressed and current limitations can be recognized. Hence, prior to arguing why a new method needs to be invented, a clear overview of these elements should be taken in mind. Table 2-2 summarizes for each method these main challenges addressed and highlights its limitations.

Table 2-2: Relevant past work in the field of motion reconstruction.

Method	*Approach*	*Challenges addressed*	*Current limitations*
Weygers et al. [15]	Incorporating tightly coupled rigid body kinematics in the sensor fusion algorithm to compensate for drift and estimate 3D joint kinematics.	Relative sensor orientation drift is compensated without relying on a magnetometer.	Only kinematic variables can be estimated as no inertial properties are included, hence no kinetic metric estimations are possible.
		Joint angles between two body-attached IMUs can be estimated with an average accuracy of <2.56 [deg].	The type of the joint is not considered, all joints between two linked rigid bodies are assumed to be spherical joints.
		Accurate 3D joint estimations are achieved for long (>5 min) gait trials.	The ranges of motion for each rigid body are not taken into account.
Dorschky et al. [6]	Estimating gait kinematics and kinetics by tracking inertial sensor data with a musculoskeletal model by solving an optimal control problem.	Combination of raw inertial sensor data with musculoskeletal models to estimate kinematics and kinetics.	2D models limit the motions and analysis to the sagittal plane, affecting the ability of the model to track IMU data.
		Create virtual sensors to generate congruent artificial sensor data to obtain dynamically consistent motions.	The method can be applied to 3D models, however, this will result in longer computation times, currently 2D only.
		Incorporating accurate participant's segment inertial properties.	Still rather large joint angle RMSE values obtained ranging between 4-8 [deg].
Karatsidis et al. [17]	Perform musculoskeletal model-based inverse dynamics based on tracking orientations of 17 IMU sensors attached to the participant's body	Driving a musculoskeletal model using exclusively IMU orientation data to estimate 3D kinematics and kinetics.	Only tested for walking conditions, hence unsure how the method performs during more dynamic movements such as running.
		Scaling of the segments of the musculoskeletal model to the measured body dimensions of the participant.	The method is based on a commercial algorithm which limits the applicability to commercial settings.
		Ability to estimate joint moments and ground reaction forces.	The average joint angles RMSE values ranged between 4-7 [deg].
OpenSense	Minimize the errors between the experimental IMU orientations and the orientations of virtual IMU frames attached on the biomechanical model.	Constrain the orientation estimates of the experimental sensors to the segments of the underlying model.	Relies on manufacturers' orientation estimation algorithm which is not 100 % accurate when magnetically distorted.
		Allows to be applied to user-created biomechanical models incorporating the joint type between two rigid bodies.	No system dynamics and inertial properties incorporated, hence only estimations of kinematic metrics are possible.
		Plug-and-play method allowing one to reconstruct human outdoor motions.	Although OpenSim is worldwide used [23], OpenSense has not yet been mentioned in the literature.

With this table presented, it can be concluded that so far accurate 3D joint estimations are possible using [15]. Unfortunately, this method currently does not allow for kinetic estimations and does not take into account the type of the joint. Contrary, OpenSense does take into account the joint type and as such constrains the orientation estimates of the sensor to the underlying model. However, this approach too lacks the information to estimate kinetics. On the other hand, [6], does incorporate this information which allows for kinetic estimations, but their method is currently limited to 2D models. Moreover, their chosen optimization-based approach limits this work to motions that have been captured and does not allow for real-time

motion reconstruction. As such, this approach would not allow for continuously adding IMU data, while the algorithm is processing the past measured IMU data sequentially. This latter observation, however, can for instance be accomplished using a filtering-based approach. With these remaining challenges noted, the main requirements that this novel framework should fulfill are summarized below.

- ***Adaptability of the algorithm to nonlinear mechanical as well as biomechanical systems which might be composed of multiple links.***
  With this work, the aim of this algorithm is to eventually be applied to a range of (bio)mechanical systems. For example, those shown in Figure 2-3, offering scientists and researchers in various fields, a new method for (human) motion reconstruction. As the models governing the motions of these systems are non-linear, the new approach should be able to deal with these nonlinear models.

- ***Continuous reconstruction of 3D motions in a markerless setting using small and wearable inertial sensors.***
  The method of [6] limited their analysis to reconstructing 2D motions using an optimization-based approach. The novel method developed in this thesis should be able to reconstruct continuous motions taking place in a 3D space not sacrificing on reconstruction accurateness. As such aiming for comparable joint angle estimations in the range of 4-8 [deg] like [6], but rather striving to accuracy's mentioned by [15] of around 2.56 [deg]. It must be noted that the results of [6] and [15] cannot be compared directly as they are based on different data sets from different human subjects. The requirement of eventual real-time continuous motion reconstruction automatically translates to a filtering-based approach. As real-time motion reconstruction is dependent on computation power, it is merely something to keep in mind. The main requirement should be to prove the concept of reconstructing 3D motions.

- ***Able to deal with noise, assumed to be of Gaussian form, and bias which are affecting the IMU measurements.***
  Like all methods mentioned above, to make accurate motion metric related estimations, the algorithm needs to be able to cope with noise and bias corrupting the inertial measurements.

- ***Incorporating kinematic constraints as well as system dynamics.***
  To gain a deeper understanding of movement mechanics and its underlying causes, especially in a biomechanical setting, including the assessment of joint angles and joint moments, would be beneficial. Clinicians require both accurate models of the subject's body and reliable measurements of motion capture methods to examine healthy and pathological movements [24]. To that aim, for this comprehensive biomechanical analysis, this algorithm needs to incorporate additional information. Including these kinematic constraints and the system's dynamics allows to make joint angle and joint moment estimations. Hence, also the masses, lengths, center of mass locations, and moments of inertia of each rigid body together with the type of the joint connecting two bodies should be incorporated.

With these requirements stated, Chapter 3, will propose the novel motion reconstruction algorithm. This algorithm aims to eliminate the limitations of the related work mentioned in this chapter and meet the requirements specified above.

Chapter 3

A novel algorithm for IMU-based motion reconstruction using dynamical models from OpenSim

This chapter introduces the Iterated Extended Kalman Filter - OpenSim (IEKF-OS) motion reconstruction algorithm. This algorithm includes noisy IMU measurement data, the system's dynamical model generated from OpenSim, and the Iterated Extended Kalman Filter (IEKF). First, in Section 3-1, the tracking control problem will be formulated to give a detailed overview of the approach taken to reconstruct the motion of the system at hand. Then, as for this technique two kinds of IMUs are required, being experimental and virtual IMUs, in Section 3-2 and Section 3-3, the measurement models of both these IMUs will be discussed. Moreover, it is also shown how the components of the IMUs can be calibrated. Having defined these measurement models, first, in Section 3-4 the augmented state vector will be presented. Then, in Section 3-5, the motion models will be discussed that govern the evolution of the state variables describing the system. Finally, in Section 3-6, the IEKF-OS algorithm will be presented which incorporates all this information to try and achieve the reconstruction of the original motion.

3-1 Formulation of the tracking control problem

To accurately reconstruct the motion of a system in OpenSim, it is beneficial to have a clear picture of this problem in mind. For that reason, a schematic overview of the motion tracking control problem is given in Figure 3-1. From this schematic, it can be seen that the tracking formulation is build up of three blocks. The first block, shown in green, illustrates the angular velocity and linear acceleration measurements coming from the experimental IMUs. These are the measurements that are obtained from body attached IMUs. For this thesis, the Xsens MTw Awinda IMUs are chosen. One advantage is that Xsens' Awinda protocol ensures time synchronization for up to 20 MTw's across the wireless network [11]. This makes it possible

to effectively fuse data from several IMUs. On the other side, the virtual measurements are depicted in the red block. These measurements should correspond to the same signals as measured by the experimental IMUs, hence being the angular velocity and linear acceleration. These virtual measurements can be predicted in the modeling and simulation environment OpenSim. To each of the rigid bodies of a user-generated model, an orthogonal XYZ-frame can be attached. From these frames, their virtual angular velocities and linear accelerations can be computed. The idea for this thesis is then to minimize the error between these experimental and virtual measurements such that a congruent virtually reconstructed motion in OpenSim is obtained. Lastly, the block in blue that can be recognized is the algorithm that tries to accomplish this error minimization objective, the Iterated Extended Kalman Filter - OpenSim (IEKF-OS).

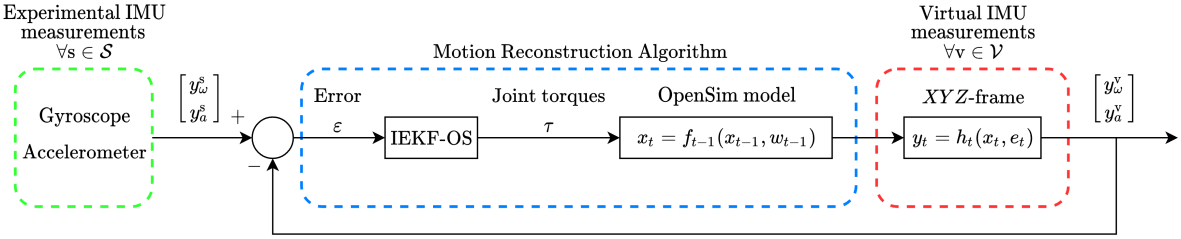


Figure 3-1: Schematic overview of the tracking control problem. The experimentally obtained IMU measurements are the reference tracking signals that the virtual IMU measurements have to match to obtain a congruent movement reconstruction in OpenSim.

Having defined these three blocks, the following sections will dive deeper into the underlying principles that make up these three blocks. First, the experimental IMUs with their measurement models will be discussed in the next section. Subsequently, the second block with the virtual IMUs will be detailed after which the IEKF-OS algorithm will be presented.

3-2 Experimental IMUs

In this thesis, a distinction is made between two types of IMUs. The first type of IMUs, defined as *experimental IMUs*, denote the sensors which will be attached to the human body segments or robotic links. The second type of IMUs, described by the term *virtual IMUs*, will characterize the sensors added to the (bio)mechanical model of the system at hand. This model can either represent a human or a robotic system. The virtual IMUs will be modeled as orthogonal XYZ-frames from which quantities of interest can be obtained. This section will begin with a brief outline of the coordinate frames used for both these types of IMUs. Moreover, note that the experimental IMUs have different measurement models than the virtual IMUs. This because the former are physical sensors that are affected by noise and bias corrupting the measurements made. For that reason, in this section, first, the measurement models of the experimental IMUs will be discussed. After that, it will be shown how these sensors can be calibrated. Subsequently, the measurement models of the virtual IMUs will be presented.

3-2-1 Coordinate frames

Throughout this thesis, use will be made of five coordinate frames which are defined as

- Sensor frame s_i: The frame in which the measurements of both the experimental gyroscope and experimental accelerometer from the *i-th* IMU are obtained. The origin of this sensor frame is located in the center of the accelerometer triad.
- Navigation frame n: Is the global coordinate reference frame for the experimental IMUs. It is defined stationary with respect to the earth.
- Body frame b_i: The frame of the rigid body segment which is located in the center of the joint.
- Virtual sensor frame v_i: The virtual frame in which the measurements of the i-th virtual IMU will be expressed. This virtual sensor frame will be defined with respect to the body frame b_i of the corresponding link.
- The ground frame G: This is an arbitrarily defined frame in the modeling and simulation environment OpenSim. As movements will be reconstructed in this digital environment, this frame is only aligned with gravity. All the frames attached to the rigid bodies of an OpenSim model are expressed with respect to the ground frame G unless otherwise indicated.

The movement reconstruction algorithm depends on the measurements of multiple sensors being attached to various robotic links or human body segments. However, for the sake of notation, the *i-th* sensor frames are dropped. Moreover, it is assumed that the IMUs are rigidly attached to their respective segments, which for human body segments is generally not true. Considering that the validation part of this movement reconstruction algorithm will be performed on a robotic arm, soft tissue artifacts (STAs) can be neglected.

3-2-2 Experimental IMU measurement models

Inertial Measurement Units (IMUs) are tri-axial sensors that, due to the advances in Micro-Electro-Mechanical System (MEMS) technology, are nowadays wearable, inexpensive, small, and easy in use [8]. These sensors typically consist of a gyroscope and an accelerometer which provide measurements of the angular velocity and the linear acceleration respectively. These quantities are corrupted by bias and noise. For the application of movement reconstruction, it is key to accurately characterize the measurement models of both these components.

Measurement model of the experimental gyroscope

The first component of the IMU, the gyroscope, measures the angular velocity, ω_t^{s}, at each time instant t. The measurements of this sensor are affected by a bias term $b_{\omega,t}^{\mathrm{s}}$ and a noise term $e_{\omega,t}^{\mathrm{s}}$ which leads to the following measurement model

$$y_{\omega,t}^{\mathrm{s}} = \omega_t^{\mathrm{s}} + b_{\omega,t}^{\mathrm{s}} + e_{\omega,t}^{\mathrm{s}}. \tag{3-1}$$

The slowly time-varying bias term, $b_{\omega,t}$, can be dealt with in various ways. When the sensor is used for short experiments, a calibration technique, as will be described in Section 3-2-3, prior to the experiment can be performed. Another way is to treat this slowly time-varying bias term by augmenting the state vector such that it can be estimated [10]. The noise term $e^{\mathrm{s}}_{\omega,t}$ on the other hand is assumed to be a Gaussian and modeled as $e^{\mathrm{s}}_{\omega,t} \sim \mathcal{N}(0, \Sigma_\omega)$. In Section 3-2-3, it will be shown how the covariance matrix Σ_ω can be estimated from gyroscope measurements.

Measurement model of the experimental accelerometer

The other IMU component used for this thesis is the accelerometer. Accelerometers measure the specific force, denoted as f^{s}_t, at each time instant t in the sensor frame s. The Coriolis acceleration related to the earth's rotation is small compared to the magnitude of the acceleration measurements. Next to that, it is assumed that the centrifugal acceleration is absorbed in the local gravity vector [10]. With these aspects taken into account, the simplified model for the specific force in sensor frame s can be given as

$$f^{\mathrm{s}} = R^{\mathrm{sn}}(a^{\mathrm{n}} - g^{\mathrm{n}}). \tag{3-2}$$

Here, R^{sn} denotes the rotation matrix mapping quantities expressed in navigation frame n, to quantities expressed in the sensor frame s.

The output of this sensor consists of this specific force, but is moreover corrupted by a bias term, $b^{\mathrm{s}}_{a,t}$, and by a noise term, $e^{\mathrm{s}}_{a,t}$. This leads to the following measurement model for the accelerometer

$$y^{\mathrm{s}}_{a,t} = R^{\mathrm{sn}}(a^{\mathrm{n}}_t - g^{\mathrm{n}}) + b^{\mathrm{s}}_{a,t} + e^{\mathrm{s}}_{a,t}. \tag{3-3}$$

The accelerometer noise term is assumed to be Gaussian and modeled as $e^{\mathrm{s}}_{a,t} \sim \mathcal{N}(0, \Sigma_a)$.

3-2-3 Calibration of experimental IMUs

From (3-1) and (3-3), it is evident that the outputs of the IMU are corrupted by bias and noise. Moreover, the outputs can be affected by misalignment errors of the triad-axes and scaling errors [28]. Therefore, significant accuracy can only be obtained when the IMU is properly calibrated to remove sensor biases and systematic errors. With calibration, the process is meant of comparing the instrument's output with known reference information and determining the coefficients that force the output to correspond with this reference information over a range of output values [29]. Commonly, some of the sensor errors change over time, and next to that, MEMS IMUs are only approximately calibrated by its producer [30]. Hence, to obtain high accuracy measurements, these sensors have to be re-calibrated prior to attaching the IMUs to the robotic arm or human body segments. To that aim, first, it will be outlined how the gyroscope can be calibrated after which the procedure to calibrate the accelerometer will be given.

Calibration of the experimental gyroscope

The calibration procedure of the gyroscope is rather straightforward. First, one needs to observe that when a gyroscope is placed on a flat surface, then independent of its orientation,

the measured rotational velocity should be zero mean. For this to be true, it is assumed that the effect of the earth's rotational velocity, compared to the sensor's noise, is too weak. Therefore, this effect can be assumed to be approximately zero, hence neglected [28]. Looking at the previously defined measurement model (3-1), one can observe that the measurements of the gyroscope are a summation of the actual angular velocity ω_t^{s}, a bias term $b_{\omega,t}^{\mathrm{s}}$ and a noise $e_{\omega,t}^{\mathrm{s}}$. Hence, the gyroscope measurements are affected by a slowly time-varying bias $b_{\omega,t}^{\mathrm{s}}$ and a zero-mean white noise denoted as $e_{\omega,t}^{\mathrm{s}} \sim \mathcal{N}(0, \Sigma_\omega)$ [10]. For a calibrated sensor, however, the three axes can be considered to yield independent measurements from one another. Therefore, the covariance matrix Σ_ω can be modeled as a diagonal matrix as

$$\Sigma_\omega = \begin{pmatrix} \sigma_{\omega,x}^2 & 0 & 0 \\ 0 & \sigma_{\omega,y}^2 & 0 \\ 0 & 0 & \sigma_{\omega,z}^2 \end{pmatrix}. \tag{3-4}$$

To calibrate the gyroscope, the simple procedure as described in [28] is taken. Here the idea is to determine the gyroscope bias by keeping it in a completely stationary state for a period of time. Then, again ignoring the weak effect of the rotation of the earth, for an uncalibrated gyroscope, this sensor's output for this stationary state should be equal to the bias term $b_{\omega,t}^{\mathrm{s}}$. Hence, the gyroscope bias can be computed as the mean of the data during this stationary period. The covariance matrix Σ_ω can then be found as the standard deviation from this mean. In Appendix F, a practical workflow for future users is presented to perform IMU calibrations.

Calibration of the experimental accelerometer

Normally, for orientation estimation, the accelerometer calibration is often neglected. Mainly, as for orientation estimation, the approximation is made that the accelerometer measurements are typically dominated by the gravity vector [10]. For this thesis, however, it is important to properly calibrate the accelerometer as the idea is to track motion-related accelerations. Here only incorporating the gravity acceleration and not the additional accelerations would lead to larger errors, resulting in inaccurate tracking performance. Therefore, a more elaborate calibration of the accelerometer is followed, which is based on the methods as described in [31], [32] and [33]. The noise affecting the accelerometer was assumed to be Gaussian and therefore, can be modeled as $e_{a,t}^{\mathrm{s}} \sim \mathcal{N}(0, \Sigma_a)$. For a properly calibrated sensor, the covariance matrix denoted as Σ_a can be modeled as a diagonal matrix of the following form

$$\Sigma_{\mathrm{a}} = \begin{pmatrix} \sigma_{a,x}^2 & 0 & 0 \\ 0 & \sigma_{a,y}^2 & 0 \\ 0 & 0 & \sigma_{a,z}^2 \end{pmatrix}. \tag{3-5}$$

Contrary to the accelerometer measurement model as depicted in (3-3), the uncalibrated measurements, $y_{a,t}$, are modeled as

$$y_{a,t} = DR_t^{\mathrm{sn}}(a_t^{\mathrm{n}} - g^{\mathrm{n}}) + b_{a,t}^{\mathrm{s}} + e_{a,t}^{\mathrm{s}}. \tag{3-6}$$

Here, sources of error are modeled by the matrix $D \in \mathbb{R}^{3\times 3}$. This D term can be seen as a product of matrices that model errors such as non-orthogonal sensors axes, cross-axis

interference, gains, and inter-sensor misalignments. The number of unknown parameters that construct the elements of this D matrix can be reduced when some of these errors are not taken into account. The matrix D can be decomposed into a matrix of scaling factors K, and a matrix T, which model these above-mentioned misalignments [34]. The accelerometer triad measures the linear acceleration in its sensor frame s. When the non-orthogonal sensitivity axes of this accelerometer triad vary by a small angle from the ideal orthogonal set of axes, then the specific force f_t^{N} in the non-orthogonal axes N can be transformed into the orthogonal axes s as shown in [35] as

$$f_t^{\mathrm{s}} = T^{\mathrm{sN}} f_t^{\mathrm{N}}, \tag{3-7}$$

where the transformation matrix T^{sN} is of the form

$$T^{\mathrm{sN}} = \begin{pmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ \alpha_{xz} & 1 & -\alpha_{zx} \\ -\alpha_{xy} & \alpha_{yx} & 1 \end{pmatrix}. \tag{3-8}$$

Here, α_{ij} denotes the rotation of the *i-th* accelerometer axis around the *j-th* orthogonal axis. When moreover, it is assumed that the X-axis of the accelerometer sensor coincides with the ideal orthogonal X-axis and that the orthogonal Y-axis lies in the X, Y-plane spanned by the accelerometer sensor axes, then the angles α_{xy}, α_{xz} and α_{yx} become zero. This reduces T^{sN} to an upper triangular matrix as

$$T^{\mathrm{sN}} = \begin{pmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{pmatrix}. \tag{3-9}$$

Lastly, the accelerometer scale factor error is also included in D. This error is expressed in the scale factor matrix K as

$$K = \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{pmatrix}. \tag{3-10}$$

With T^{sN} and K defined, D can then be written as

$$D = \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{pmatrix} \begin{pmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{pmatrix}. \tag{3-11}$$

Uncalibrated accelerometer measurements in a 3D space will form an ellipsoid centered around the sensor bias $b_{a,t}^{\mathrm{s}} \in \mathbb{R}^{3\times 1}$, while calibrated accelerometer measurements will form a sphere of radius $||g||_2$ centered around the origin [33]. To map the data to a sphere, the ellipsoid fitting method of [33] is taken. This method is based on the principle that the local gravity field is constant. To perform the calibration, the IMU needs to be slowly rotated in as many orientations as possible. These orientations, however, do not need to be known. By slowly rotating the sensor, it is assumed that the sensor is moving with a constant linear velocity while being in a constant gravitational field with gravitational acceleration g. To achieve several IMU orientations, in which the sensor is stationary, and rotating the IMU between orientations with approximately a constant velocity, an orientation prism was created as shown in Figure 3-2.

Figure 3-2: An orientation prism with 18 faces in which the Xsens MTw can be placed to obtain stationary acceleration data for these 18 different orientations.

Hence, it is assumed that the sensor does not experience an external acceleration a_t^{n} which reduces (3-6) to

$$y_{a,t} = -DR_t^{\mathrm{sn}} g^{\mathrm{n}} + b_{a,t}^{\mathrm{s}} + e_{a,t}^{\mathrm{s}}. \tag{3-12}$$

Thus the desired error-free accelerometer output is equal to $-R_t^{\mathrm{sn}} g^{\mathrm{n}}$. When the terms D and $b_{a,t}^{\mathrm{s}}$ in (3-12) are known, then the calibrated accelerometer measurements can be computed as

$$y_{a,t}^{cal} = D^{-1}\left(y_{a,t} - b_{a,t}^{\mathrm{s}}\right). \tag{3-13}$$

Assuming thus that the system is in a constant gravitational field and has a constant linear velocity during the calibration procedure. First, the norm of the local gravitational field will be scaled such that this is equal to 1. The idea to find this D matrix and bias $b_{a,t}^{\mathrm{s}}$ can then be posed as

$$\begin{aligned} &||-R_t^{\mathrm{sn}} g^{\mathrm{n}}||_2^2 - 1 = 0, \\ &||D^{-1}(y_{a,t} - b_{a,t}^{\mathrm{s}} - e_{a,t}^{\mathrm{s}})||_2^2 - 1 = 0. \end{aligned} \tag{3-14}$$

Observing that the accelerometer measurements are corrupted by noise, this equality does not hold exactly [32]. The ellipsoid fitting problem can for that reason be reformulated as described in [31] as

$$y_{\mathrm{a},t}^{\top} A y_{\mathrm{a},t} + \beta^{\top} y_{\mathrm{a},t} + \gamma \approx 0, \tag{3-15}$$

with

$$A \triangleq D^{-\top} D^{-1}, \quad \beta^{\top} \triangleq -2 b_a^{\mathrm{s}\top} A, \quad \gamma \triangleq b_a^{\mathrm{s}\top} A b_a^{\mathrm{s}} - 1. \tag{3-16}$$

When the matrix A is assumed to be positive definite, this problem can be viewed as the definition of an ellipsoid with the parameters A, β and γ [36]. The fitting problem given in (3-15) can be rewritten as a linear relation of the parameters M and ξ as shown in [31] as

$$M\xi \approx 0, \tag{3-17}$$

where

$$M = \begin{pmatrix} y_{a,1} \otimes y_{a,1} & y_{a,1} & 1 \\ y_{a,2} \otimes y_{a,2} & y_{a,2} & 1 \\ \vdots & \vdots & \vdots \\ y_{a,N} \otimes y_{a,N} & y_{a,N} & 1 \end{pmatrix}, \quad \xi = \begin{pmatrix} \mathrm{vec}A \\ \beta \\ \gamma \end{pmatrix}, \tag{3-18}$$

where vec denotes the vectorization operator and $\otimes$ denotes the Kronecker product. To circumvent the solution of obtaining the trivial solution $\xi = 0$ and guaranteeing the positive definiteness of the matrix A, this problem is instead solved as a semidefinite program [32]

$$\begin{aligned} \min_{A,\beta,\gamma} \ & \tfrac{1}{2} \left\| M \begin{pmatrix} \operatorname{vec} A \\ \beta \\ \gamma \end{pmatrix} \right\|_2^2, \\ \text{s.t.} \ & \operatorname{Tr} A = 1, \quad A \in S_{++}^{3\times 3}. \end{aligned} \tag{3-19}$$

Here the trace of the matrix A is constrained such to avoid the trivial solution $\xi = 0$ and $S_{++}^{3\times 3}$ denotes the set of 3×3 positive definite symmetric matrices. Note that a convex optimization problem is obtained which has a globally optimal solution. To solve (3-19), CVX was used which is a package for specifying and solving convex programs [37][38]. From the computed $\widehat{A}$, $\widehat{\beta}$ and $\widehat{\gamma}$ terms, estimations for the calibration matrix D and the bias term b_a^s can be made using (3-16) as

$$\begin{aligned} 1 &= b_a^{s\top} D^{-\top} D^{-1} b_a^s - \gamma, \\ &= \frac{1}{4}\beta^\top A^{-1}\beta - \gamma, \\ &= \alpha\left(\frac{1}{4}\beta^\top A^{-1}\beta - \gamma\right), \end{aligned} \tag{3-20}$$

leading to the following relation for the scaling factor α being

$$\alpha = \left(\frac{1}{4}\widehat{\beta}^\top \widehat{A}^{-1}\widehat{\beta} - \widehat{\gamma}\right)^{-1}. \tag{3-21}$$

Using this scaling factor, the estimates for both the calibration matrix D and the bias term b_a^s can be found using (3-16) as

$$\begin{aligned} \widehat{D}^{-\top}\widehat{D}^{-1} &= \alpha\widehat{A}, \\ b_a^s &= -\frac{1}{2}\widehat{A}^{-1}\widehat{\beta}. \end{aligned} \tag{3-22}$$

From 3-22 the estimate for $\widehat{D}$ can not uniquely be computed. For that reason, the initial estimate is determined using a Cholesky decomposition which yields a lower triangular matrix denoted by $\tilde{D}$ [32]. Moreover, as the norm of the local gravitational field was scaled to 1 for the ellipsoid fitting problem, the bias term needs to compensated for this normalization effect. Hence, the estimate of the bias is computed as

$$b_a^s = b_a^s \mathrm{N}_{LGF}, \tag{3-23}$$

where N_{LGF} is the estimate obtained of the local gravitational field norm in the position where the calibration procedure was performed. The calibrated data fitted to a sphere can be seen in Figure 3-3.

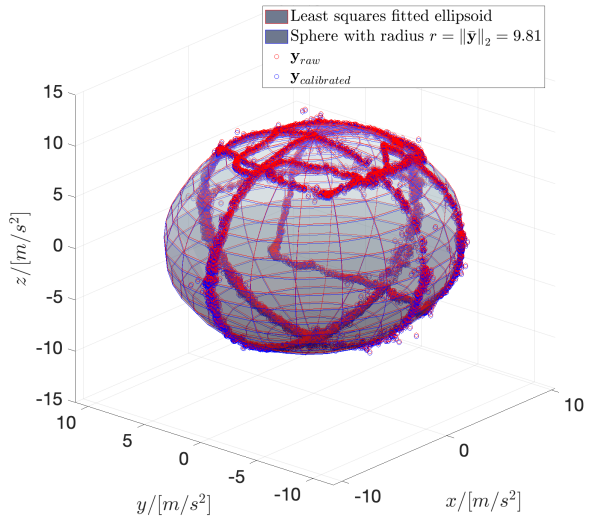


Figure 3-3: The results obtained from the ellipsoid fitting method. The raw accelerometer data is shown in red dots whereas the calibrated accelerometer data is shown in blue dots. Ideally, all data should lie on the blue plotted sphere with a radius of g. The least-squares fitted ellipsoid is shown in red.

3-3 Virtual IMUs

As it has now been outlined how the experimental gyroscopes and accelerometers can be calibrated, the focus is now laid on the virtual IMUs. Contrary to the experimental IMUs, these virtual IMUs do not have to be calibrated. Virtual IMUs are not affected by bias and noise and correspondingly thus have different measurement models. Once the model for the system is created in OpenSim, virtual sensors can be attached to its various segments. The position of a virtual sensor can be arbitrarily defined and is placed with respect to its parent frame, the body frame b. These frames are orthogonal XYZ-frames from which quantities of interest such as position, angular velocity and angular acceleration, and linear velocity and linear acceleration can be obtained. A visualization of a virtual IMU is shown in Figure 3-4. The virtual IMU measurement models are discussed in Section 3-3-1. These measurement models yield both the desired sensor dependent angular velocity and sensor dependent linear acceleration.

3-3-1 Virtual IMU measurements

The idea behind reconstructing the original motion relies on minimizing the error between the simulated virtual IMU measurements and the experimentally obtained IMU measurements. For that reason, it is key that the same signals are subtracted from each other in the same local sensor frame. Hence, the output of the measurement model, $h_t(x_t)$, should produce the angular velocity and the linear acceleration for each attached virtual IMU. Moreover, as these virtual IMUs, contrary to the physical experimental IMUs, are simulated frames, their outputs are not corrupted by bias and noise. As such, this yields the following measurement

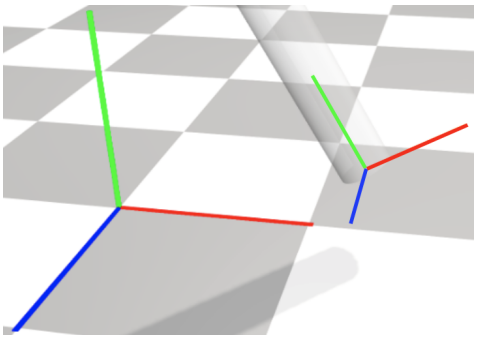


Figure 3-4: A virtual IMU modeled as a XYZ-frame attached to a body segment in the modeling and simulation software environment OpenSim. The OpenSim ground frame G is shown to the left of the body.

model

$$h_t\left(x_t\right) = \begin{pmatrix} \omega^{\mathrm{v}} \\ a^{\mathrm{v}} \end{pmatrix} \quad \forall \mathrm{v} \in \mathcal{V}, \tag{3-24}$$

where v denotes the virtual sensor in the set of all virtual sensors $\mathcal{V}$. Rather then formulating the explicit formulas to obtain the values for these outputs as in [6], for this thesis, they will be obtained from OpenSim. The user-generated models for both the robotic arm and the human scapulothoracic joint consist of several body segments. The connection of these segments in OpenSim is modeled by a so-called "*mobilizer*". This mobilizer connects a body to its unique parent body. Instead of restricting degrees of freedom (DoF) as constraints do, the concept of the mobilizer relies on initially allowing zero DoFs to the body. From there, the mobilizer technique grants the permitted motion of the body's frame relative to the parent's body frame and provides a parametrization of that motion [39]. This permitted mobility is characterized in terms of $n_q \geq n$ scalar generalized coordinates q and n terms of generalized speeds denoted by u. By this definition, the state vector that will be described in Section 3-4, is based on this concept of generalized coordinates. The permitted spatial motion of these body segments, which includes both rotations and translations, are related to the internal coordinate formulation q, via a hinge matrix H. The multibody physics API of OpenSim, called Simbody, uses this spatial notion which combines the rotational and translational components in a single object as was first described by [40]. Here, the first sub-vector of this object is always the rotational component of the motion whereas the second sub-vector involves the translational movement. This yields the spatial velocity V and spatial acceleration A objects as

$$V \triangleq \begin{pmatrix} V_\omega \\ V_v \end{pmatrix}, \quad A \triangleq \begin{pmatrix} A_\beta \\ A_a \end{pmatrix}, \tag{3-25}$$

where V_ω denotes the angular velocity and V_v corresponds to the linear velocity. Moreover, A_β is the angular acceleration, and A_a depicts the linear acceleration. Simbody determines all these variables with respect to the OpenSim ground frame G unless otherwise indicated. Therefore, the default symbols in (3-25) depict

$$V = V^{GB} \triangleq \begin{pmatrix} V_\omega^{GB} \\ V_v^{GB} \end{pmatrix}, \quad A = A^{GB} \triangleq \begin{pmatrix} A_\beta^{GB} \\ A_a^{GB} \end{pmatrix}. \tag{3-26}$$

The transform in Simbody is denoted as X^{GB} which implies frame B measured from and expressed in frame G [39]. Transforms are used to specify the configuration of one frame with respect to the other. Transforms combine a position as a translation, and a rotation. As such being composed of the rotation matrix R^{GB} which expresses the orientation of frame B in the frame G, and the position vector p^{GB} which is the vector from the origin of the G frame to the origin of the B frame [39]. For example, with the rotation matrix R^{GB}, a vector v^B expressed in the B frame, can be re-expressed as v^G in the ground frame G as $v^G = R^{GB}v^B$. As such, the spatial position is defined as

$$X^{GB} = \left(R^{GB} \mid p^{GB}\right). \tag{3-27}$$

The spatial position is thus a combination of the rotation, described using the rotation matrix R^{GB}, immediately followed by a translation vector denoted as p^{GB}. To get the desired signals, the virtual sensor dependent angular velocity ω^{v} and the virtual sensor dependent linear acceleration a^{v}, first note that the body's relative permitted motion is described using a pair of coordinate frames which correspond to the so-called *MobilizedBody* B. The term *MobilizedBody* refers to the combination of a body segment with its unique mobilizer. The first frame in this pair is the "*fixed*" frame F which is attached to the parent's body P with a constant transform X^{PF} as can be seen in Figure 3-5. The second frame, which is the unique mobilizer "*moving*" frame M, is attached to the body B with the constant transform X^{BM}.

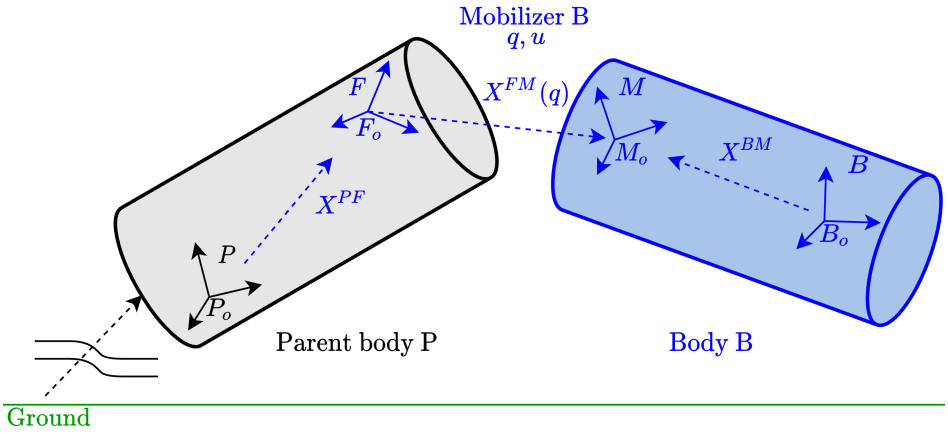


Figure 3-5: The MobilizedBody B, shown in blue, with respect to its parent body B, shown in grey. Every frame and transform that is associated with the MobilizedBody B is also shown in blue and the origins of these frames are denoted with O. Figure is adapted from [39].

With these definitions, the equations that define these generalized coordinates q and generalized speeds u using the hinge matrix H can be given as

$$X^{FM}(q) \triangleq \left(R^{FM}(q) \mid p^{FM}(q)\right), \tag{3-28a}$$

$$V^{FM}(q,u) \triangleq \begin{pmatrix} V_\omega^{FM} \\ V_v^{FM} \end{pmatrix} = \begin{pmatrix} \omega^{FM}(q,u) \\ v^{FM}(q,u) \end{pmatrix} = H^{FM}\left(X^{FM}(q)\right)u, \tag{3-28b}$$

$$A^{FM}(q,u,\dot{u}) \triangleq \dot{V}^{FM} = \begin{pmatrix} \beta^{FM}(q,u,\dot{u}) \\ a^{FM}(q,u,\dot{u}) \end{pmatrix} = H^{FM}\dot{u} + \dot{H}^{FM}u, \tag{3-28c}$$

$$\dot{q} = N(q)u. \tag{3-28d}$$

Note that in (3-28c), the time derivative of the generalized speed is taken, which results in the generalized acceleration $\dot{u}$. Moreover, the kinematic coupling matrix N in (3-28d) governs the evolution of the generalized coordinates q with the generalized velocities u. For most joints, the kinematic coupling matrix N will be equal to the identity matrix $\mathcal{I}$, e.g., like in hinge joints. In such joints, the spatial angular velocity is equal to the derivatives of the Euler angles or derivatives of the quaternions. This, of course, depending on the chosen rotation parametrization.

The virtual sensor dependent measurement models yielding both the angular velocity and linear acceleration signals, obtained from Simbody, can be given as

$$h_t(x_t) = \begin{pmatrix} \omega^{\mathrm{v}} \\ a^{\mathrm{v}} \end{pmatrix} = \begin{pmatrix} \omega^{FM}(q,u) \\ a^{FM}(q,u,\dot{u}) \end{pmatrix} = \begin{pmatrix} V_{\omega}^{FM} \\ A_{a}^{FM} \end{pmatrix} = \begin{pmatrix} H_{\omega}^{FM} u \\ H_{v}^{FM}\dot{u} + \dot{H}_{v}^{FM} u \end{pmatrix} \forall \mathrm{v} \in \mathcal{V}. \tag{3-29}$$

Where H_{ω}^{FM} and H_{v}^{FM} are the upper and lower $3 \times n$ partitions of the hinge matrix H and n denotes the number of mobilities granted by the mobilizer for body B with respect to its parent's body P.

3-3-2 Expressing the virtual IMU measurements in local virtual IMU frame

The angular velocity and the linear acceleration of an attached virtual frame to a body can be obtained using OpenSim functions. These functions yield the angular velocity and linear acceleration of the frame, measured with respect to and expressed in the ground frame. Note, however, that the measurements logged by the experimental Xsens IMUs are not expressed in the ground frame, but rather take place in the sensor frame, located in the center of the accelerometer triad. As the idea of this thesis is to achieve the movement reconstruction by means of minimizing the discrepancy between the experimental sensor measurements and the virtual sensor measurements, it is key that both measurements are expressed in the same frame. Therefore, the virtual measured angular velocity and linear acceleration need to be re-expressed in their local frame. Next to that, the gravity-induced acceleration needs to be incorporated, which will be outlined in the next section.

3-3-3 Incorporating the gravity-induced acceleration

Contrary to the real experimental IMUs, the virtual IMUs which are modeled as orthogonal XYZ-frames in OpenSim, are just reference frames. The acceleration of this OpenSim frame, similar to a body in OpenSim, is the resultant of forces which include reaction forces and gravity. This means, that when the body and thus the frame is at rest, its net force and acceleration are zero. Logically, experimental IMUs are not reference frames but are actual sensors, which register the reaction force opposing the weight of the test mass of the IMU. Thus when stationary, the output of the experimental IMU will read $[a_x, a_y, a_z] = [0, g, 0]$ with the Y-axis in this example pointing towards the sky. It is assumed that g is equal to 9.81 [m/s^2] from now on. Thus to obtain similar signals and to model a virtual IMU sensor, the acceleration induced by the gravity force needs to be accounted for in the virtual IMU measurements in OpenSim. In OpenSim, the Y-axis of the ground frame G, the world frame in OpenSim, is pointing out of the plane towards the sky. The gravity vector in this OpenSim ground frame G is equal to $g^G = [g_x, g_y, g_z] = [0, 9.81, 0]$. To add this gravity vector in the

virtual IMU frame at the current state, the rotation matrix $R^{vG}(q)$ was used. Re-expressing the gravity vector in the OpenSim ground frame G to the local virtual sensor frame v. It must be noted that this rotation matrix is state dependent. This leads to the following relation used

$$g^{v} = R^{vG}(q)g^{G}. \tag{3-30}$$

Then the final measurement model, adapted from (3-24) can be written as

$$h_t\left(x_t\right) = \begin{pmatrix} \omega^{v} \\ a^{v} + g^{v} \end{pmatrix} \quad \forall v \in \mathcal{V}. \tag{3-31}$$

With the measurement models in place for both the experimental and virtual IMUs, attention is now turned to the state vector which will be discussed in the next section.

3-4 Augmenting the state with the control input

To obtain a congruent reconstruction of the original motion, first, the model's states have to be defined. For this new algorithm, a similar but adapted approach from Dorschky et al. (2019) [6] is taken. Similarly to their approach, the state vector for the model at hand will consist of the joint generalized coordinates q and their rates of change, the joint generalized velocities u. However, in this IEKF-OS motion reconstruction algorithm, a different formulation of the tracking control problem is posed. The idea is to also include the joint control torques τ in the model's state vector that actuate the (bio)mechanical model. Augmenting the state with these torques allows for estimating these inputs in the IEKF-OS algorithm and subsequently applying them to the model. This results in the augmented state vector x as

$$x = \begin{pmatrix} q \\ u \\ \tau \end{pmatrix}. \tag{3-32}$$

With the state vector defined, the next step is to look at the equations that govern the updates of these state variables, the so-called equations of motion (EoM), which will be discussed next.

3-5 Motion models

The EoM, represented by a Simbody system, are the equations that govern the instantaneous rates of change for the state variables [39]. When the rate of change equations for these states are integrated, the trajectories of these states through time can be obtained. First, the motion model of the generalized coordinates q and generalized velocities u will be given. After that, it will be shown what model was devised for the unknown control input which has to be estimated and applied during simulation to obtain an accurate motion reconstruction.

3-5-1 Motion model for the generalized coordinates and generalized speeds

For systems where the state variables, consisting of both the generalized coordinates q, and the generalized speeds u, are defined dynamically by differential equations [39], the EoM that

describe the system are given by

$$\dot{q} = N(q)u, \tag{3-33a}$$

$$M(q)\dot{u} = f_{app}(t, q, u) - f_{bias}(q, u), \tag{3-33b}$$

$$n(q) = 0. \tag{3-33c}$$

Here (3-33c) specifies additional constraints that q must satisfy for the case when there are not enough equations of the form (3-33a) to specify q in a unique way [39]. This could for instance occur when there are fewer u's than q's as Simbody models q as quaternions for which $n(q)$ is then the quaternion normalization constraint. Moreover, $N(q)$ is again the kinematic coupling matrix as was previously introduced in Section 3-3-1. $M(q)$ is the $n \times n$ symmetric, positive definite mass matrix in the mobility space (u-space). Here n is the number of Degrees of Freedom (DoF) of the system. This matrix $M(q)$ contains all the inertial properties of the model in the current configuration. The term $f_{bias}(q, u) \in \mathbb{R}^{n\times 1}$ is quadratic in u and zero if $u = 0$ and corresponds to the forces representing velocity-induced Coriolis acceleration and gyroscopic terms. Lastly, the term $f_{app}(t, q, u) \in \mathbb{R}^{n\times 1}$ denotes the set of all applied forces and torques, including the gravity force. This set is mapped into an equivalent set of n generalized forces that act along the mobilities.

With these EoM defined for both the generalized coordinates q and the generalized speeds u, the time stepper study then seeks to obtain the trajectories for $q(t)$ and $u(t)$ as

$$q(t) = q\left(t_0\right) + \int_{\mathrm{T}=t_0}^{t} \dot{q}(\mathrm{T})d\mathrm{T}, \tag{3-34a}$$

$$u(t) = u\left(t_0\right) + \int_{\mathrm{T}=t_0}^{t} \dot{u}(\mathrm{T})d\mathrm{T}. \tag{3-34b}$$

Note that (3-33b) is just a version of Newton's second law $F = ma$, which relates forces to accelerations. Formally, one would solve for the accelerations as

$$\dot{u} = M^{-1}(f_{app} - f_{bias}). \tag{3-35}$$

However, Simbody does not solve these equations "$formally$" by computing the inverse of this mass matrix M as this is quite an expensive computational operation for large systems [41]. There is always a special structure to the mass matrix M that can be taken advantage of, from which the accelerations can be determined directly in $O(n)$ time. Contrary, computing this matrix inversion would take $O(n^3)$ time [39]. For that reason, Simbody does not form nor factor M while it solves (3-35). Interested readers can consult [41] and [42] for more details. Moreover, for interested readers who want to simulate their own user-defined OpenSim model, the implementation is presented in Appendix B.

With the EoM defined for both the generalized coordinates q and generalized speeds u, the motion model for the joint torque τ will now be presented.

3-5-2 Motion model for the joint torque

Contrary to the EoM that define the evolution of the generalized coordinates q and generalized speeds u of the system, the joint torque τ in the state vector does not have a motion model

that is based on physics. As the value of this state variable is arbitrary and unknown, the idea is to model the evolution of this state using a random walk model as is similarly done in [43]. Random walk models are a good initial guess when the underlying model that describes how the input τ is generated is missing. The random walk model for the joint torque τ is defined as

$$\tau(t+1) = \tau(t) + w_\tau(t), \tag{3-36}$$

with $w_\tau(t)$ a white-noise sequence that is uncorrelated with the process noises of both the generalized coordinate $w_q(t)$ and the generalized speed $w_u(t)$. Moreover, this process noise $w_\tau(t)$ is also assumed to be uncorrelated with the measurement noise. This white-noise sequence has the following covariance representation

$$w_\tau(t) = Q_\tau^{1/2}\tilde{w}_\tau(t), \qquad \tilde{w}_\tau(t) \sim (0, \mathcal{I}_m), \tag{3-37}$$

where m is the number of joint torques. For these kinds of signals, the problem is to find the input covariance matrix Q_τ which yields the best tracking behavior in terms of the Root-Mean Square Error (RMSE) between the estimates of the joint angles and the robot encoder measurements. This covariance matrix Q_τ can, therefore, be considered as a tuning parameter. When it is assumed that this input, the joint torque τ, is "almost" constant, the elements should be set equal to small values. Increasing the values of these elements will allow for more variation in the control signals applied at the various joints of the system. To test whether this is a good choice, the reader can refer to Appendix C. There the motion reconstruction algorithm was applied to a double pendulum with a known control input allowing for this verification.

3-6 The IEKF-OS motion reconstruction algorithm

With the state vector and all the motion and measurement models defined, the state estimation algorithm used in this thesis can now be introduced. This is the main keystone of this motion reconstruction algorithm. First, observe that the measurement model as shown in (3-29) and the motion model as given in (3-33), are nonlinear functions. Combining these models result in the nonlinear equations that describe the system as

$$x_t = f_{t-1}(x_{t-1}, w_{t-1}), \tag{3-38a}$$
$$y_t = h_t(x_t, e_t). \tag{3-38b}$$

Due to the nonlinearity of these equations, and the idea to estimate the states consisting of the generalized coordinates q, generalized speeds u, and the joint torques τ, the Iterated Extended Kalman Filter (IEKF) technique was chosen. This filtering approach allows for dealing with nonlinear models and estimating the system's states. The equations (3-38), describing the dynamics of the system, are internally formulated in OpenSim for a user-defined system. Hence the name *Iterated Extended Kalman Filter-OpenSim* (IEKF-OS), was chosen for the motion reconstruction algorithm. Prior to presenting this algorithm, first, the working principles of the general Kalman Filter (KF) will be discussed for readers unfamiliar with this filter. Especially, as this IEKF is based on this Kalman filtering method.

3-6-1 The Kalman filtering technique

The Kalman Filter is a recursive filter that gives an unbiased minimum variance estimate of the state of a linear dynamic system [43]. Hence, this filter works under the assumption that both the motion and measurement models are described by linear equations. Furthermore, the algorithm of the linear Kalman Filter is based on conditional probability theory. It assumes that both the process and measurement noises are Gaussian, that is, that they can be described using Gaussian distributions. An important characteristic of this filtering approach has to do with the fact that the probabilistic temporal model is a *First-Order Markov process*. The *Markov property* states that: a stochastic process has the Markov property if the conditional probability density function of the current state, given the previous state, only depends on this previous state and not on earlier states or other measurements [44]. Kalman filters are conceptualized with two stages, namely the *predict* stage and the *update* stage. For each time step, the filter starts with the prediction stage which only uses the state estimate from the previous time instance to produce an estimate of the state at the current time step. This predicted current state estimate is commonly defined as the *a-priori* estimate as it does not take into account information from the current measurement. Once this *a-priori* estimate is computed, the *update* step follows. This step combines this *a-priori* estimate with the current measurement to improve the estimate of this state. This updated state is usually denoted as the *a-posteriori* state estimate.

The important key point here is that when the state transition function $f(x)$ and measurement function $h(x)$ are linear and both assumed to be affected by Gaussian noise, then after undergoing this linear transformation, the distribution maintains its Gaussian property. Contrary, when $f(x)$ and $h(x)$ are nonlinear functions, which is the case for the system in this thesis, then the resulting state distribution might not be linear. This could result in the Kalman algorithm not converging. The Iterated Extended Kalman Filter used in this thesis, is able to deal with these nonlinear models and will be presented next.

3-6-2 The Iterated Extended Kalman Filter

The Iterated Extended Kalman Filter (IEKF) is called "extended" as it is an extension of the KF to nonlinear systems. The main difference between the IEKF with the KF is that it linearizes the nonlinear motion and measurement functions by computing Jacobian matrices. The choice of using an IEKF is a good option for state estimation when these nonlinear models can be well approximated by linearization. The drawback of this approach may be that it is difficult to calculate these Jacobians analytically due to the complicated derivatives. Moreover, one also must note that the IEKF cannot be applied to systems with a discontinuous model as those are not differentiable. The IEKF is called "iterated" as it computes multiple iterations of the update step. These iterations, however, all rely on the same measurement information of the current time step. The aim here is to approximate the actual underlying state values better by computing these iterations.

In the IEKF algorithm, the belief function for x_t is constrained to be Gaussian and given as

$$p(x_t|\check{x}_0, y_{0:t}) = \mathcal{N}(\hat{x}_t, \hat{P}_t), \tag{3-39}$$

where $\hat{x}_t$ is the mean and $\hat{P}_t$ the covariance [45]. As shown in the beginning of Section 3-6, the IEKF thus makes use of a *nonlinear* state-space model. Moreover, it is assumed that both

the process and measurement noises are Gaussian distributed with zero-mean and constant covariance [10]. This yields the following state-space model

$$x_t = f_{t-1}(x_{t-1}, w_{t-1}), \tag{3-40a}$$
$$y_t = h_t(x_t, e_t), \tag{3-40b}$$

where both the noises can be described as: $w_{t-1} \sim \mathcal{N}(0, Q_{t-1})$ and $e_t \sim \mathcal{N}(0, R_t)$. The first step of the IEKF algorithm consists in linearizing the nonlinear motion model about the posterior mean estimate of the previous state. The linearization is performed using a Taylor expansion which is of the following general form

$$f(x) = f(a) + \frac{\partial f(x)}{\partial x}\bigg|_{x=a} (x - a) + H.O.T., \tag{3-41}$$

where a symbolizes the operation point around which the linearization is performed and $(H.O.T.)$ denotes the *Higher Order Terms* which are neglected.

Instead of working with double subscripts $x_{t|t-1}$ and $x_{t|t}$ as is commonly done, e.g., [10], throughout this section, $\check{x}_t$ is used to denote the prediction given the state transition model at time t which corresponds to the common notation $x_{t|t-1}$. Next to that, $\hat{x}_t$ is used to denote the corrected prediction given the measurement at time t which corresponds to the common notation of $x_{t|t}$. In the IEKF algorithm, the system is linearized around the most recent state estimate and zero noise [45]. For the nonlinear motion model as given in (3-40a), this means that the linearization is performed about the posterior mean estimate of the previous state, denoted as $\hat{x}_{t-1}$. For the measurement model as stated in (3-40b), this means that the linearization for the first iteration takes place about the current predicted state, denoted as $\check{x}_t$. The linearizations of the measurement model for the subsequent iterations will take place around the iterated computed state denoted as $\bar{x}_t^i$ as will be shown later in this section. As the prediction step of the IEKF algorithm only makes use of the motion model, the *linearized* motion model can be written as

$$x_t = f_{t-1}(x_{t-1}, w_{t-1}), \tag{3-42a}$$
$$\approx \check{x}_t + F_{t-1}(x_{t-1} - \hat{x}_{t-1}) + L_{t-1} w_{t-1}. \tag{3-42b}$$

Here the current predicted state is given by

$$\check{x}_t = f_{t-1}(\hat{x}_{t-1}, 0), \tag{3-43}$$

and the current predicted state covariance matrix $\check{P}_t$ is computed as

$$\check{P}_t = F_{t-1} \hat{P}_{t-1} F_{t-1}^\top + L_{t-1} Q_{t-1} L_{t_1}^\top. \tag{3-44}$$

As can be seen in (3-44), the prediction step of the IEKF relies on computing the F_{t-1} and L_{t-1} Jacobian matrices of the motion model derived to the state vector and noise vector respectively. These Jacobians are computed as follows

$$F_{t-1} = \frac{\partial f_{t-1}\left(x_{t-1}, w_{t-1}\right)}{\partial x_{t-1}}\bigg|_{\substack{x_{t-1}=\hat{x}_{t-1} \\ w_t=0}} \quad , \quad L_{t-1} = \frac{\partial f_{t-1}\left(x_{t-1}, w_{t-1}\right)}{\partial w_{t-1}}\bigg|_{\substack{x_{t-1}=\hat{x}_{t-1} \\ w_t=0}} \quad . \tag{3-45}$$

In Section 3-6-3, it will be shown how these Jacobians are computed using the finite difference technique and OpenSim commands.

Contrary to the KF algorithm, in the prediction and update steps of the IEKF algorithm, the *nonlinear* models are used to propagate the state estimate and compute the measurement residual. This for the reason that the motion model is linearized about the posterior mean estimate of the previous state $\hat{x}_{t-1}$ and the measurement model is linearized about the current iterated state $\bar{x}_t^i$. By definition, this linear model exactly coincides with the nonlinear model at this operating point. Besides, note that the L and M Jacobians, which are the motion model and measurement model derived with respect to the process noise and measurement noise vectors respectively, could be different depending on the state-space model used. These matrices will be equal to identity matrices for the case that both noises w and e are additive. This is, however, not always the case.

Once the first step of the algorithm, the prediction step, is completed, the second phase is initiated which takes into account the current measurement information. When only one iteration of the algorithm's second phase is computed, one obtains the Extended Kalman Filter (EKF). Note that, in the EKF case, the linearization is performed around the mean which does not necessarily has to equal the true state. Therefore, the underlying idea of computing multiple iterations of the measurement update step is to approximate this actual state better.

From the previous step, note that the prior at time t is given by

$$p(x_t|\check{x}_0, y_{0:t-1}) = \mathcal{N}(\check{x}_t, \check{P}_t). \tag{3-46}$$

Contrary to the EKF, the linearization of the nonlinear measurement model takes place around an arbitrary operating point, which is denoted as $\bar{x}_t$, yielding

$$h_t(x_t, e_t) \approx \bar{y}_t + H_t(x_t - \bar{x}_t) + M_t e_t, \tag{3-47}$$

where

$$\bar{y}_t = h_t(\bar{x}_t, 0), \quad H_t = \left.\frac{\partial h_t\left(x_t, e_t\right)}{\partial x_t}\right|_{\substack{x_t=\bar{x}_t \\ e_t=0}} \quad , \quad M_t = \left.\frac{\partial h_t\left(x_t, e_t\right)}{\partial e_t}\right|_{\substack{x_t=\bar{x}_t \\ e_t=0}} \quad . \tag{3-48}$$

The measurement model and the Jacobians H_t and M_t are thus evaluated at $\bar{x}_t$, where for ease of notation, the iteration i has been left out. With this linearized model, the joint density for the state and measurement at time t can be expressed as approximately Gaussian as [45]

$$\begin{aligned} p(x_t, y_t|\check{x}_0, y_{0:1-t}) &\approx \mathcal{N}\left(\begin{bmatrix}\mu_{x,t} \\ \mu_{y,t}\end{bmatrix}, \begin{bmatrix}\Sigma_{xx,t} & \Sigma_{xy,t} \\ \Sigma_{yx,t} & \Sigma_{yy,t}\end{bmatrix}\right), \\ &= \mathcal{N}\left(\begin{bmatrix}\check{x}_t \\ \bar{y}_t + H_t(\check{x}_t - \bar{x}_t)\end{bmatrix}, \begin{bmatrix}\check{P}_t & \check{P}_t H_t^\top \\ H_t\check{P}_t & H_t\check{P}_t H_t^\top + M_t R_t M_t^\top\end{bmatrix}\right). \end{aligned} \tag{3-49}$$

When the measurement information is incorporated, hence known, the Gaussian conditional density for the *posterior* x_t, equivalent to equation (3-39) as given in the beginning of this section, can be written as

$$p(x_t|\check{x}_0, y_{0:t}) = \mathcal{N}\Big(\underbrace{\mu_{x,t} + \Sigma_{xy,t}\Sigma_{yy,t}^{-1}(y_t - \mu_{y,t})}_{\hat{x}_t}, \underbrace{\Sigma_{xx,t} - \Sigma_{xy,t}\Sigma_{yy,t}^{-1}\Sigma_{yx,t}}_{\hat{P}_t} \Big), \tag{3-50}$$

where the mean is defined as $\hat{x}_t$ and $\hat{P}_t$ as the covariance. For the probabilistic derivation of this *posterior* x_t as given in (3-50), the reader is referred to Appendix A. Observing (3-50) closely, one can see that the Kalman gain is given by $\Sigma_{xy,t}\Sigma_{yy,t}^{-1}$. Moreover, $\mu_{x,t}$ and $\Sigma_{xx,t}$ are set to $\mu_{x,t} = \check{x}_t$ and $\Sigma_{xx,t} = \check{P}_t$. At this point, deducing from (3-50), the generalized Gaussian update equations can be derived as

$$K_t = \Sigma_{xy,t}\Sigma_{yy,t}^{-1}, \tag{3-51a}$$
$$\hat{P}_t = \check{P}_t - K_t\Sigma_{xy,t}^{\top}, \tag{3-51b}$$
$$\hat{x}_t = \check{x}_t + K_t(y_t - \mu_{y,t}). \tag{3-51c}$$

For the IEKF setting, contrary to the general Kalman Filter, now the moments of $\mu_{y,t}, \Sigma_{yy,t}$ and $\Sigma_{xy,t}$ as shown in (3-49) are now substituted which yield the equations for the second phase of the IEKF algorithm

$$K_t = \check{P}_t H_t^{\top}(H_t\check{P}_t H_t^{\top} + M_t R_t M_t^{\top})^{-1}, \tag{3-52a}$$
$$\hat{P}_t = (\mathcal{I} - K_t H_t)\check{P}_t, \tag{3-52b}$$
$$\hat{x}_t = \check{x}_t + K_t\Big(y_t - \bar{y}_t - H_t(\check{x}_t - \bar{x}_t)\Big). \tag{3-52c}$$

Note that, as was already stated, if the linearization in the IEKF takes place around the mean of the predicted prior $\bar{x}_t = \check{x}_t$, then, for that case, this results in the EKF algorithm. The point of the IEKF, however, is that a much better approximation can be obtained when (3-52) is computed iteratively [45]. The algorithm is initialized with $\bar{x}_t = \check{x}_t$. As explained previously, the mean $\hat{x}$ does not necessarily need to represent the actual state, hence the call for this IEKF approach to approximate the underlying actual state better. Commonly, the algorithms' iterations, denoted by i, for the state x at each time instant t are terminated when a maximal number of user-set iterations ϵ have been reached. The iterations can also be stopped when the change between $\bar{x}_t^i$ and $\bar{x}_t^{i+1}$ is smaller than a user-defined threshold δ. In this thesis, the former criterion is taken. Typically, the algorithm converges within three measurement update iterations. The IEKF algorithm is shown in Algorithm 1.

Algorithm 1 The Iterated Extended Kalman Filter

INPUTS: Measurement data and an initial $\hat{x}_0$ and $\hat{P}_0$. Q_{t-1} and R_t are the covariance matrices of w_{t-1} and e_t, respectively. A threshold δ or maximal number of iterations ϵ.
OUTPUTS: Estimates of the state $\hat{x}_t$ and its covariance $\hat{P}_t$.

Note:

$\check{x}_t$ = Prediction given the state transition model at time t.

$\hat{x}_t$ = Corrected prediction given the measurement at time t.

for t=1:N **do**

Step 1: Time update / Prediction step

$$\check{x}_t = f_{t-1}(\hat{x}_{t-1}, 0), \tag{3-53a}$$

$$\check{P}_t = F_{t-1}\hat{P}_{t-1}F_{t-1}^\top + L_{t-1}Q_{t-1}L_{t-1}^\top. \tag{3-53b}$$

Set $i = 1$.

while $||\bar{x}_t^{i+1} - \bar{x}_t^i|| \leq \delta$ **do or for** i=1:ϵ

Step 2: Measurement update iterations / Correction step iterations

$$H_t^i = \left.\frac{\partial h_t(x_t, e_t)}{\partial x_t}\right|_{\substack{x_t=\bar{x}_t^i \\ e_t=0}}, \qquad M_t^i = \left.\frac{\partial h_t(x_t, e_t)}{\partial e_t}\right|_{\substack{x_t=\bar{x}_t^i \\ e_t=0}}. \tag{3-54}$$

$$K_t^i = \check{P}_t {H_t^i}^\top (H_t^i \check{P}_t {H_t^i}^\top + M_t^i R_t {M_t^i}^\top)^{-1}, \tag{3-55a}$$

$$\bar{x}_t^{i+1} = \check{x}_t + K_t^i\Big(y_t - h_t(\bar{x}_t^i, 0) - H_t^i(\check{x}_t - \bar{x}_t^i)\Big). \tag{3-55b}$$

Set $i = i + 1$.

end while or end for

Step 3: Update the state and the covariance matrix by setting:

$$\hat{x}_t = \bar{x}_t^{i+1}, \tag{3-56a}$$

$$\hat{P}_t = \Big(\mathcal{I} - K_t^i H_t^i\Big)\check{P}_t. \tag{3-56b}$$

end for

3-6-3 Computing the Jacobian matrices

Given the motion model and the measurement model, as defined in (3-38), one can easily obtain an approximation to the four various Jacobians required for the IEKF algorithm by perturbing the state variables one by one. This method is commonly referred to as a finite difference approximation. The Jacobians for the prediction step, F_{t-1} and L_{t-1}, and the Jacobians for the update step, H_t and M_t will be computed column wise. For the computation of these Jacobians, Table 3-1 will at each row show which various functions F(χ) and vectors χ need to be used as an input to Algorithm 2 such that the desired Jacobian can be computed. For instance, if one wants to compute the F_{t-1} Jacobian, from Table 3-1, it can be seen that

$\mathrm{F}(\chi)$ should be the motion model $f(x)$ and χ should be the state vector x_{t-1} that are to be used as inputs to Algorithm 2.

Table 3-1: The various functions and vectors used for computing the Jacobians.

Jacobian to be computed	**Function $\mathrm{F}(\chi)$ used for evaluation**	**Vector χ used for evaluation and perturbation**
F_{t-1}	Motion model: $f_{t-1}(x)$	State vector: x_{t-1}
L_{t-1}	Motion model: $f_{t-1}(x)$	Noise vector: w_{t-1}
H_t	Measurement model: $h(x)$	State vector: x_t
M_t	Measurement model: $h(x)$	Noise vector: e_t

Iteratively perturbing the various states/noises one by one in the vector χ to compute the Jacobian is summarized in Algorithm 2.

Algorithm 2 Numerical computation of the Jacobian

INPUTS: The length n of the vector χ and the size of the perturbation step ϵ. Depending on the Jacobian to be computed, the corresponding function $\mathrm{F}(\chi)$ and vector χ as shown in Table 3-1.
OUTPUTS: Numerically computed Jacobian J.

Note:

$\tilde{\chi}$ = Perturbed state vector.
χ = Original state vector.

Set perturbed vector equal to the original vector

$$\tilde{\chi} = \chi. \tag{3-57}$$

for i=1:n **do**

$$\tilde{\chi}(i) = \tilde{\chi}(i) + \epsilon, \tag{3-58a}$$
$$J(:, i) = \left(\mathrm{F}(\tilde{\chi}) - \mathrm{F}(\chi)\right)/\epsilon, \tag{3-58b}$$
$$\tilde{\chi}(i) = \chi(i). \tag{3-58c}$$

end for

To evaluate the current function $\mathrm{F}(\chi)$, which is *not perturbed*, it is important to realize the acceleration stage internally in OpenSim. This command will, for instance, compute the updates of the state variables being $\dot{q}$ and $\dot{u}$. An important note must be made, however. If one wants to perturb the vector, a "copy" of the original OpenSim `state` object must be created. The idea for copying this `state` object lies in the fact that in this way, the values of the OpenSim `state` object, which are used in the actual simulation, are not affected. Moreover, if one wants to evaluate the function with the perturbed vector, $\mathrm{F}(\tilde{\chi})$, again it is important to realize the acceleration stage internally in OpenSim on this copied `state` object.

3-6-4 Discretize motion model

The motion models introduced in Section 3-5 govern the dynamics of the system. Note, however, that these equations are still given in continuous time, while the IEFK presented in Section 3-6-2 is a discrete filter. For that reason, the computed F and L Jacobian matrices have to be discretized such that they can be used in this filter. Various ways to discretize these Jacobian matrices exist. A common approach to discretize is by using the matrix exponential which results in the following discrete F and L Jacobian matrices

$$\begin{aligned} F_{dis} &= e^{F_{con}\Delta t}, \\ L_{dis} &= e^{L_{con}\Delta t}, \end{aligned} \tag{3-59}$$

where F_{dis} and L_{dis} are the discretized Jacobian versions of the continuous time Jacobians F_{con} and L_{con}. Moreover, Δt is the sampling time which is set equal to the sampling time of the experimental Xsens MTw Awinda IMUs.

With all the steps detailed, the resulting Iterated Extended Kalman Filter - OpenSim (IEKF-OS) can be presented. The IEKF-OS algorithm is shown in Algorithm 3.

3-7 Conclusions

In this chapter, a novel approach to reconstruct a system's motion is presented. The method developed is based on the IEKF algorithm and the system's dynamical model generated from OpenSim.

Compared to the previously discussed sensor fusion algorithm [15] in Section 2-2, the idea behind this thesis' approach is to include more information about the system. For instance incorporating the rigid-body masses and lengths, the moments of inertia, and their center of mass locations. Moreover, this approach also takes into account the type of the joint between two linked rigid-body segments. As an extension to this previously discussed sensor fusion method, including this additional system information allows to simultaneously estimate the required joint torques needed to drive the system. The overall solution only requires a (bio)mechanical model of the system at hand in OpenSim, and experimentally obtained raw angular velocity and linear acceleration data from the system's attached IMUs.

Prior to validating the algorithm by carrying out experiments, first, this algorithm needs to be verified. The algorithm will be verified on the KUKA's OpenSim model via numerical simulations. This verification will be described in Chapter 4.

Algorithm 3 The Iterated Extended Kalman Filter - OpenSim (IEKF-OS) algorithm

INPUTS: Measurement data and an initial $\hat{x}_0$ and $\hat{P}_0$. Q_{t-1} and R_t are the covariance matrices of w_{t-1} and e_t, respectively. A threshold δ or maximal number of iterations ϵ. An OpenSim model with one virtual IMU frame attached to each individual body segment.
OUTPUTS: Estimates of the state $\hat{x}_t$ and its covariance $\hat{P}_t$.

Note:

$\check{x}_t$ = Prediction given the state transition model at time t.

$\hat{x}_t$ = Corrected prediction given the measurement at time t.

for t=1:N **do**

Step 1: Time update / Prediction step

A: Set $\hat{q}_{t-1}$, $\hat{u}_{t-1}$ and apply $\hat{\tau}_{t-1}$. Then realize the OpenSim *acceleration stage*.

B: Form F_{t-1} and L_{t-1} Jacobians using Algorithm 2 and discretize using (3-59).

$$F_{t-1} = \left.\frac{\partial f_{t-1}\left(x_{t-1}, w_{t-1}\right)}{\partial x_{t-1}}\right|_{\substack{x_{t-1}=\hat{x}_{t-1} \\ w_t=0}} \quad , \; L_{t-1} = \left.\frac{\partial f_{t-1}\left(x_{t-1}, w_{t-1}\right)}{\partial w_{t-1}}\right|_{\substack{x_{t-1}=\hat{x}_{t-1} \\ w_t=0}} \quad . \tag{3-60}$$

C: Integrate EoM via OpenSim using (3-34) and as detailed in Appendix B.

D: Get predicted states $\check{x}_t$ from OpenSim and compute $\check{P}_t$.

$$\check{x}_t = f_{t-1}(\hat{x}_{t-1}, 0), \tag{3-61a}$$

$$\check{P}_t = F_{t-1}\hat{P}_{t-1}F_{t-1}^\top + L_{t-1}Q_{t-1}L_{t-1}^\top. \tag{3-61b}$$

Set $i = 1$.

while $||\bar{x}_t^{i+1} - \bar{x}_t^i|| \leq \delta$ **do or for** i=1:ϵ

Step 2: Measurement update iterations / Correction step iterations

A: Set $\bar{q}_t^i$, $\bar{u}_t^i$ and apply $\bar{\tau}_t^i$. Then realize the OpenSim *acceleration stage*.

B: Compute virtual IMU frame local angular velocity and local linear acceleration.

C: Form H_t^i and M_t^i Jacobians using Algorithm 2.

$$H_t^i = \left.\frac{\partial h_t(x_t, e_t)}{\partial x_t}\right|_{\substack{x_t=\bar{x}_t^i \\ e_t=0}} \quad , \qquad M_t^i = \left.\frac{\partial h_t(x_t, e_t)}{\partial e_t}\right|_{\substack{x_t=\bar{x}_t^i \\ e_t=0}} \quad . \tag{3-62}$$

D: Load angular velocity and linear acceleration IMU measurements.

$$K_t^i = \check{P}_t {H_t^i}^\top (H_t^i \check{P}_t {H_t^i}^\top + M_t^i R_t {M_t^i}^\top)^{-1}, \tag{3-63a}$$

$$\bar{x}_t^{i+1} = \check{x}_t + K_t^i\Big(y_t - h_t(\bar{x}_t^i, 0) - H_t^i(\check{x}_t - \bar{x}_t^i)\Big). \tag{3-63b}$$

Set $i = i + 1$.

end while or end for

Step 3: Update and set the state. Update the covariance matrix.

$$\hat{x}_t = \bar{x}_t^{i+1}, \tag{3-64a}$$

$$\hat{P}_t = \Big(\mathcal{I} - K_t^i H_t^i\Big)\check{P}_t. \tag{3-64b}$$

end for

Chapter 4

Simulation-based verification of the motion reconstruction algorithm

This chapter describes the simulations performed to verify the IEKF-OS algorithm's reconstruction ability and to assess its performance in doing so. In particular by performing simulations on the OpenSim model of the KUKA LBR iiwa 7 R800 robot manipulator. This model corresponds to the robotic arm system on which actual experiments, as later detailed in Chapter 5, will be conducted. In this chapter, however, the algorithm will first be evaluated for a scenario incorporating no virtual sensor modeling placement errors. As such, verifying the applicability of the IEKF-OS algorithm for this intricate robotic system. Next to that, a brief sensitivity analysis will be performed for different scenarios incorporating sensor placement errors like translational and rotational offsets. Lastly, this chapter is concluded with some final remarks.

4-1 Problem statement for the KUKA robot manipulator system

In Chapter 3, the motion reconstruction algorithm has been presented. Compared to a classical control approach, one would first have to filter the incoming noisy measurement data using a dedicated filtering block. Then a control block is needed to achieve the desired reconstruction by means of a tracking objective. Looking at this formulation, given a (bio)mechanical system's motion to be reconstructed, the proposed method in this thesis combines both of these blocks in one scheme. As such, it provides both the filtered state estimation and computation of the control input through a single minimization function. This takes place in the Kalman filtering update equation leading to the motion's reconstruction.

Initially, the estimated joint angles $\hat{q}$, joint angular velocities $\hat{u}$ and the joint torques $\hat{\tau}$, will differ from the actual system state variables q, u and τ. Hence, the only way to minimize this discrepancy is by computing proper control actions. Usually, in classical control approaches, the control input, here denoted as τ, remains separate from the systems state variables x.

However, in this algorithm, the state vector, compromising the joint angles q and joint angular velocities u, is augmented with this control input τ. This augmentation allows for estimating the proper control torque in the Kalman filtering scheme, required to minimize this discrepancy between the actual system's motion and its reconstructed one.

As was described in Section 3-5-2, the motion model for the joint torque τ was chosen to be a random walk model. To test whether this is a good choice, a simulation on an actuated double pendulum was performed. As the aim for this chapter was to reconstruct the robot manipulator's motion, for conciseness of this chapter, interested readers can refer to Appendix C for the double pendulum results obtained. From these results, as shown in Figure C-4 and Figure C-5, it can be observed that the original motion and the original control input were accurately reconstructed.

In fact, the interconnection between the state variables and control input is the key feature of this algorithm together with the inclusion of the dynamical models governing the motion of the system. This approach, moreover, also allows to adapt to systems composed of multiple rigid body links as will be seen in Chapter 5. There the algorithm's performance will be tested and validated on a real robotic arm. To conclude, the best reconstruction of the motion is obtained when the internal belief about the system's states coincide with the actual dynamics observed from the attached IMUs.

First, the KUKA robot manipulator will be presented, after which a schematic overview of the workflow will be illustrated.

The KUKA LBR iiwa 7 R800 robot

The system at hand is the industrial manipulator, the KUKA LBR iiwa 7 R800 robot. This robotic arm was chosen as this manipulator, consisting of seven links, is equipped with position sensors in each of the joints. These encoders ensure noise-free measurements of the angular position of an axle in a given rotary motion. Having access to these encoder measurements will be especially useful for the validation purpose of the IEKF-OS estimates which will be presented in the next chapter. Mainly, as these joint encoder values can be considered as gold standard ground truth data to which the estimates obtained from the motion reconstruction algorithm can be validated. It must be noted, however, that it is not always possible to place these encoders at the desired positions. Especially in the context of biomechanics. Moreover, these encoders are usually bulky and rather expensive devices [46]. On the contrary, with the KUKA robot, these drawbacks are eliminated as these encoders have been implemented inside the links of this manipulator robot. Besides, the actual KUKA robot features torque controlled motors allowing to change each joint angle individually from the other joint angles. The KUKA, compared to other conventional industrial manipulators, features 7 DoFs. This extra DoF results in an increased dexterity and is thus able to avoid certain singularities. The ranges of motion and the maximal joint velocities of this robot are given in Appendix D-2. For this thesis, only six Xsens MTw Awinda IMUs were available. Therefore, the seventh DoF, being the orientation of the end-effector, will not be tracked.

Performed tests

The IEKF-OS performance will be tested on the KUKA robot OpenSim model for four different simulation scenarios being

1. An idealized scenario which includes no sensor placement errors.
2. Scenarios incorporating translation sensor placement errors.
3. Scenarios incorporating rotational sensor placement errors.
4. Scenarios incorporating translation and rotational sensor placement errors.

For each scenario presented, the same movement, being an infinity shaped trajectory, will be reconstructed. A schematic overview of the workflow is illustrated in Figure 4-1.

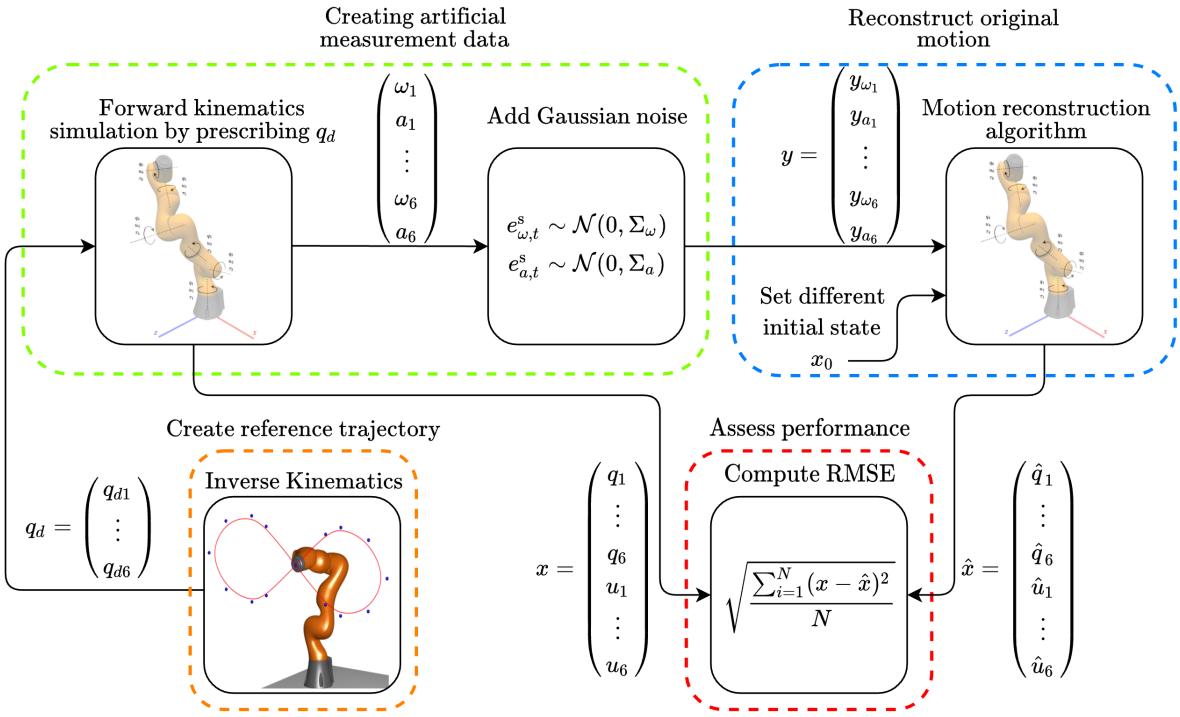


Figure 4-1: Workflow overview of the performed simulations. First, in the orange box, an infinity shaped reference trajectory is created and the corresponding desired joint angles q_d are obtained from an IK solution. These angles q_d are then prescribed to the OpenSim KUKA model and a forward kinematics simulation is performed. With created artificial measurement data, the motion is then reconstructed using the developed IEKF-OS algorithm. Lastly, the reconstruction performance is assessed by computing the RMSE values between actual state variables q and $\hat{q}$, u and $\hat{u}$.

The idea is to have the robot track a smooth and continuous infinity shaped reference trajectory. Therefore, first, this reference trajectory was created which will be used for all four numerical simulation scenarios. The artificial measurement data, as required as input for the IEKF-OS algorithm, will be created by performing a forward kinematics simulation. This forward kinematics simulation will be achieved by prescribing desired joint angles to the KUKA

robot model in OpenSim. This results in the end-effector tracking this reference infinity shaped trajectory. To obtain these desired joint angles, an inverse kinematics (IK) algorithm was run. Note that, for these simulated scenarios, no joint torques are computed in the green box in Figure 4-1. Rather the desired joint angles are prescribed to the OpenSim KUKA model. For that reason, the torques as estimated by the IEKF-OS reconstruction cannot be compared. In the following sections, the reconstruction of the original motion will be explained concisely. With the workflow illustrated, first, the KUKA LBR iiwa 7 R800 OpenSim model will be presented.

4-2 The KUKA LBR iiwa 7 R800 OpenSim model

The KUKA LBR iiwa 7 R800 system was modeled using the inertial properties of each KUKA's rigid body. This term inertial properties encompasses the rigid body's mass, its center of mass locations, and the moments of inertia. These inertial properties for each link of the KUKA robot were obtained from the GitHub file of Chatzilygeroudis et al. (2019) [47]. The values of these inertial properties for each KUKA's rigid body are given in Appendix D in Table D-1. The links of the KUKA are connected by revolute joints, allowing the child body to rotate with respect to its parent body. Using this modeling information and inertial properties, an OpenSim model was constructed in `XML`-code. From this code, an `osim` model can be created and subsequently used in OpenSim. The OpenSim model, together with the six generalized coordinates q denoting the joint angles, the six generalized velocities u denoting the joint angular velocities, and the generalized moments τ denoting the joint torques are shown in Figure 4-2a.

For this thesis, the motion of the end-effector, shown in black in Figure 4-2b, is not tracked. Specifically, as for the real experiment only six IMUs were available. This end-effector is able to rotate with respect to its parent link. For this thesis, the interest is primarily focused on reconstructing the overall motion of the KUKA robot. Not on tracking the relative orientation of the end-effector with respect to the sixth link. Thus this rotation was neglected. With the model in place, the next step is to generate a reference trajectory that the end-effector has to track.

4-3 Creating artificial IMU measurements

Prior to doing a real experiment, one first would be interested to simulate the system in a realistic simulator. Verifying that the IEKF-OS algorithm developed, can be used to reconstruct motions of this robot. Hence, the procedure to create artificial IMU measurements will be outlined starting with creating the infinity shaped reference trajectory.

Reference trajectory and computing the desired joint angles

To reconstruct a motion performed by the KUKA robot, an infinity shaped reference trajectory was created. This infinity shape was used as it is a continuous and smooth trajectory that excites all the different joints. The trajectory was designed such that the whole trajectory is tracked by the end-effector in 5 [s]. Points were assigned in a 3D space. A spline

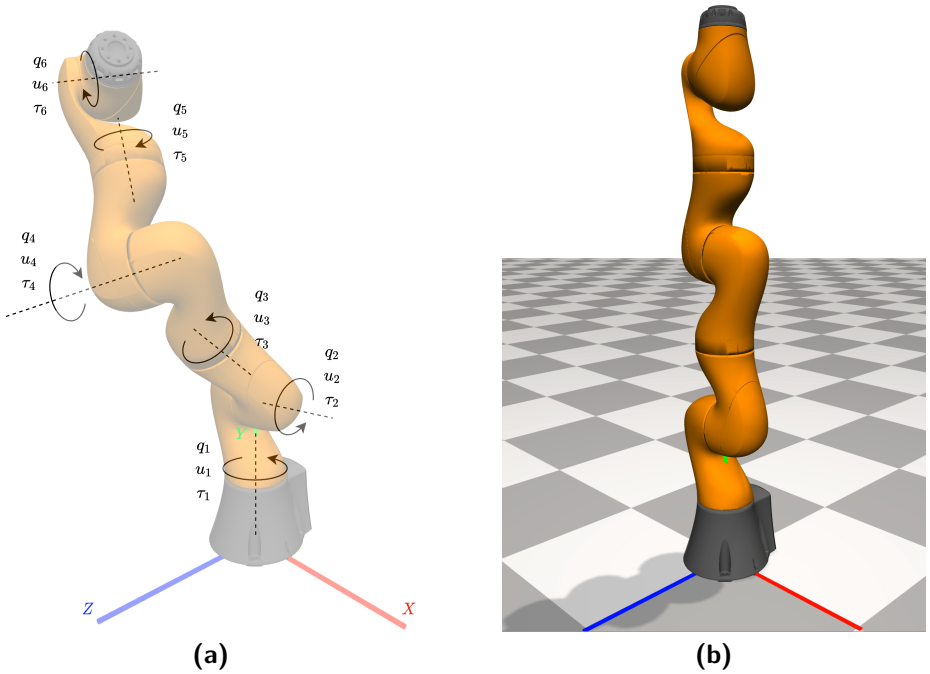


Figure 4-2: (a) Schematic overview of the KUKA robot manipulator and its generalized coordinates. (b) The modeled KUKA robot shown in the OpenSim environment in its default configuration, $q_i = 0$ [deg] $\forall i \in \mathcal{Q}$, where $\mathcal{Q}$ is the set of all joints. The OpenSim reference frame can be seen partly in the center of the base of the KUKA robot.

function was fitted with a sampling frequency of 100 [Hz] to obtain a continuous and smooth trajectory in space. This frequency was chosen similar to the sampling frequency of the later used real Xsens MTw IMUs. The XYZ-locations of these spacial points are shown in Table 4-1 and were determined with the KUKA ranges of motion in mind.

Table 4-1: The ordered XYZ-locations of the points used to create an infinity shaped trajectory.

X	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
Y	0	-0.25	-0.375	-0.6	-0.75	-0.6	-0.375	-0.25	0	0.25	0.375	0.6	0.75	0.6	0.375	0.25	0
Z	0.75	0.95	1	0.95	0.75	0.55	0.5	0.55	0.75	0.95	1	0.95	0.75	0.55	0.5	0.55	0.75

With the trajectory generated, the IK solver was called which is available in the *MATLAB Robotics System Toolbox*. This function performs the IK such that the end-effector of a robot manipulator follows the desired set of user-generated waypoints. An illustration of the generated infinity reference trajectory in 3D space is shown in Figure 4-3.

When a solution is obtained, the IK solver will return the desired values for each of the six joint angles q_d that result in having the end-effector track the reference trajectory. Looking back to the schematic overview in Figure 4-1, the next step will be to prescribe these obtained desired joint angles to the OpenSim version of the KUKA as shown there in the green box.

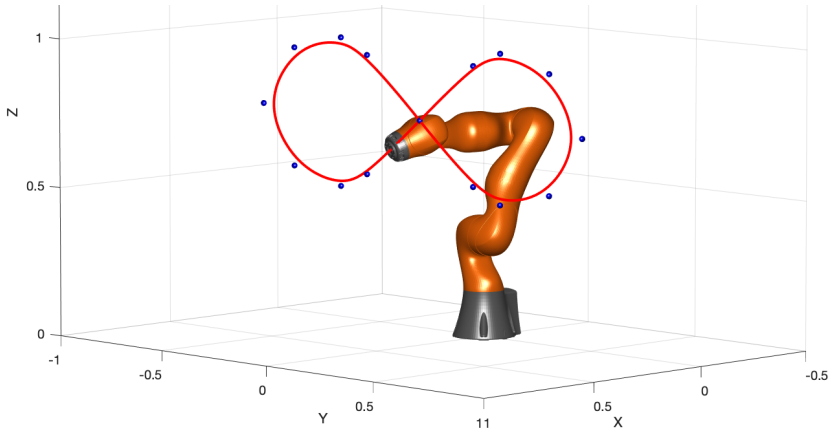


Figure 4-3: The generated infinity spline trajectory shown together with the KUKA.

Prescribing the inverse kinematics obtained solution

Using a forward kinematics simulation, the desired joint angles q_d will be prescribed to the six joint angles of the KUKA model in OpenSim. Prior to this forward kinematics simulation, virtual IMUs will be created and attached to the OpenSim model to generate artificial measurement data. Note, however, that this forward kinematic simulation does not include joint torques, but just simulates the prescribed desired coordinate values. Hence, the torques as estimated and applied by the IEKF-OS algorithm to the KUKA model can't be compared.

Creating the required measurement data

To create the required angular velocity and linear acceleration measurement data, virtual IMUs are modeled as reference frames. For each rigid body of the KUKA, such a virtual frame is attached. These virtual frames are defined with respect to their corresponding parent frames being the body frames b. The locations and orientations of these virtual IMU frames are given in Appendix D-2 in Table D-3. From the attached frame, the angular velocity ω^s and linear acceleration a^s can be obtained. The artificially created measurement data is simulated again at 100 [Hz]. This corresponds to the maximal available sampling frequency of the later used Xsens MTw Awinda's. To mimic experimental data, Gaussian noise corresponding to the actual noise disturbing the experimental IMUs was created. This was achieved by determining the covariances of stationary experimental Xsens IMUs. The noise obtained, was then added to these artificially created measurements y. This concludes the procedure in the green box in Figure 4-1. The next step is to reconstruct the original motion of the KUKA using the just created measurement data and will be presented in the next section.

4-4 Reconstructing the original motion for the idealized scenario

The actual KUKA robot manipulator has seven links, hence seven DoFs. However, for later conducted experiments, only six IMUs were available. This allows to only track just six of the links of this KUKA robot. Only the motion of the end-effector will not be reconstructed.

Therefore, this system will be described using six generalized coordinates q. Each coordinate has a generalized velocity u, and a generalized moment τ. Hence, the total state vector which is estimated in the IEKF-OS algorithm will consist of 18 states resulting in

$$x = (q_1\ q_2\ q_3\ q_4\ q_5\ q_6\ u_1\ u_2\ u_3\ u_4\ u_5\ u_6\ \tau_1\ \tau_2\ \tau_3\ \tau_4\ \tau_5\ \tau_6\ )^\top. \tag{4-1}$$

It would be interesting to see if the IEKF-OS algorithm converges to the actual state values when the IEKF-OS is initialized differently. Only the relative heading can be estimated once the system becomes more dynamic. A drawback is thus that the absolute heading cannot be estimated. This absolute heading estimate can only be obtained when magnetometers are used. For this thesis, these sensors were not used as the goal was to try and reconstruct motions based on angular velocities and linear accelerations solely. For that reason, the base joint angle was set similar to the computed desired joint angle: $q_1 = q_{1d}$. The other five joint angles were set at the desired joint angles q_d plus 5 [deg].

The measurement covariance matrix R was a block diagonal matrix consisting of the noise covariances of the six Xsens MTw Awinda IMUs. The noise covariances of these stationary IMUs sensors were found experimentally at 100 [Hz]. The measurement covariance matrix R was then set to $R \times 150$. This value was, in an iterative procedure, found to work best for this scenario. The standard deviations for all the random walk models of the joint torques were set to $\sigma_{\tau,i} = 1$ [Nm] $\forall i = 1:6$. Hereby allowing room for large deviations of the applied control input to the KUKA as estimated by the IEKF-OS. As the same model is used for motion reconstruction, the process noise on the state variables q and u were set to 0 [rad] and 0 [rad/s] respectively. The number of measurement iterations ϵ in the update step of the Kalman filter is essentially a trade-off parameter. More iterations could result in better convergence, however, at the cost of more computations. For this scenario, the parameter was set to $\epsilon = 3$.

The results obtained, for reconstructing the KUKA robot in terms of RMSE values, are shown in Table 4-2. The corresponding estimated state evolutions are shown in Figure 4-4 and Figure 4-5. The RMSE values were determined after the first second allowing the IEKF-OS algorithm to converge to the actual state variables.

Table 4-2: RMSE values obtained between the actual states q and u and the estimated states $\hat{q}$ and $\hat{u}$. The gyroscope and accelerometer measurements used, had Gaussian noise applied corresponding to the covariance matrices of experimental Xsens IMUs.

Joint angle RMSE [deg]						Joint angular velocity RMSE [deg/s]					
q_1	q_2	q_3	q_4	q_5	q_6	u_1	u_2	u_3	u_4	u_5	u_6
0.183	0.102	0.103	0.108	0.159	0.302	0.382	0.711	0.457	0.894	0.427	1.062

From the RMSE values and state trajectories obtained, it can be concluded that the IEKF-OS algorithm is able to reconstruct the original motion. One must note that the results presented here are for an idealized scenario as no sensor misalignments were introduced. Moreover, the artificial measurements created, do not suffer from drift. This is a common phenomenon observed in actual IMU sensor data, for an example see [10]. Still, the IEKF-OS algorithm provides accurate estimations of the actual states describing the motion of the KUKA system. Even when the states in this filter are initialized differently compared to the original state values.

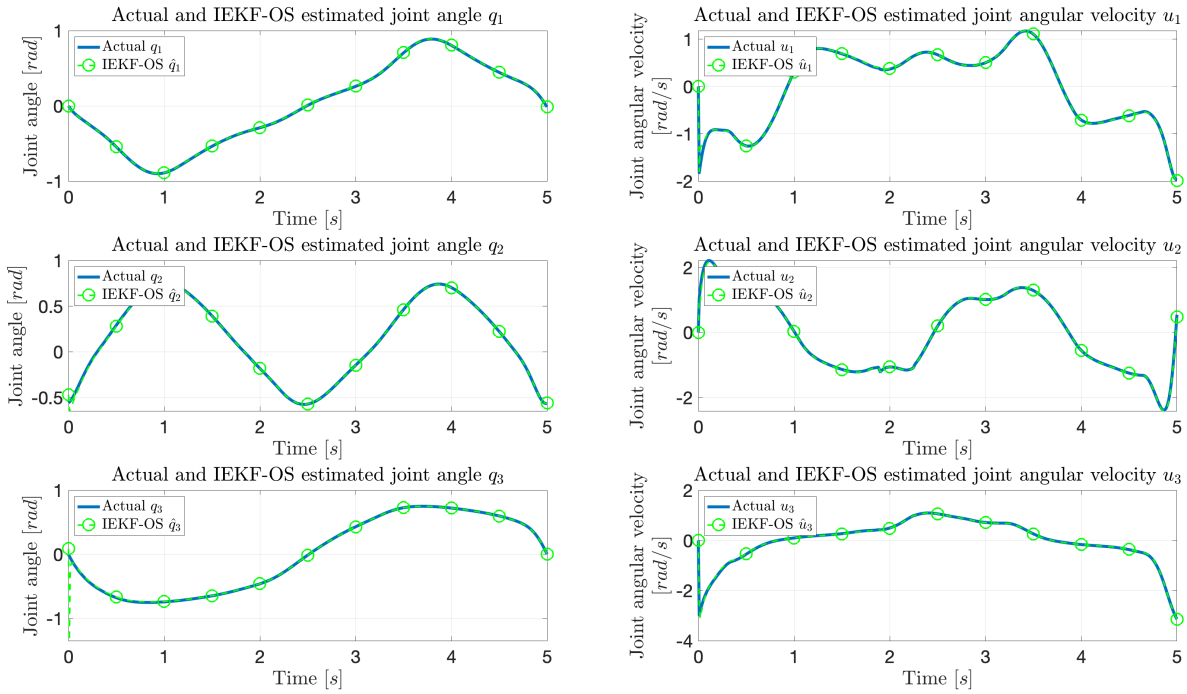


Figure 4-4: The left column of plots shows the actual and estimated joint angles q_1, q_2 and q_3. The right column of plots shows the actual and estimated joint angular velocities u_1, u_2, and u_3.

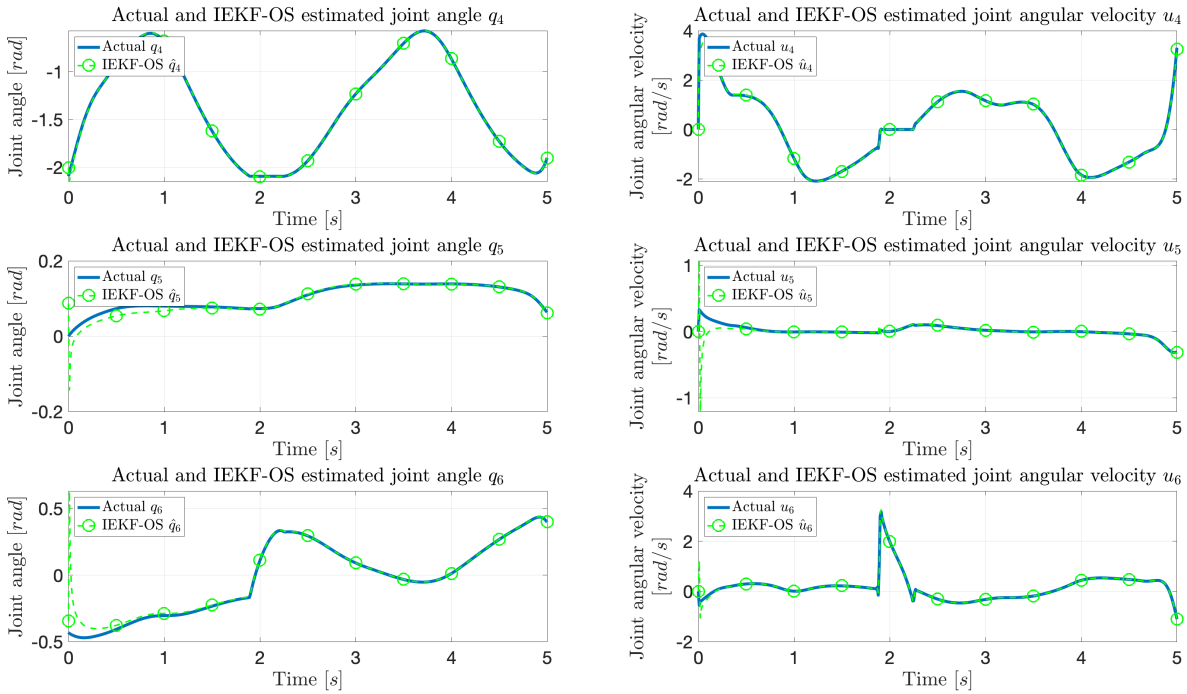


Figure 4-5: The left column of plots shows the actual and estimated joint angles q_4, q_5 and q_6. The right column of plots shows the actual and estimated joint angular velocities u_4, u_5, and u_6.

From Figures 4-4 and 4-5, it can be seen that the estimated state variables, green dashed lines

with circle markers, after converging, closely follow the actual state variable evolutions. These results illustrate that the IEKF-OS algorithm is capable of near perfect motion reconstruction.

4-5 Reconstructing the original motion incorporating sensor placement errors

The numerical simulation presented, did not incorporate sensor placement errors. However, in a real experiment, there will be a position mismatch between the modeled virtual IMUs on the OpenSim model and the experimental Xsens IMUs on the actual system. Therefore, prior to carrying out experiments, it is interesting to observe what the effect is of translational and rotational errors. Hence, a brief sensitivity analysis will be performed using Monte Carlo simulations. These translational and rotational offsets will be applied to the virtual IMUs modeled on the system creating the motion and measurements. Whereas the virtual IMUs on the IEKF-OS system will be modeled unaltered. For these latter positions and orientations, the reader is referred to Table D-3.

During the actual experiments conducted in Chapter 5, an experimental sensor-to-segment calibration needs to be performed. It needs to be known how the experimental Xsens IMU is oriented and positioned on the link it is attached to. With this information, the corresponding virtual IMU on the OpenSim rigid body can be modeled. This experimental sensor-to-segment calibration is outlined in Appendix F. This calibration procedure needs to be performed prior to each conducted new experiment.

For the orientation part of the experimental sensor-to-segment calibration, this calibration step relies on the Xsens Kalman Filter for 3D human motion (XKF3hm) algorithm. This XKF3hm algorithm estimates the experimental sensor's orientation with respect to the navigation frame n. The translation part of the experimental sensor-to-segment calibration will be performed by manually measuring the offset. Each time, the distance will be measured from the joint center, of the respective segment the IMU is attached to, to the lower-left corner of the Xsens IMU. A schematic drawing of the Xsens IMU is shown in Appendix F in Figure F-2. This is done for the reason that from this lower-left corner, the accelerometer origin is known. Moreover, all XYZ-distances, as measured from this joint center, will also be expressed in the joint center's coordinate frame.

Monte Carlo simulations were performed to analyze the IEKF-OS algorithm's motion reconstruction under the presence of these translational and rotational errors. First, in Section 4-5-1, it will be analyzed how only translational errors affect motion reconstruction performance. Then, in Section 4-5-2, no translational errors will be imposed, but rotational errors will be introduced. These different two scenarios should yield insights in which modeling error mainly dominates the degradation of the motion tracking performance. Lastly, in Section 4-5-3, both translational and rotational errors will be introduced. This latter scenario should resemble a more realistic scenario similar to the later carried out experiments.

Again, for all the Monte Carlo simulations performed, the IEKF-OS KUKA system will be initialized with the five joint angles increased by 5 [deg]. This being similar to the idealized scenario of Section 4-4.

4-5-1 Incorporating translational errors

Due to the geometry of the links of the KUKA being quite curved, it could be difficult to accurately position experimental IMUs on these links. Hence, to analyze the IEKF-OS algorithm's motion reconstructing performance under presence of translational errors, Monte Carlo simulations were performed. It is assumed, moreover, that measuring distances using a tape measure is equally error-prone for each X, Y, Z-axis in which distances are needed. For each of the six links of the KUKA robot, a virtual IMU will be attached. As such, for each individual virtual IMU, a different translation vector t_i will be created prior to each Monte Carlo simulation. This i-th vector will be of the form

$$t_i = \alpha \cdot \begin{pmatrix} \pm t_X \\ \pm t_Y \\ \pm t_Z \end{pmatrix} [cm], \tag{4-2}$$

where α is the translational perturbation value. For each individual virtual IMU, a different translational offset t_i will be added to the i-th original virtual IMU location as depicted in Table D-3.

In this section, three trials will be performed where each trial corresponds to 100 Monte Carlo simulations. The three translations perturbation values were chosen to be $\alpha_1 = 1$ [cm], $\alpha_2 = 2$ [cm] and $\alpha_3 = 3$ [cm]. For example, for the first trial with $\alpha_1 = 1$, the first translation perturbation vector applied to the first virtual IMU sensor could be of the form $t_1 = 1 \cdot [1 \quad -1 \quad 1]^\top$ [cm]. Whereas the second translation perturbation vector applied to the second virtual sensor could be of the form $t_2 = 1 \cdot [-1 \quad 1 \quad -1]^\top$ [cm]. In this fashion, prior to each Monte Carlo simulation, initially six of these random translation perturbation vectors t were created. One for each virtual IMU, and applied to the virtual IMU's original position respectively.

After 100 Monte Carlo simulations, the means and standard deviations of the Root Mean Square Error (RMSE) values were calculated and are given in Table 4-3. For comparison, the RMSE means and standard deviations for 100 simulations of the idealized scenario with $\alpha = 0$ [cm] are also depicted in the first row of Table 4-3. In Figure 4-6, box plots corresponding to these three perturbed trial Monte Carlo simulations, are shown to visualize the RMSE values obtained.

From the results obtained, it can be concluded that translational errors do not notably affect the degradation of the reconstruction performance. From Table 4-3, the means of the RMSE values for all the joint angles q for the three trials conducted are still in the order of 10^{-1} [deg]. This could be explained as the IEKF-OS also tries to minimize the differences between the estimated and actual measured angular velocities. As long as the frames in which this minimization occurs, are properly aligned, the IEKF-OS is then still able to reconstruct the state due to this angular velocity information. Note that, for a given body, a translated but not rotated frame with respect to an original frame, will still measure the same angular velocity. This compared to the angular velocity as measured by this original frame and expressed in this original frame. Obviously, the angular velocity of a body expressed in this original frame is independent of the position of the frame from which these measurements are derived.

Thus it can be deduced that translational errors between modeled sensors do not really affect the degradation of the motion reconstruction. The next section will analyze the IEKF-OS

Table 4-3: The RMSE means and standard deviations obtained of 100 Monte Carlo simulations for each of the three trials with different translational perturbation values α. RMSE means and standard deviations are shown for each joint angle q and joint angular velocity u.

α	RMSE	Joint angle [deg]						Joint angular velocity [deg/s]					
[cm]		q_1	q_2	q_3	q_4	q_5	q_6	u_1	u_2	u_3	u_4	u_5	u_6
0	Mean	0.293	0.103	0.100	0.111	0.183	0.319	0.386	0.707	0.462	0.889	0.427	1.064
	σ	0.117	0.002	0.003	0.005	0.011	0.012	0.004	0.004	0.005	0.005	0.003	0.004
1	Mean	0.261	0.163	0.100	0.168	0.142	0.267	0.418	0.728	0.524	0.933	0.499	1.092
	σ	0.226	0.190	0.064	0.157	0.070	0.094	0.047	0.156	0.110	0.164	0.084	0.052
2	Mean	0.298	0.169	0.131	0.208	0.169	0.375	0.461	0.795	0.599	1.066	0.644	1.184
	σ	0.264	0.099	0.047	0.084	0.091	0.096	0.067	0.244	0.090	0.268	0.094	0.122
3	Mean	0.370	0.239	0.187	0.294	0.240	0.503	0.557	0.946	0.753	1.311	0.825	1.282
	σ	0.459	0.215	0.151	0.128	0.133	0.129	0.107	0.357	0.137	0.368	0.158	0.148

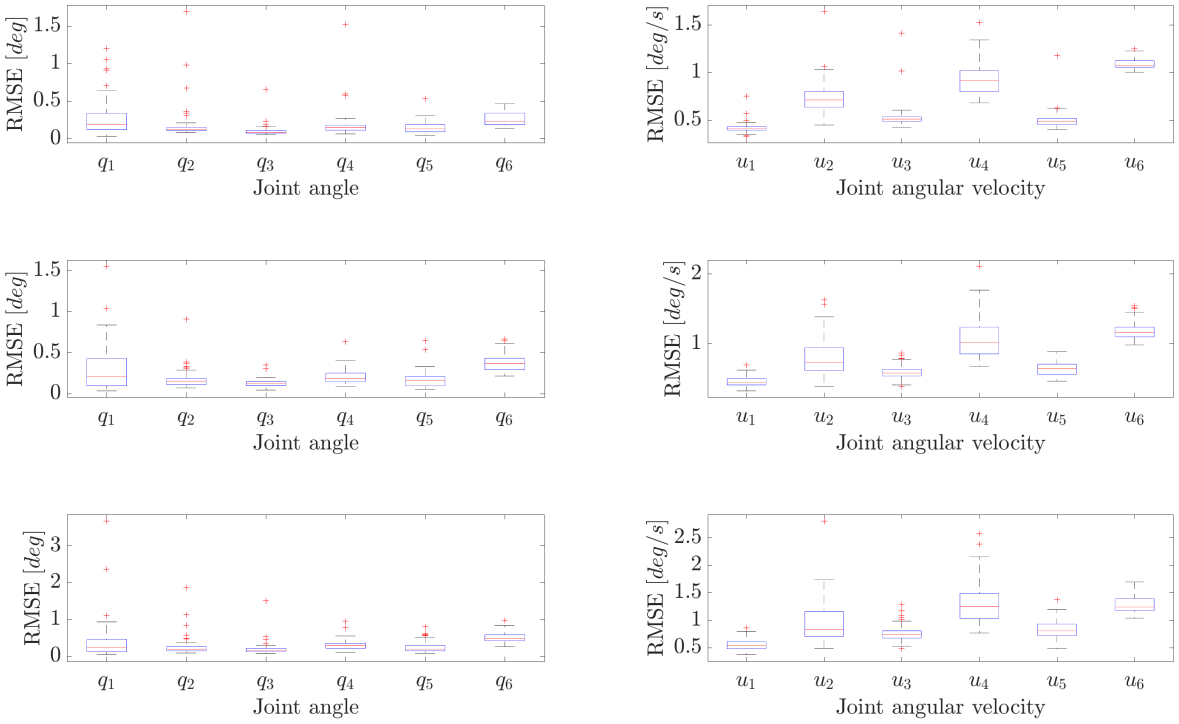


Figure 4-6: Upper row of box plots corresponds to trial 1 with $\alpha_1 = 1$ [cm]. Middle column of box plots corresponds to trial 2 with $\alpha_2 = 2$ [cm]. Last row of box plots corresponds to trial 3 with $\alpha_3 = 3$ [cm].

reconstruction performance under the presence of rotational errors. These rotational errors are introduced at the virtual IMUs modeled on the system creating the measurements. Again, the virtual IMUs modeled on the system of the IEKF-OS algorithm will be left unchanged. It is assumed that these imposed rotational errors will affect the degradation to a greater extent compared to translational errors. Mainly, as re-orienting these virtual IMU frames, will as similar to translational errors affecting accelerometer measurements, moreover, also affect the angular velocity measurements. This for the reason that now both the angular velocity and the linear acceleration measurements are expressed in this re-oriented frame, differing from

the original frame. How much this degrades performance is quantified in Section 4-5-2 using Monte Carlo simulations again.

4-5-2 Incorporating rotational errors

The orientation error will mainly depend on how accurately the orientation of the experimental Xsens IMU can be estimated. This error is defined between the experimental Xsens IMU on the actual system and the modeled virtual IMU on the corresponding OpenSim model. As previously stated, the orientation of the Xsens IMU will be obtained from the XKF3hm algorithm. From Xsens' orientation performance specifications, as shown in Table G-2, it can be seen that the maximal RMS orientation error is equivalent to 1.5 degrees in one axis [48]. This orientation error will be further denoted as $X_{RMS} = 1.5$ [deg]. Again, prior to each Monte Carlo simulation, each time six different r_i perturbation vectors will be created. The i-th rotational perturbation vector will be applied in the XYZ order to the i-th original virtual IMU orientation. The original orientation here is defined in Table D-3. This r_i vector will be of the form

$$r_i = \beta \cdot \begin{pmatrix} \pm r_x \\ \pm r_y \\ \pm r_z \end{pmatrix} \text{[deg]}, \tag{4-3}$$

where β is the rotational perturbation value now.

Three trials were performed, where again, each trial corresponds to 100 Monte Carlo simulations. The three rotational perturbations values, β_i, were set at the following fractions of X_{RMS} being 0.5, 1 and 1.5. This yields the following values: $\beta_1 = 0.5 \cdot X_{RMS} = 0.75$ [deg], $\beta_2 = 1 \cdot X_{RMS} = 1.5$ [deg] and $\beta_3 = 1.5 \cdot X_{RMS} = 2.25$ [deg]. As an example of these modeled offsets, consider for instance trial 3. The first rotational perturbation vector could be of the form $r_1 = 2.25 \cdot [-1 \;\; 1 \;\; 1]^\top$ [deg]. Whereas the second rotational perturbation vector could be of the form $r_2 = 2.25 \cdot [1 \;\; -1 \;\; -1]^\top$ [deg]. Six of these different rotational perturbation vectors r_i were created for the six virtual IMUs on the system creating the measurements. These vectors were re-computed each time prior to each Monte Carlo simulation.

Like Section 4-5-1, after 100 Monte Carlo simulations, the RMSE means and standard deviations for each trial were determined. The values obtained are depicted in Table 4-4. To facilitate comparison, in the first row of Table 4-4, the RMSE means and standard deviations for 100 simulations of the idealized scenario with $\beta = 0$ [cm] are also presented. To visualize the RMSE values, in Figure 4-7, box plots corresponding to these three perturbed trial Monte Carlo simulations are shown.

Compared to the translational perturbations as described in the previous Section 4-5-1, it can be inferred that rotational offsets impact the degradation more significantly. This was already presumed. As can be seen, there is one order of magnitude between RMSE values for the translational perturbed scenario and the RMSE values for the rotational perturbed scenario. The performance degradation especially becomes evident for the scenario with $\beta_3 = 2.25$ [deg], where a maximum for the mean value of joint angle q_6 is observed. However, from the RMSE values, it can not be concluded uniformly that segments located further away from the root of the chain are estimated worse compared to the closer linked rigid bodies.

The definition of what error is acceptable is obviously dependent on the type of motion of the system and its particular application, e.g., possibly clinical or sport settings. However,

Table 4-4: The RMSE means and standard deviations obtained of 100 Monte Carlo simulations for each of the three trials with various rotational perturbation values β. RMSE means and standard deviations are shown for each joint angle q and joint angular velocity u.

β	RMSE	Joint angle [deg]						Joint angular velocity [deg/s]					
[deg]		q_1	q_2	q_3	q_4	q_5	q_6	u_1	u_2	u_3	u_4	u_5	u_6
0	Mean	0.293	0.103	0.100	0.111	0.183	0.319	0.386	0.707	0.462	0.889	0.427	1.064
	σ	0.117	0.002	0.003	0.005	0.011	0.012	0.004	0.004	0.005	0.005	0.003	0.004
$0.5{\cdot}X_{RMS}$	Mean	0.440	0.530	0.609	0.745	0.664	0.892	0.629	0.944	0.923	1.246	1.120	1.553
	σ	0.233	0.238	0.154	0.351	0.310	0.410	0.111	0.271	0.268	0.304	0.279	0.313
$1{\cdot}X_{RMS}$	Mean	0.736	1.093	1.198	1.437	1.211	1.721	1.028	1.415	1.538	2.054	2.095	2.697
	σ	0.355	0.438	0.319	0.660	0.499	0.753	0.245	0.454	0.408	0.707	0.600	0.868
$1.5{\cdot}X_{RMS}$	Mean	1.194	1.557	1.808	2.184	1.875	2.589	1.529	1.982	2.494	2.860	3.159	3.613
	σ	0.516	0.725	0.467	1.041	0.934	1.170	0.336	0.643	0.669	1.015	0.874	1.300

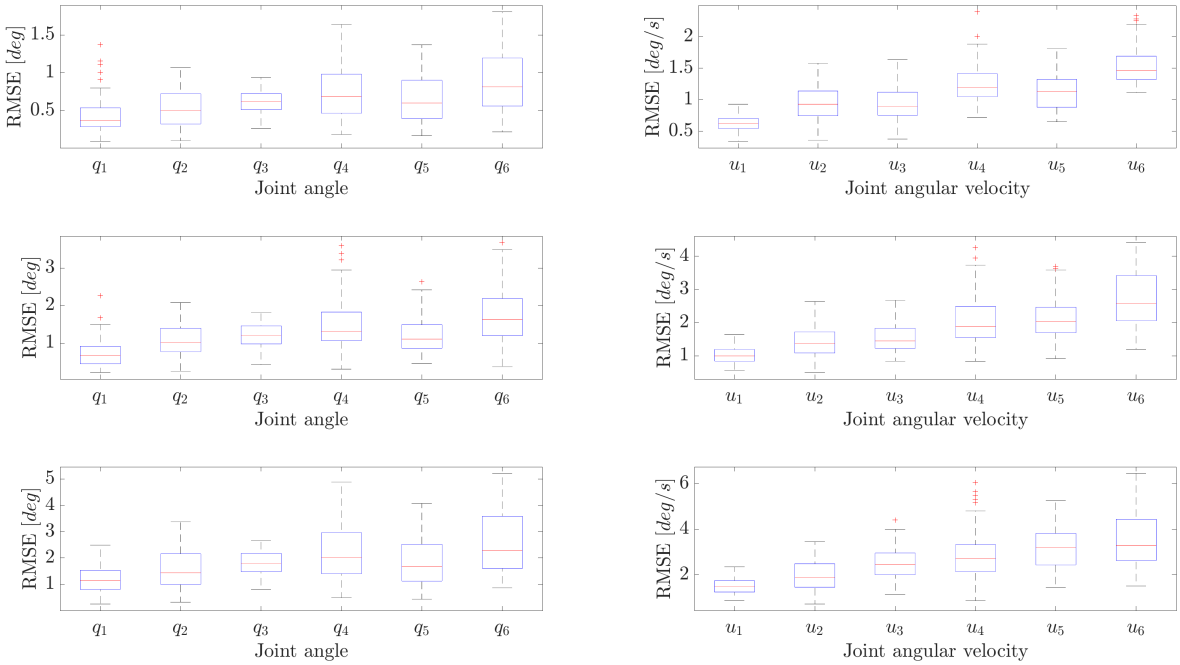


Figure 4-7: Upper row of box plots corresponds to trial 1 with $\beta_1 = 0.75$ [deg]. Middle column of box plots corresponds to trial 2 with $\beta_2 = 1.5$ [deg]. Last row of box plots corresponds to trial 3 with $\beta_3 = 2.25$ [deg].

to approach clinically acceptable errors, RMSE joint angle values of ≤ 5 [deg] should be strived [49]. Hence, it can be noticed that the RMSE values are still acceptable for motion reconstruction, as all the joint angle RMSE values are below this 5 [deg]. On the other hand, for larger orientation perturbation values, this IEKF-OS algorithm will likely exceed this joint angle RMSE of 5 [deg]. On that note, the next section will increase these rotational perturbation values to a maximal value of $5 \cdot X_{RMS} = 7.5$ [deg]. Yet, as the aim here was to perform a sensitivity analysis to sensor placement errors, it has to be shown from actual experiments whether the algorithm produces joint angle estimates lower than this 5 [deg] error. Moreover, in comparison with [6] and [17], which reported mean RMSE joint angle errors between 4-8 [deg], it can be stated that a similar performance could be obtained for this

robotic system. This, of course, depends if indeed the orientation error achieved, is around this X_{RMS} value as specified by Xsens [48]. This claim, obviously still has to be justified. To that aim, the IEKF-OS algorithm will be validated against ground truth joint encoder measurements from actual experiments. To mimic a more realistic scenario, Section 4-5-3 will introduce both translational as well as rotational errors.

4-5-3 Incorporating translational and rotational errors

The motion reconstruction performance will now be assessed when both translational and rotational errors are applied to the virtual sensors creating the measurements. The first three trials that will be analyzed are combinations of the previously specified trials. Again, each trial corresponds to 100 Monte Carlo simulations. As such, trial 1 will be the combination of introducing a translational offset first with $\alpha_1 = 1$ [cm] after which a rotational perturbation of $\beta_1 = 0.75$ [deg] will be applied. Hence as there are six virtual sensors to be perturbed, prior to each Monte Carlo simulation, first each i-th virtual IMU will be given an translation offset of form $t_1 = 1 \cdot [\pm 1 \quad \pm 1 \quad \pm 1]^\top$ [cm] after which its orientation will be perturbed in an XYZ-order with the perturbation vector $r_1 = 0.75 \cdot [\pm 1 \quad \pm 1 \quad \pm 1]^\top$ [deg]. Similarly, the other trials will be performed where trial 2 corresponds to $\alpha_2 = 2$ [cm] and $\beta_2 = 1.50$ [deg], and trial 3 corresponds to $\alpha_3 = 3$ [cm] and $\beta_3 = 2.25$ [deg].

Additional offset vectors were created to check when the IEKF-OS performance really shows to degrade. The maximal translational offsets were kept at a value of 3 cm. This as it is very unlikely that manually measuring will lead to a larger translational error of 3 cm. Moreover, as rotational perturbations really affect performance degradation, this value was further increased by multiples of Xsens' specified RMS $X_{RMS} = 1.5$ [deg]. These multiples were chosen to be $\alpha_4 = 3 \cdot X_{RMS} = 4.5$ [deg], $\alpha_5 = 4 \cdot X_{RMS} = 6.0$ [deg] and $\alpha_6 = 5 \cdot X_{RMS} = 7.5$ [deg].

The means and standard deviations of the RMSE values for the joint angles q and joint angular velocities u for the three trials are depicted in Table 4-5. To facilitate visualization of these results to the reader, box plots as shown in Figure 4-8 and Figure 4-9 are included. Besides, to evaluate the degradation of the IEKF-OS performance compared to the scenario with no translational and rotational perturbations, the RMSE means and standard deviations for 100 simulations are presented in the first row of Table 4-5 again.

Comparing the overall performance of the IEKF-OS algorithm under the presence of translational and rotational errors to the performance of the algorithm under the presence of rotational errors only, it can be said that comparable RMSE values are obtained. The addition of the translational errors to the already present rotational offsets hence only minimally degrade performance. This as the joint angle reconstruction performance is degraded by a maximal amount of 0.2 [deg] of a given joint angle. This observation can be made when one compares the RMSE means of the joint angles q in Table 4-4 and those in Table 4-5. With this insight, it can be deduced that the main degradation of the algorithm's performance is due to the rotational offsets. This can be explained as re-oriented sensor frames both affect the linear acceleration and the angular velocity measurements. This due to the fact that these measurements are now expressed in a different frame.

Looking at the results of Table 4-5 and the box plots in Figure 4-9, real tracking performance degradation can be observed for rotational errors induced in the order of $\beta_5 = 4 \cdot X_{RMS}$ and

Table 4-5: The RMSE means and standard deviations obtained of 100 Monte Carlo simulations for various combinations of translational α and rotational β perturbation values. RMSE means and standard deviations are shown for each joint angle q and joint angular velocity u.

α	β	RMSE	Joint angle [deg]						Joint angular velocity [deg/s]					
[cm]	[deg]		q_1	q_2	q_3	q_4	q_5	q_6	u_1	u_2	u_3	u_4	u_5	u_6
0	0	Mean	0.293	0.103	0.100	0.111	0.183	0.319	0.386	0.707	0.462	0.889	0.427	1.064
		σ	0.117	0.002	0.003	0.005	0.011	0.012	0.004	0.004	0.005	0.005	0.003	0.004
1	$0.5X_{RMS}$	Mean	0.419	0.530	0.589	0.716	0.663	0.973	0.625	0.965	0.863	1.252	1.169	1.586
		σ	0.237	0.260	0.182	0.317	0.307	0.420	0.127	0.309	0.220	0.313	0.304	0.362
2	X_{RMS}	Mean	0.856	1.017	1.207	1.522	1.258	1.984	1.121	1.381	1.694	2.097	2.108	2.586
		σ	0.367	0.513	0.276	0.800	0.518	0.932	0.299	0.518	0.407	0.674	0.602	0.822
3	$1.5X_{RMS}$	Mean	1.217	1.462	1.819	2.014	2.030	2.674	1.612	1.978	2.411	2.687	3.195	3.519
		σ	0.611	0.705	0.496	1.078	0.861	1.377	0.361	0.692	0.658	0.910	0.979	1.282
3	$3X_{RMS}$	Mean	2.513	3.024	3.786	4.414	4.263	4.946	3.154	3.821	4.980	5.603	6.353	6.505
		σ	1.170	1.358	1.093	1.971	1.723	2.686	0.857	1.516	1.215	2.365	2.045	2.741
3	$4X_{RMS}$	Mean	2.911	4.048	4.803	5.545	5.402	7.111	3.971	5.306	6.274	7.343	7.935	8.351
		σ	1.375	1.737	1.500	2.552	2.180	3.730	1.066	1.755	1.701	2.711	2.233	3.363
3	$5X_{RMS}$	Mean	3.771	5.024	6.243	7.642	6.697	9.701	4.904	6.415	7.583	9.439	10.089	10.965
		σ	1.857	2.438	1.574	3.300	2.761	4.852	1.198	2.290	2.085	3.647	3.458	4.881

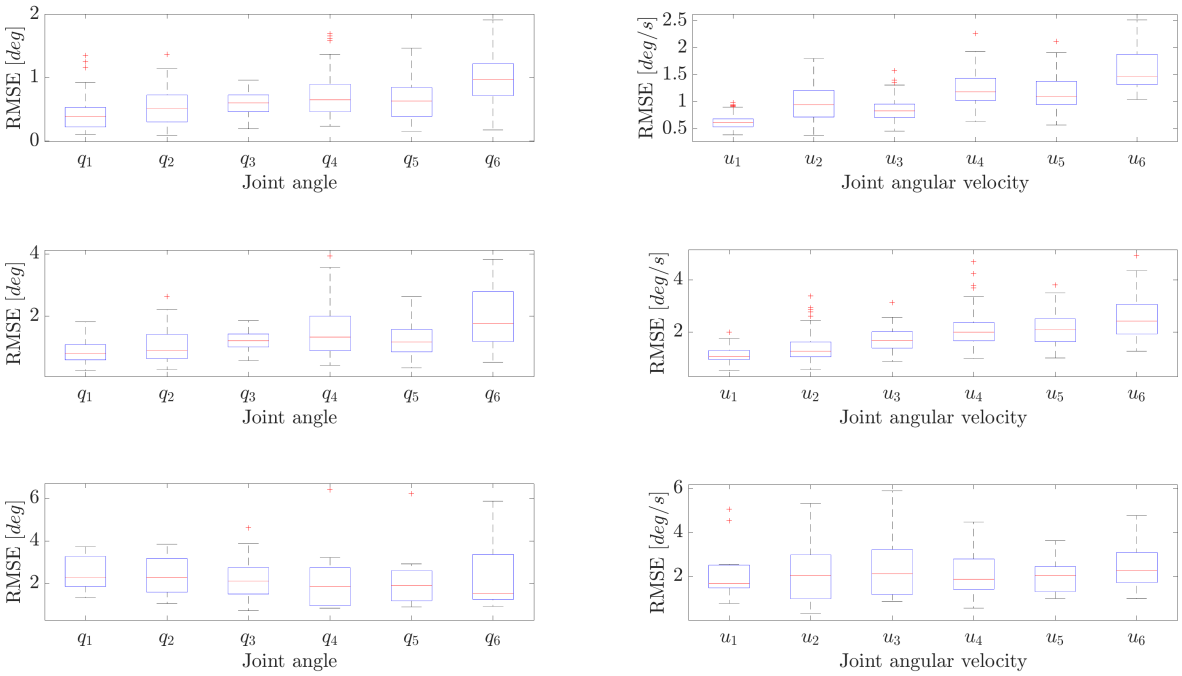


Figure 4-8: Upper row of box plots corresponds to trial 1 with $\alpha_1 = 1$ [cm] and $\beta_1 = 0.5 \cdot X_{RMS} = 0.75$ [deg]. Middle column of box plots corresponds to trial 2 with $\alpha_2 = 2$ [cm] and $\beta_2 = X_{RMS} = 1.5$ [deg]. Last row of box plots corresponds to trial 3 with $\alpha_3 = 3$ [cm] and $\beta_3 = 1.5 \cdot X_{RMS} = 2.25$ [deg].

$\beta_6 = 5 \cdot X_{RMS}$. As one can derive from these results, the joint angle RMSE values are then approaching 7-10 [deg]. Future users aspiring to use this algorithm should therefore, need to orient the sensors carefully with a maximal orientation error of $3 \cdot X_{RMS} = 4.5$ [deg] in each axis. Note that these results only hold for this system and cannot be translated directly to other systems such as human bodies. When applied to human motion capture, one also needs

to take into account the soft tissue artifacts (STAs). These disturbances perturb the IMU measurements even more.

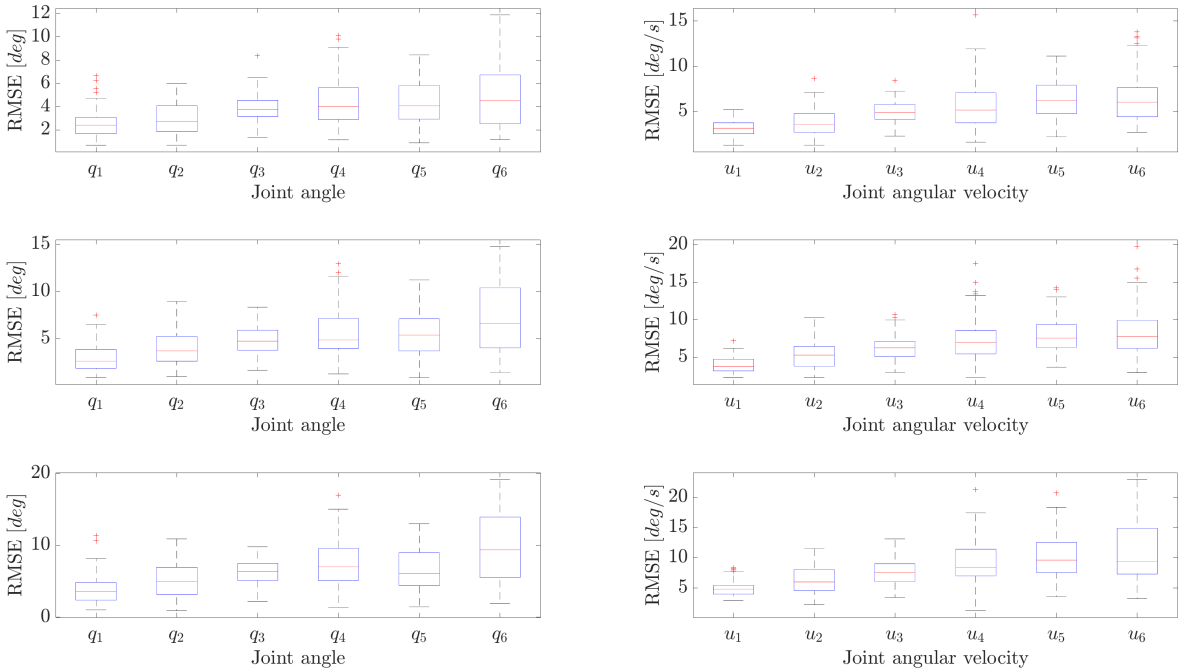


Figure 4-9: Upper row of box plots corresponds to trial 4 with $\alpha_4 = 3$ [cm] and $\beta_4 = 3 \cdot X_{RMS} = 4.50$ [deg]. Middle column of box plots corresponds to trial 5 with $\alpha_5 = 3$ [cm] and $\beta_5 = 4 \cdot X_{RMS} = 6.00$ [deg]. Last row of box plots corresponds to trial 6 with $\alpha_6 = 3$ [cm] and $\beta_6 = 5 \cdot X_{RMS} = 7.50$ [deg].

4-6 Conclusions

In this chapter, the IEKF-OS algorithm's performance was tested on a 7 DoF robot manipulator which was allowed to move in a 3D space. For this system, numerical simulations have been performed to assess the novel algorithm's motion reconstruction performance.

The algorithm has been shown to work for a simulation on a robotic manipulator which is composed of multiple linked rigid bodies. With the results presented, the door is opened for future users who want to extend this algorithm to human motion reconstruction. This later desire can be accomplished when the user has an accurate OpenSim biomechanical model, tuned to the participant's body. The algorithm developed then utilizes the kinematical and dynamical information to reconstruct the original motion.

With the Monte Carlo simulations presented, the IEKF-OS algorithm reconstructed the motions of the six links of the robot even under the presence of common sensor placement errors. For moderate sensor placement errors, $\alpha = 3$ [cm] and $\beta = 1.5 \cdot X_{RMS} = 2.25$ [deg], all the RMSE joint angles showed to be below 5 [deg] for this motion. Next to that, it was observed that the IEKF-OS motion reconstruction accuracy is mainly affected by rotational errors introduced. The IEKF-OS estimations made, showed clear degradation when rotational errors of 7.5 [deg] around each virtual sensor's axis were imposed. For that scenario, mean RMSE joint angle and joint angular velocities values in the ranges of 7-10 [deg] and 5-11 [deg/s] were observed, respectively.

Chapter 5

Experiment-based validation of the motion reconstruction algorithm

The motion tracking algorithm will now be validated by performing experiments on the physical KUKA LBR iiwa 7 R800 robot manipulator. The Xsens MTw Awinda IMUs will be attached to the links of this robot manipulator from which experimental IMU data can be obtained. Moreover, the IEKF-OS motion reconstruction accuracy will be compared to OpenSense's motion tracking accuracy. OpenSense does not take into account the system dynamics, but solely incorporates the system's kinematic constraints. Lastly, the chapter is concluded with some final remarks regarding the IEKF-OS and OpenSense comparison.

5-1 Tests performed on the KUKA robot manipulator

The motion reconstruction accuracy performance of the devised IEKF-OS will be validated on the KUKA LBR iiwa 7 R800 robot. From the numerical simulations performed in Chapter 4, it was observed that the algorithm developed, was able to reconstruct the original motion. It even reconstructs the original motion in the presence of common sensor placement errors. From the results obtained, it was observed that imposed rotational errors between modeled virtual sensors, mainly degrade reconstruction accuracy.

For the experiments conducted in this chapter, it is thus key to model the virtual IMUs as closely as possible to their corresponding experimental IMUs. The virtual IMUs will again be modeled as orthogonal XYZ-frames on each link of this OpenSim model. Accurate orientation estimates can be obtained for these experimental IMUs using the Xsens Kalman Filter for 3D human motion (XKF3hm) [11]. This filter is provided with the Xsens IMUs used. The orientation estimates obtained from XKF3hm, provide information on how each experimental IMU is oriented on the KUKA's link. With this information, the corresponding virtual IMUs can be modeled on the respective links of the OpenSim KUKA model. For interested readers, this sensor-to-segment calibration is extensively outlined in Appendix F-1.

Positions of the experimental IMUs were manually measured using a tape measure. From the previous numerical simulations, it was shown that translational offsets do not degrade performance. Therefore, it was assumed that manually measuring yields sufficient position estimates. In short, the positions were measured from each KUKA link's joint center to the left down corner of the attached IMU. From this left down corner, the distances are known to the accelerometer position in the Xsens MTw IMU case. The location of the accelerometer is shown in Figure F-2. For interested readers, this procedure is also outlined in Appendix F-1. Future work could look into more advanced methods to estimate the sensor orientations and locations on the body.

It must be noted that only six Xsens MTw Awinda IMUs were available for the experiments. The KUKA robot, however, has seven links permitting seven DoFs. Due to this shortcoming of MTw Awinda IMUs, the preference was given to reconstruct the motions of the first six links. The end-effector, the last link shown in Figure 5-1, is allowed to rotate relative to the sixth link. The rotary motion of the last link was thus not reconstructed. For that reason, throughout this chapter, only joint angle evolutions of the first six links will be presented.

Performed tests

Three different analyses were performed to evaluate the motion tracking accuracy performance of the IEKF-OS algorithm. These analyses were all carried out on the KUKA robot manipulator and are presented below.

1. Experimental validation of the IEKF-OS algorithm for various ranges of motion. Joint angle and joint angular velocity RMSE values are computed between IEKF-OS estimations and the robot's ground truth joint encoder values. The robot's joint angular velocities were approximated from the joint encoder values using a forward-difference scheme.

2. Comparison of IEKF-OS estimated joint angles with respect to OpenSense estimated joint angles. For OpenSense, Xsens's XKF3hm algorithm [11] is used to estimate sensor orientations from experimentally obtained gyroscope, accelerometer and magnetometer data. The analysis is performed for the same motions as for the IEKF-OS experimental validation analysis.

3. Comparison of IEKF-OS estimated joint angles with respect to OpenSense estimated joint angles. For OpenSense, Madgwicks's algorithm [26] is used to estimate sensor orientations from experimentally obtained gyroscope and accelerometer data and OpenSim synthesized magnetometer data. The analysis is performed for the same motions as for the IEKF-OS experimental validation analysis.

For all these tests, the same KUKA OpenSim model, as presented in Chapter 4, will be used. As outlined in Section 2-5, OpenSense is an inverse kinematics method. This method computes the set of joint angles at each time instant that minimizes the error between the virtual IMU frame orientations and the experimental IMU sensor orientations. IMU sensor orientation estimates can be obtained from various orientation estimation algorithms, for instance [10], [11], [26]. It must be noted that OpenSense highly relies on the method used

that estimates the orientations of the experimental sensors from IMU data. If inaccurate sensor orientation estimates are computed, the motion reconstruction accuracy of OpenSense will deteriorate.

The choice for comparing the IEKF-OS algorithm against OpenSense was made for three reasons. Firstly, OpenSense is the current method available to the OpenSim community for reconstructing motions. Therefore, it is interesting to compare the IEKF-OS method against it. Secondly, both methods rely on the same OpenSim model and lastly, they use the same oriented virtual IMUs. As both methods will use the exact same initialized model with the same placed virtual sensors, and use the same experimentally obtained IMU data, this allows for a fair comparison. The difference, however, is that OpenSense only incorporates the kinematic constraints of the OpenSim model, e.g. the type of the joint connecting two segments. The IEKF-OS algorithm next to that also includes the system dynamics.

5-2 Validation experiments

Initially, to mimic arbitrary motions, the idea was to have the end-effector track the infinity shaped trajectory as designed in Chapter 4. Unfortunately, errors occurred when trying to compile this code for the KUKA robot. As such, the choice was made to apply sine waves as input to each joint motor. These inputs would allow to excite each joint individually at various frequencies and amplitudes.

Desired motion specification

The intention of the experiments was to increase the frequency of the sine waves applied to the joint motors between trials. In this way, it can be observed whether motion reconstruction accuracy is preserved for varying motion excitations. Four trials were conducted. For the first trial, a sine wave with a frequency of 0.025 [Hz] and amplitude of 1.2 [rad] was applied to each joint. For subsequent trials, the frequencies of the sine waves applied to each joint were increased in steps of 0.025 [Hz]. As the KUKA is a collaborative robot, it is mainly meant to work with humans. This resulted that for the two trials with applied sine waves of frequencies 0.075 [Hz] and 0.1 [Hz], the robot stopped moving. This is likely due to joint velocity limits for safe human-robot interaction that were exceeded. For that reason, for these two frequencies, the amplitudes had to be decreased to 0.7 [rad].

Calibration procedure

As outlined in Section 3-2-3, both the IMU's gyroscope and the accelerometer components need to be calibrated. For interested readers, a practical workflow of this procedure is included in Appendix F. After calibrating the gyroscope and accelerometer, the sensor-to-segment calibration needs to be performed. It should be noted, that this sensor-to-segment calibration is a crucial step. Misalignment between the orientation of the experimental Xsens IMU and the orientation of the virtual IMU in OpenSim will lead to inaccurate motion reconstruction. As was noted in Section 4-5-3, for orientation errors in the order of 7.5 [deg] around each X-Y-Z axis, joint angle RMSE values approaching 10 [deg] were observed. However, it is expected that orientation errors as specified by Xsens in the order of 1.5 [deg] will occur. For

that scenario with additional translational errors, Table 4-5 in Section 4-5 shows that joint angle RMSE values in the range of 1-3 [deg] are likely to be obtained.

For the sensor-to-segment calibration, it was required that the system is configured in its fully extended upward position. This entails that all the generalized coordinates q should be equal to 0 [deg]. This requirement was validated by reading of the values of the joint encoders. To configure the KUKA robot in this fully extended position, the joints were prescribed using the KUKA interface towards 0 [deg]. Eventually, each joint had a maximal angle error of 0.1 [deg].

Xsens Velcro straps were then positioned on each of the KUKA's links. These straps allow for easy IMU attachment on the links of the KUKA robot. The sensor-to-segment calibration, as outlined in Appendix F-1, was performed prior to the actual experiments. Each Xsens IMU was attached as follows. The IMU's X-axis faced towards the sky, its Z-axis points away from the center of the KUKA's link, and then automatically the IMU's Y-axis points towards the left. The creation of the corresponding virtual IMUs on the KUKA OpenSim model will be further detailed in Appendix D-2 and Appendix F-1. The six Xsens IMUs attached to the KUKA robot's first six links are shown in Figure 5-1.

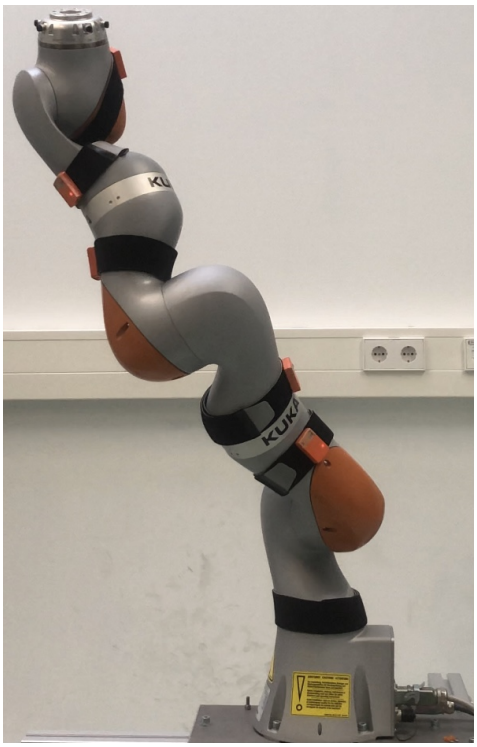


Figure 5-1: The KUKA LBR iiwa 7 R800 robot with the six Xsens MTw Awinda IMUs.

Initialization of the IEKF-OS algorithm

After the experiment, the data logged from the attached Xsens IMUs and the KUKA robot's encoders was processed. Similar to Chapter 4, the variances σ_q^2 and σ_u^2 of the process noises for the six generalized coordinates q, and the six generalized velocities u, were set at 0 [rad]

and 0 [rad/s] respectively. Hence, $Q_q = \mathcal{O}_6$ [rad] and $Q_u = \mathcal{O}_6$ [rad/s], where $\mathcal{O}_6$ denotes a matrix with zeros of size 6×6. The variances of the process noises for the six joint torques τ, were all set to $\sigma_\tau^2 = 1$ [Nm], yielding $Q_\tau = 1 \cdot \mathcal{I}_6$ [Nm]. Here, $\mathcal{I}_6$ denotes the 6×6 identity matrix. The high joint torque variance allows for large dynamic motion differences. The process covariance matrix Q in the IEKF-OS algorithm was formed with the Q_q, Q_u, and Q_τ process noise covariances on the diagonal. From the stationary inertial data logged prior to the experiments, for each i-th IMU, the covariance matrices of both the gyroscope $\Sigma_{\omega,i}$ and the accelerometer $\Sigma_{a,i}$ were determined. The measurement covariance matrix R was then formed as a block diagonal matrix with all these covariances as shown in (5-1). The measurement covariance R was set at $R = 150 \cdot R$. The multiplication factor of 150 was, from the simulations performed in Chapter 4, found to work best. For the experiments conducted, this can be partly explained. The noise in the inertial data obtained, was observed to increase at higher joint velocities. This can be seen in Figure 5-2. Increasing the measurement covariance matrix R

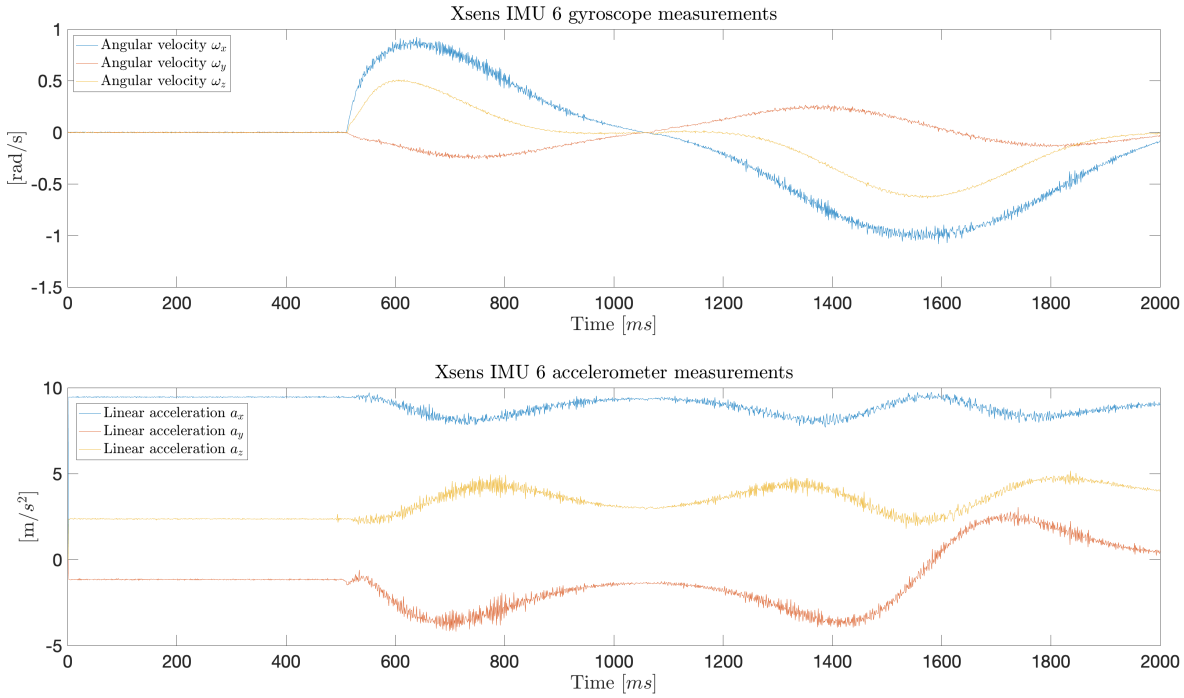


Figure 5-2: The gyroscope and accelerometer data for the first 20 seconds. It can be observed that the noise affecting the IMU measurements increases at higher velocities.

implies that the measurements should be weighed less. As such, this yields the following Q and R matrices

$$Q = \begin{pmatrix} Q_q & \dots & \mathcal{O}_6 \\ \vdots & Q_u & \vdots \\ \mathcal{O}_6 & \dots & Q_\tau \end{pmatrix}, \quad R = 150 \cdot \begin{pmatrix} \Sigma_{\omega,1} & \mathcal{O}_3 & \dots & \mathcal{O}_3 & \mathcal{O}_3 \\ \mathcal{O}_3 & \Sigma_{a,1} & \dots & \mathcal{O}_3 & \mathcal{O}_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathcal{O}_3 & \mathcal{O}_3 & \dots & \Sigma_{\omega,6} & \mathcal{O}_3 \\ \mathcal{O}_3 & \mathcal{O}_3 & \dots & \mathcal{O}_3 & \Sigma_{a,6} \end{pmatrix}. \tag{5-1}$$

Lastly, the number of measurement iterations ϵ, in the update step of the Kalman filter needs to be defined. More measurement update iterations allow for estimating the actual underlying

state value more accurately, when the estimates of the filter converge, for example, see [50]. The downside of choosing a lot of these iterations translates to an increase in computational cost and time. Like Section 4-4, the value was set at $\epsilon = 3$ as for more iterations, the performance gained, was negligible.

The initial joint encoder values were used to configure the joint angles of the KUKA OpenSim model accordingly. Note that, the IEKF-OS algorithm samples at the maximal frequency of the Xsens IMUs, being 100 [Hz]. The robot encoder measurements, however, are sampled at 200 [Hz]. For that reason, these encoder values were downsampled to 100 [Hz]. Time synchronization between the joint encoder values and the Xsens IMUs was performed manually. From the logged inertial data, the time instant was selected when the angular velocities of the sixth attached IMU started to increase. From this time instant, the IEKF-OS algorithm was initialized.

Results

After processing the inertial data through the IEKF-OS algorithm, joint angle $\hat{q}$ and joint angular velocities $\hat{u}$ estimations were obtained. These estimations were compared against the robot's ground truth joint angles q and joint angular velocities u. The joint angular velocities were approximated from the robot's joint angle encoder measurements using a forward-difference scheme. The results obtained for the four various experiments conducted are presented in Table 5-1. The RMSE values were computed after the first second such that the estimates of the IEKF-OS could converge. For conciseness, only the estimated states $\hat{q}$

Table 5-1: The joint angle q and joint angular velocities u RMSE results obtained for the trials.

Trial	Sine wave		Joint angle RMSE [deg]						Joint angular velocity RMSE [deg/s]					
	[Hz]	[rad]	q_1	q_2	q_3	q_4	q_5	q_6	u_1	u_2	u_3	u_4	u_5	u_6
1	0.025	1.2	0.966	0.370	1.908	0.750	1.770	1.492	0.375	0.299	0.404	0.551	0.301	0.912
2	0.05	1.2	1.053	0.448	1.971	0.874	1.204	1.565	0.565	0.476	0.527	0.680	0.477	1.555
3	0.075	0.7	2.220	0.322	2.181	0.781	2.789	1.426	0.595	0.756	0.609	1.000	0.565	1.416
4	0.1	0.7	0.846	0.486	2.066	0.998	0.955	1.509	0.715	0.589	0.649	0.952	0.498	1.936

and $\hat{u}$ of the six links of trial 2 are shown in Figure 5-3 and Figure 5-4. For the results of the other trials, the reader is referred to Appendix E-1. For comparison, the actual joint angles q and approximated joint angular velocities u from the joint encoders are shown in blue in these figures. From the RMSE results as shown in Table 5-1, it can be concluded that the IEKF-OS algorithm developed, is able to reconstruct the robot's movements for varying ranges of motion. The RMSE values for the joint angles q for these four trials are all within the range of 0.4 - 2.8 [deg]. This indicates that accurate joint angle reconstruction for these scenarios on the KUKA robot manipulator can be achieved using the IEKF-OS algorithm.

As previously stated, by observing the joint angular velocities closely, it can be seen that oscillations are present at higher angular velocities. This phenomenon can particularly be observed in Figure 5-4 for joint angular velocity u_6. Still, the IEKF-OS algorithm is able to estimate the joint angular velocities u with RMSE values within the range of 0.3 - 2 [deg/s].

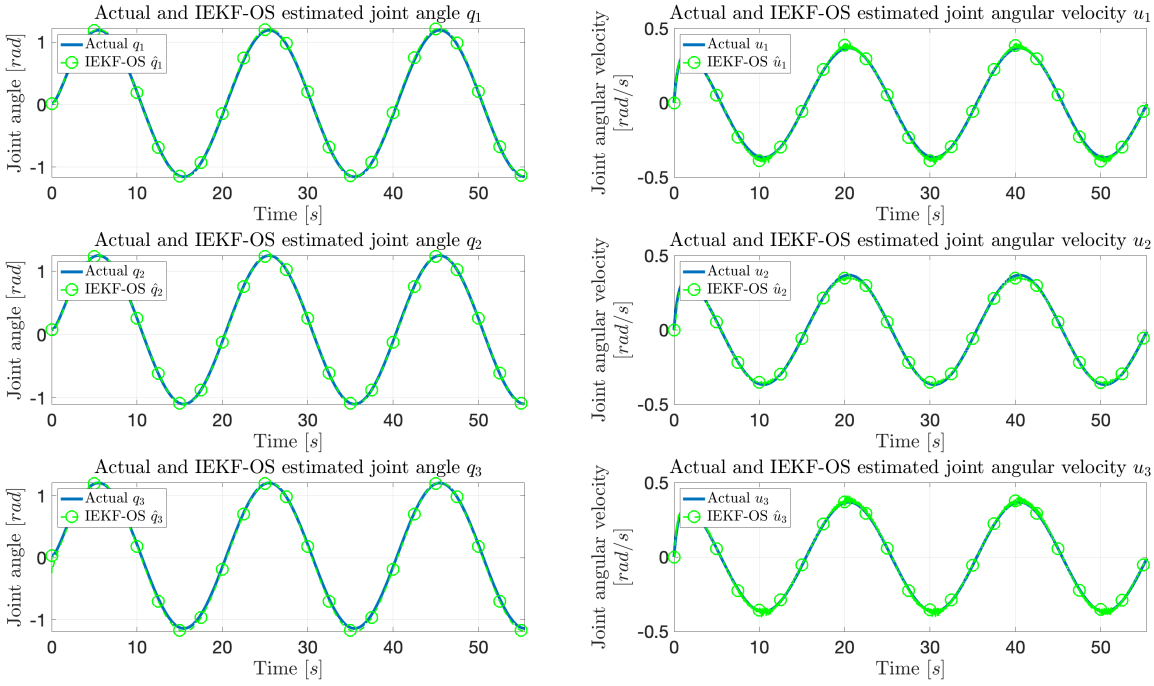


Figure 5-3: Trial 2 with sine waves applied to each joint with a frequency of 0.05 [Hz] and an amplitude of 1.2 [rad]. Depicted are the estimated and actual joint angles q_1, q_2, and q_3 (left column), and estimated and actual joint angular velocities u_1, u_2, and u_3 (right column).

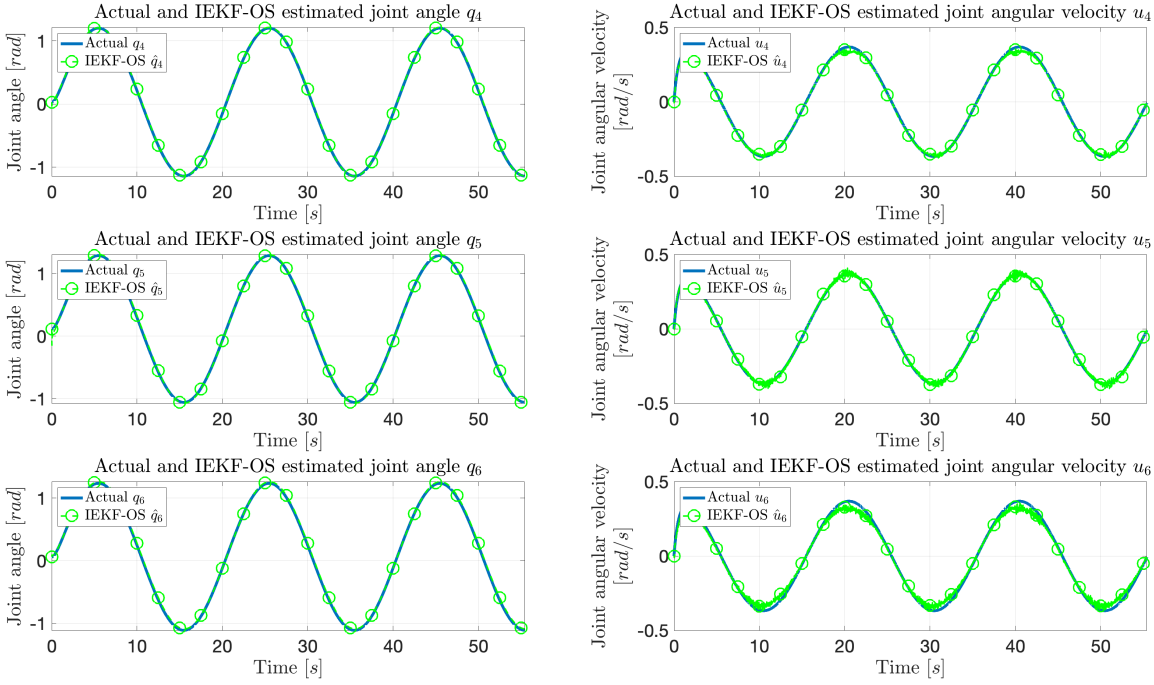


Figure 5-4: Trial 2 with sine waves applied to each joint with a frequency of 0.05 [Hz] and an amplitude of 1.2 [rad]. Depicted are the estimated and actual joint angles q_4, q_5, and q_6 (left column), and estimated and actual joint angular velocities u_4, u_5, and u_6 (right column).

5-3 Comparing accuracy of OpenSense-XKF3hm and IEKF-OS

With the IEKF-OS algorithm validated, it is interesting how the method developed compares to OpenSense. During the experiments, next to the raw angular velocity and raw linear acceleration, the orientation estimates from Xsens' XKF3hm algorithm were logged. OpenSense uses the orientation estimates of each attached Xsens IMU and with inverse kinematics, constrains these orientation estimates to the underlying OpenSim model. For OpenSense to work, accurate orientation estimates are needed.

Xsens' XKF3hm algorithm relies on three inertial components in the IMU. By integrating the angular velocity from the gyroscope, the sensor's orientation can be computed. Due to noises and biases corrupting the measurements made, the orientation estimates drift from the true orientation [10]. The second sensor, the accelerometer, reveals information on the direction of the gravity vector. With this accelerometer, information can be obtained about the roll angle (around the X-axis), and the pitch angle (around the Y-axis), to compensate for orientation drift in these two angles. A magnetometer is then usually included to compensate for the drift around the yaw angle, (around the Z-axis). This magnetometer measures the local magnetic field and gives information about this yaw angle. The local field measurements consist of both the magnetic field due to the presence of magnetic material and the earth magnetic field [10].

Therefore, to obtain accurate orientation estimations, one must make sure that the magnetometer measures the undistorted earth magnetic field. However, as these Xsens IMUs are attached to the links of the KUKA robot, consisting of ferromagnetic material, an orientation-dependent magnetic distortion is introduced [27]. As such, the magnetometer needs to be calibrated for the presence of this magnetic material [10]. Next to that, large currents applied to the joint motors also influence the local magnetic field [51]. For these reasons, Xsens' Magnetic Field Mapper calibration procedure [51] was used to calibrate the magnetometer. For correct magnetometer calibration, each IMU should capture as many orientations as possible, while being in a constant local magnetic field [32]. With Xsens' Magnetic Field Mapper calibration procedure, it was tried to calibrate the magnetometers of the six Xsens IMUs used. It must be noted that with the Xsens IMUs attached, not all orientations can be captured. This then resulted in a failure to accurately calibrate the magnetometers.

Still, the OpenSense method was compared against the IEKF-OS algorithm. The results for trial 1 and trial 4 are presented in Figure 5-5 and Figure 5-6. As the estimates of OpenSense are not reliable, no RMSE values have been computed. From Figure 5-5, it can be observed that for applied sine waves with large amplitudes, the results of OpenSense start to deteriorate. This can especially be observed for joint angles q_4, q_5 and q_6 in Figure 5-6. This is likely due to two reasons. Firstly, larger applied currents to the motors disturb the local magnetic field more. Secondly, for these large ranges of motion of 1.2 [rad], the sixth link came in the neighborhood of other located magnetic material, such as desks and computers. These additional disturbances presumably have affected the magnetometer readings more. For smaller ranges of motion, as shown in Figure 5-5, OpenSense estimates approach the ground truth estimates better. Still, for that scenario as well, no conclusions with respect to comparing OpenSense and the IEKF-OS algorithm can be drawn. Again for the reason that the magnetometer is not calibrated.

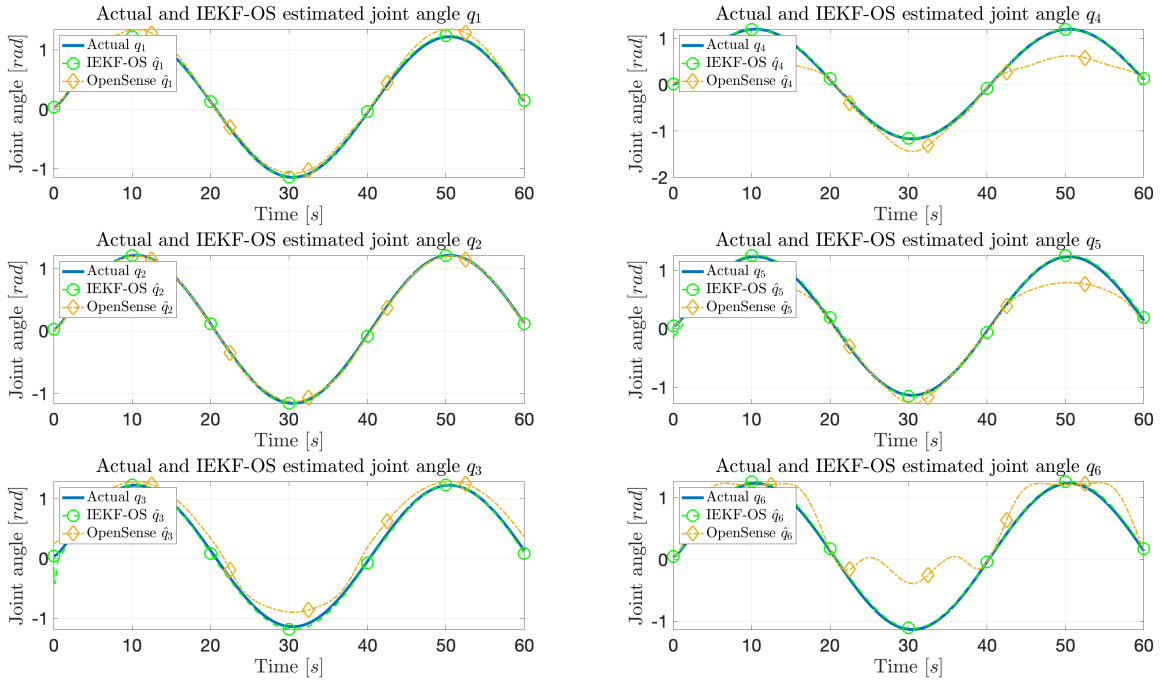


Figure 5-5: Trial 1 with sine waves applied to each joint with a frequency of 0.025 [Hz] and an amplitude of 1.2 [rad]. Comparison between actual joint encoders values shown in blue, IEKF-OS joint angle estimations shown in green, and OpenSense joint angle estimations shown in orange.

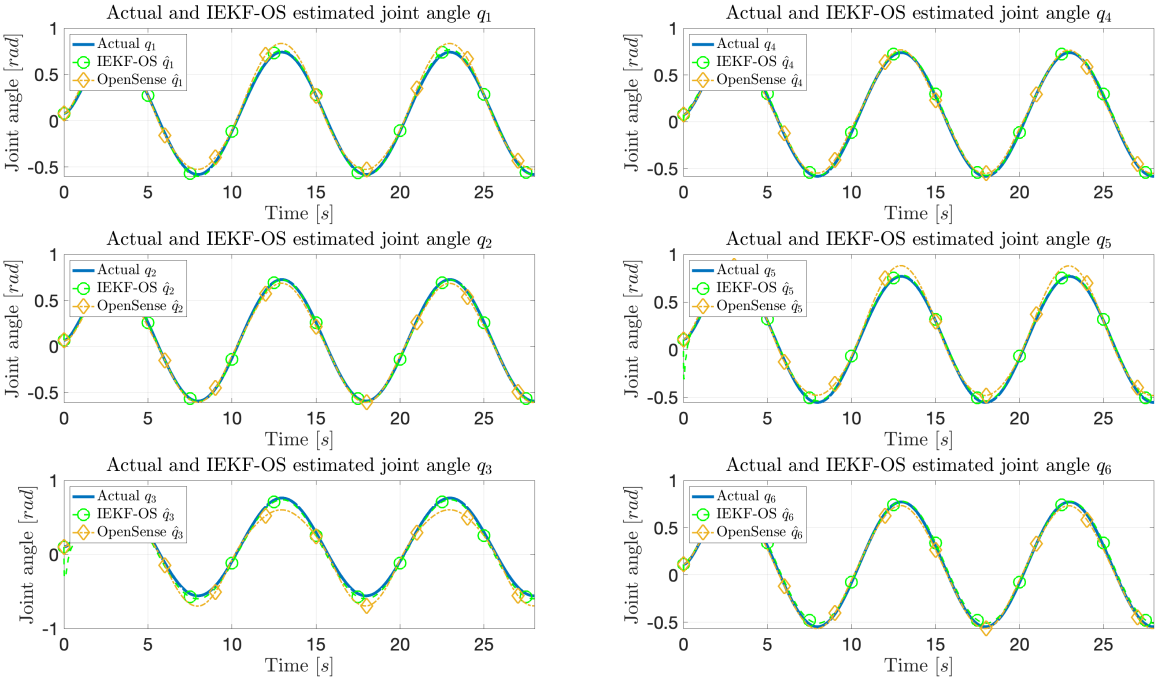


Figure 5-6: Trial 4 with sine waves applied to each joint with a frequency of 0.1 [Hz] and an amplitude of 0.7 [rad]. Comparison between actual joint encoders values shown in blue, IEKF-OS joint angle estimations shown in green, and OpenSense joint angle estimations shown in orange.

Initially, the aim was to compare the motion reconstructions obtained from both the IEKF-OS and OpenSense methods for experimentally obtained inertial data. As the magnetometer was not calibrated, from these results it cannot be concluded whether the IEKF-OS method developed performs better compared to the method of OpenSense. Yet, it can be stated that the IEKF-OS method performs motion reconstruction independent of the environment and under the presence of magnetic disturbances. This is, however, expected as the IEKF-OS algorithm does not rely on magnetometer data. For the scenario where magnetic material is present, OpenSense will likely lead to these strange observed results. This only is the case when these magnetic disturbances are unaccounted for.

5-4 Comparing accuracy of OpenSense-Madgwick and IEKF-OS

Due to the inaccurately calibrated magnetometer, the XKF3hm algorithm could not accurately estimate the IMU orientations as the magnetometer measurements were distorted. As a result, the joint angle estimates computed by OpenSense are not valid for comparison with the IEKF-OS estimated joint angles. For that reason, an additional analysis was performed. Ideal magnetometer data was created in simulation without any bias or noise. This magnetometer data was created as follows. From the experimental KUKA robot manipulator, the joint encoder values of each trial were available. These ground truth joint angles were then prescribed to the six joint angles of the KUKA OpenSim model to create the same motion in simulation. With the KUKA OpenSim model and the virtual IMU frames in place, a forward kinematics simulation was run. In the OpenSim ground frame, the artificial magnetic field was aligned with the ground frame's X-axis as $m^G = (X, Y, Z) = (1, 0, 0)$. Note that the magnitude of the field is irrelevant. Only the information about the direction of the artificial magnetic field is of importance for orientation estimation [10]. From OpenSim, the exact rotation matrices can be obtained which express this magnetic field in the local virtual IMU frame. This step is equivalent to the addition of the gravity vector as was outlined in Section 3-3-3. This leads to the following relation yielding the magnetometer data expressed in the local virtual IMU frame v

$$m^{\mathrm{v}} = R^{\mathrm{v}G}(q)m^G. \tag{5-2}$$

Here $R^{\mathrm{v}G}(q)$ is the state dependent rotation matrix between the ground frame G and the virtual IMU frame v. For each virtual IMU frame, artificial magnetometer data was synthesized at 100 [Hz] as similar to the sampling frequency of the Xsens MTw IMU. Note that in real experiments, the magnetometer data will be disturbed due to noise. However, for this scenario, no noise was added yielding perfect heading information.

To compute the sensor's orientation required as input for OpenSense, Madwick's orientation estimation algorithm was used [26]. This algorithm was chosen as it is easy in use, is computationally inexpensive, and allows for accurate sensor orientation estimation [26]. It is easy in use as it only requires the IMU data and a single gain value β. This gain value β characterizes the gyroscope measurement error defined as the magnitude of a quaternion derivative. Readers unfamiliar with quaternions can for example refer to [10] and [52]. It is convenient to choose β using the angular velocity quantity $\tilde{\omega}_{max}$. This $\tilde{\omega}_{max}$ represents the maximum static gyroscope measurement error of each axis and can be estimated from the gyroscope's measurements for each axis [53]. The value β is then found from the derivation given in [26]

yielding (5-3)

$$\beta_i = \sqrt{\frac{3}{4}}\tilde{\omega}_{max,i}, \tag{5-3}$$

where the subscript i denotes the i-th IMU considered. The values for the six Xsens MTw gyroscopes were computed being $\beta_1 = 0.0012$, $\beta_2 = 0.0019$, $\beta_3 = 0.0033$, $\beta_4 = 0.0039$, $\beta_5 = 0.0017$, $\beta_6 = 0.0032$. The algorithm uses quaternions to represent the sensor's orientation. The filter fuses the magnetometer data and obtained accelerometer data in an optimized and analytically derived gradient-descent algorithm to determine the direction of the gyroscope measurement error. This error is expressed as a quaternion derivative.

The experimentally logged accelerometer and gyroscope data are not affected by the magnetic material of the KUKA links. For that reason, the experimentally logged gyroscope and accelerometer data from the actual experiment were then fused using the synthesized noise-free magnetometer data in the Madgwick filter to compute sensor orientations. The six sensor orientations were then given as input to OpenSense. The IEKF-OS and OpenSense algorithms were given the same model with the same initial robot configuration and the same placed virtual sensors. The initial configuration for each trial was set using the initial joint encoder values obtained from the KUKA robot.

Results

The same four motions performed and presented in Section 5-2 were used for the motion tracking accuracy comparison of the IEKF-OS algorithm with respect to the method of OpenSense. Both method's joint angle estimations were compared to the ground truth joint angles obtained from the KUKA's encoders. The motion tracking accuracy performances of both methods were assessed using the RMSE metric. The results obtained for all trials are presented in Table 5-2 and the joint angles of trial 3 are depicted in Figure 5-7.

Table 5-2: Comparison of the IEKF-OS joint angle RMSE values with the OpenSense joint angle RMSE values. The RMSE values are computed with respect to the actual ground truth joint encoder values. The mean RMSE joint angle $\bar{q}$ for each trial is shown in the last column.

Trial	Sine wave		Method	RMSE [deg]						RMSE [deg]
	[Hz]	[rad]		q_1	q_2	q_3	q_4	q_5	q_6	$\bar{q}$
1	0.025	1.2	IEKF-OS	0.966	0.370	1.908	0.750	1.770	1.492	1.209
			OpenSense	1.335	4.994	5.881	1.328	1.346	1.643	2.755
2	0.05	1.2	IEKF-OS	1.053	0.448	1.971	0.874	1.204	1.565	1.186
			OpenSense	1.221	5.010	4.619	1.183	1.004	1.393	2.405
3	0.075	0.7	IEKF-OS	2.220	0.322	2.181	0.781	2.789	1.426	1.620
			OpenSense	1.039	6.475	5.514	2.191	0.877	2.128	3.037
4	0.1	0.7	IEKF-OS	0.846	0.486	2.066	0.998	0.955	1.509	1.143
			OpenSense	0.631	2.112	1.641	1.772	0.762	2.312	1.538

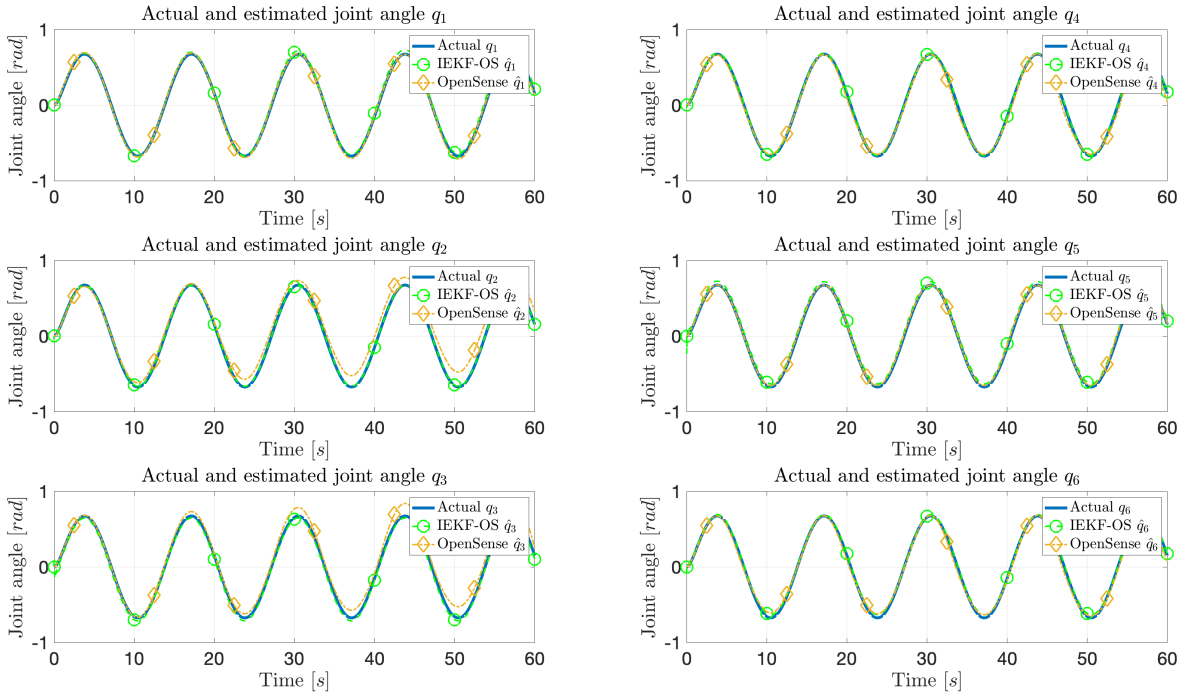


Figure 5-7: Trial 3 with sine waves applied to each joint with a frequency of 0.075 [Hz] and an amplitude of 0.7 [rad]. Comparison between actual joint encoder values shown in blue, IEKF-OS joint angle estimations shown in green, and OpenSense joint angle estimations shown in orange.

For conciseness, the actual and estimated joint angles of the other three trials are shown in Appendix E-2. From the results presented, it can be concluded that the IEKF-OS algorithm incorporating the system dynamics has lower tracking errors for the motions performed on the KUKA robot compared to OpenSense. As can be seen from Table 5-2, looking at the max RMSE values, OpenSense shows values around 6 [deg], whereas the IEKF-OS algorithm does not exceed 3 [deg]. The mean RMSE joint angle $\bar{q}$ of the six RMSE joint angle values is shown in the last column of Table 5-2. Clearly, the mean RMSE $\bar{q}$ of the IEKF-OS algorithm is lower for each trial compared to the mean RMSE $\bar{q}$ of OpenSense. Note that, the IEKF-OS method reconstructs the motion based on the real experimentally obtained gyroscope and accelerometer data. Whereas the sensor orientations computed by the Madgwick filter are based on synthesized perfect magnetometer data. This data gives the exact information about the sensor's heading yielding accurate orientation estimations. This fact, even more, accentuates the motion reconstruction accuracy improvement gained when the system dynamics are included as incorporated in the IEKF-OS algorithm.

5-5 Conclusions

In this chapter, the IEKF-OS algorithm developed, was validated for varying dynamic motions on the physical KUKA robot manipulator. Sensor placement errors are inevitably present between the virtual IMUs and the experimental Xsens IMUs. Despite those sensor placement errors, the IEKF-OS algorithm was still able to reconstruct the original motion. For the four trials, RMSE values with respect to the ground truth joint encoder values are in the range

of 0.4-2.8 [deg]. Vibrations occurring at higher joint velocities resulted in more noisy inertial measurements at these velocities. Yet, the IEKF-OS joint angular velocities showed to be accurate for the four trials with RMSE values ranging between 0.3-2 [deg/s].

As it was not feasible to calibrate the Xsens MTw magnetometers while being attached to the KUKA links, the results obtained from OpenSense with the underlying XKF3hm orientation estimation algorithm are not reliable. Therefore, the IEKF-OS algorithm and OpenSense-XKF3hm could not be compared in terms of motion reconstruction accuracy. From this comparison, it could only be concluded that the IEKF-OS algorithm is able to accurately reconstruct motions regardless of magnetic disturbances. This is because the IEKF-OS method does not rely on magnetometer data.

To be able to compare the IEKF-OS algorithm against OpenSense, noise-free exact magnetometer data was synthesized in a forward kinematics simulation. This motion was the same motion as performed during the actual experiments. Fusing the magnetometer data in a Madgwick filter together with the experimentally obtained gyroscope and accelerometer data allowed OpenSense to reconstruct the KUKA robot motion. From the results obtained, it was shown that the IEKF-OS technique showed lower tracking errors for the trials conducted on the KUKA robot compared to the OpenSense-Madgwick approach. For the IEKF-OS algorithm, mean RMSE joint angle values were in the range of 0.4-2.8 [deg], whereas the OpenSense-Madgwick mean RMSE joint angle values ranged between 0.6-6.4 [deg]. It must be noted that OpenSense, being an inverse kinematics method, highly relies on the underlying sensor orientations computed. Therefore, it would be interesting to test the IEKF-OS algorithm against OpenSense-XKF3hm in a magnetically undistorted environment. Still, it must be noted that accurate orientation estimations were computed as the Madgwick filter used perfect magnetometer data.

With the goal of future human motion reconstruction, the IEKF-OS algorithm first has to be validated for human motions. When validated, the algorithm could open up for the desire of human motion reconstruction. For example, when motions are desired to be captured for patient monitoring at home. The patient will, during its day-to-day tasks, come into close contact with various magnetic materials. When unaccounted for these magnetic disturbances, using OpenSense with magnetometer-based orientation estimates, will not lead to the desired motion reconstruction results. The IEKF-OS algorithm does not rely on this magnetometer data to reconstruct the subject's motion. As such, the IEKF-OS method developed, could be used as an alternative for capturing 3D motions in a markerless and unconstrained environment even when this environment is magnetically disturbed.

Chapter 6

Conclusions and Recommendations

This concluding chapter gives a general summary of this thesis work presented and answers the research questions as they were posed in the introductory chapter. Finally, the recommendations are discussed for future work.

6-1 General summary

The main goal of this thesis was to develop a method that allows to reconstruct 3D motions using inertial measurements and the system's dynamical model together with its kinematic constraints. To that aim to devise this novel technique, this thesis functioned as a bridge between the fields of sensor fusion and biomechanical modeling.

The proposed structure of the motion reconstruction algorithm allows to include more kinematic information compared to the state-of-the-art sensor fusion approach of Weygers et al. (2020) [15]. In their approach, the authors only assume that two bodies are linked by a spherical joint allowing the bodies to freely rotate relative to each other. Compared to their method, including the type of joint, as presented in this thesis, constrains the kinematically possible movements between two linked rigid bodies. This allows for accurate motion reconstruction on the KUKA robot manipulator as was shown using numerical simulations and experiments. Next to that, also inertial properties of each rigid body such as segment masses and lengths, center of mass locations, and moments of inertia are taken into account. This similar to the approach taken by Dorschky et al. (2019) [6]. However, the method of [6], is currently limited to 2D human motion reconstruction. For that reason, this thesis focused on including a 3D model such that 3D motions of the KUKA robot manipulator could be reconstructed.

It was observed that sensor placement errors, mainly rotational errors, affect the motion reconstruction accuracy. This is expected as rotational errors affect the measurements from both the gyroscope and the accelerometer. Translation errors, on the other hand, only affect the readings of the accelerometer. This can be explained as the angular velocity, expressed in a known frame, is only dependent on the orientation of that frame, not on its location on

the body. Hence, is expected that the motion reconstruction accuracy will improve when the sensor placement errors, between virtual and experimental sensors, are minimized.

The IEKF-OS algorithm was verified and validated on a robot manipulator using numerical simulations and experiments, respectively. From these analyses, it could be concluded that the 3D motions of the robot's six interconnected links could be reconstructed. Although these results cannot be translated directly to human motion capture, the next step would be to do so and assess the IEKF-OS' motion reconstruction accuracy for human motions.

6-2 Research questions

The main research question was for this thesis was defined as:

How can the inertial measures of multiple IMUs and the system's dynamical model be used to improve 3D motion reconstruction accuracy?

To answer this question, three sub-research questions were initially posed in the introductory chapter. The results of this thesis led to the following answers.

What method can be developed to exploit the system's nonlinear dynamical model while simultaneously addressing the noise affecting the inertial measurements?

The IEKF-OS algorithm derived in Chapter 3, showed that accurate motion reconstruction is possible on a six link robot manipulator. The first stage, the time update in the Kalman filter, incorporates the subject's dynamics, as computed by OpenSim, to predict the motion. Due to the nonlinearity of the motion models governing the motion of the system, linearizations at the current state are computed. As such allowing to form the algorithm's required Jacobians matrices. The second stage of the algorithm, the correction step, then fuses the IMU measurements from multiple IMUs to improve the model estimates of joint angles and speeds at the current state. Besides, using the iterative nature of the IEKF-OS then allows to increase the accuracy of the estimations made. As a result, a congruent virtual motion reconstruction is obtained. By including the control torque τ in the state vector, the technique devised allows to approximate the torque needed to drive the model. Although no validation of applied control inputs has been performed on the KUKA robot manipulator, this is shown on an actuated double pendulum simulation as presented in Appendix C. There, from numerical simulations performed, it is shown that due to the inclusion of the inertial properties of the model, accurate reconstruction of the original torque applied, was possible. It must be noted that that simulation was again for an idealized scenario in which no sensor placement errors were imposed.

How sensitive is the new algorithm to common sensor placement errors when applied to a robot manipulator?

The results in Chapter 4 show that for moderate sensor placement errors, the IEKF-OS algorithm is still able to reconstruct the original motion. When each attached IMU was translated in an X, Y and Z direction of 3 [cm] with respect to its original location, from 100 Monte Carlos simulations, it was shown that joint angle RMSE values remained below 1 [deg] for all joint angles. This could be explained as the angular velocity measurements were not affected

by these translated frames. Contrary, it was observed that imposing rotation errors around each IMU's axis of $1.5 \cdot X_{RMS} = 2.25$ [deg] resulted in a higher joint angle RMSE. For that scenario, RMSE values ranged between 1-5 [deg] per joint angle. This could be explained as re-orienting the frame also affects the angular velocities measured. The algorithm's motion tracking performance really showed to deteriorate when rotational errors around each attached IMU's axis were applied of $5 \cdot X_{RMS} = 7.5$ [deg] next to already present translation errors of 3 [cm] in each axis. For that scenario, joint angle RMSE values ranging between 7-10 [deg] were observed. Care must, therefore, be taken when attaching IMUs to rigid bodies. Modeling the virtual IMU's orientation in agreement with the attached experimental IMU is thus key to obtain accurate motion reconstruction. It should be noted that these outcomes are only an indication of the algorithm's robustness to sensor placement errors when these IMU sensors are attached to the KUKA robot manipulator. For human motion capture, one also needs to take into account the additional soft tissue artifacts (STAs) for the case when sensors are attached to human skin. These STAs will disturb the IMU measurements made to a larger extend.

How does the novel algorithm utilizing the system's dynamical model compare to a method which solely uses kinematical constraints in terms of tracking performance for various ranges of motion?

The IEKF-OS method was compared against the method developed by the creators of OpenSim, called OpenSense. OpenSense is an inverse kinematics based approach. This tool uses the sensor orientation estimates and constrains these to the underlying model. For this technique to work, accurate orientation estimations are required. For this thesis, Xsens IMUs were used, from which the IMUs orientation can be estimated using Xsens' XKF3hm algorithm. This XKF3hm uses the magnetometer to compensate for the drift in the yaw angle. Due to the joint torques applied and the KUKA link's magnetic material, these magnetometer measurements were distorted. No comparison could therefore be made between the IEKF-OS and OpenSense-XKF3hm methods in terms of motion reconstruction accuracy. It could only be concluded that the IEKF-OS method is able to reconstruct motions in a magnetically disturbed environment as it does not rely on magnetometer data. To be able to compare both method's motion tracking accuracy performances, an additional analysis was performed. To that aim, exact magnetometer data was synthesized in a forward kinematics simulation congruent to the actual motions performed during the experiments. The Madgwick orientation estimation algorithm was then used to compute sensor orientations from the artificial exact magnetometer data, and experimentally obtained gyroscope and accelerometer data. The exact magnetometer data having no noise added, provided the ideal heading information for the Madgwick estimated IMU orientations. Still, from the results obtained, it was observed that the IEKF-OS algorithm, solely relying on experimentally obtained IMU data, had lower joint angle tracking errors. The IEKF-OS algorithm for the four experiments had joint angle RMSE values in the range of 0.4-2.8 [deg]. The OpenSense-Madgwick method contrary resulted in RMSE values ranging between 0.6-6.4 [deg]. For the various trials conducted, both method's joint angle estimations were compared against the ground truth joint encoder values from the KUKA robot. One point needs to be made regarding this comparison. OpenSense highly relies on the underlying method that computes the sensor orientations. As such, no remarks can be made whether the IEKF-OS algorithm also shows lower tracking errors when it is compared against OpenSense with Xsens' XKF3hm orientation estimation algorithm.

6-3 Recommendations for future work

With the new method proposed for reconstructing 3D motions, some recommendations for future work are discussed.

Application to biomechanical models

With the IEKF-OS algorithm validated on the KUKA robot manipulator, it would be interesting to assess the IEKF-OS motion reconstruction performance for human motions. Future work could, therefore, extend this work to for instance the scapulothoracic joint model created by [54] to examine human shoulder motions. From the results presented in this thesis, it can be stated that the algorithm developed allows to reconstruct motions for six interconnected robotic links. Hence it would, therefore, be assumed that motions of the human thorax, shoulder, upper arm, and lower arm, modeled as a four-link system, could also be reconstructed using this algorithm. This statement, however, cannot be validated as such human motion experiments have not been performed.

Including muscle physiology into the biomechanical model

Extending on the previous future work direction, it would be interesting to include muscle physiology into the biomechanical model. In this thesis, the actuators were modeled as coordinate actuators, as only mechanical systems have been concerned. For biomechanical models, however, a more realistic approach would be to model these actuators as muscles. Recently, the scapulothoracic joint model [54] was updated which includes muscles [24]. With this musculoskeletal model, even more human movement-related metrics could be estimated.

Comparison against Optical Motion Capture

When the IEKF-OS algorithm was compared against the OpenSense-XKF3hm method with experimentally obtained IMU data, no conclusions could be made about which method has lower tracking errors. This due to the fact that the magnetometers couldn't be calibrated correctly due to the configuration and motions possible by the KUKA. Moreover, subsequent links nearing other links located closer to the root of the KUKA influenced the local magnetic field. This due to the steel material inferring with the local magnetic field. As such, when both methods are applied to human motions in a magnetically undisturbed field, this would allow for a fair comparison. This due to the fact that human bodies consisting of bones and muscle logically do not interfere with the local magnetic field. The downside then would be that there is no ground truth data available. As a solution, both methods should be tested on human motions and compared to a data set obtained from an Optical Motion Capture (OMC) system.

C++ implementation

The current limitation of the approach presented, lies in the fact that complex systems composed of multiple linked rigid body segments are described by a larger set of state variables.

The state vector x will grow with $3n$ for each added DoF n. Besides, for each added body, an additional IMU is needed for which predicted measurements need to be computed. This extension of the state vector with additional IMUs would translate to larger motion and measurement model Jacobians. Together with the measurement update iterations, rendering an increase in computational time. As a result, this then diverges from the eventual desire towards real-time motion reconstruction in the field. As the IEKF-OS algorithm is currently only incorporated in `MATLAB`, future work could look towards implementing the algorithm in `C++` for faster computational times.

Improve the sensor-to-segment calibration

For this thesis, a simple sensor-to-segment calibration method was developed. This method relies on Xsens' XKF3hm algorithm. Using XKF3hm, the orientation of the Xsens IMU can be obtained. The IMU's position was manually measured using a tape measure. With this information, the corresponding virtual IMU on the OpenSim model was modeled. Future work, however, could look at more advanced calibration procedures to minimize the sensor placement error between modeled virtual IMUs and experimental IMUs.

Concluding remarks

To conclude, with this novel approach presented, a powerful motion reconstruction algorithm has been developed. The author, therefore, encourages, in particular future researchers and scientists in the biomechanical/clinical field, to extend this work to human motion reconstruction. To that potential, the author hopes that this thesis can make a small contribution one day to clinical studies investigating sport activities or research examining abnormal or impaired human movements. The author is convinced that fascinating insights will be achieved when motions of patients are unobtrusively logged using these small and lightweight inertial sensors during the patient's day-to-day life. To facilitate that step, the author plans to make the code, model, results, and experiment data available on the GitHub https://github.com/DaandeKanter/IEKF-OS_Algorithm, for those researchers interested.

Appendix A

Probabilistic derivation posterior in IEKF

To derive the posterior x_t as given in (3-50) in the *Iterated Extended Kalman Filter*, the joint density for the state and measurement at time t, as shown in (3-49), is evaluated and can be expressed as approximately Gaussian [45]. For plainness, (3-49) is presented here again

$$\begin{aligned} p(x_t, y_t|\check{x}_0, y_{0:1-t}) &\approx \mathcal{N}\left(\begin{bmatrix}\mu_{x,t}\\ \mu_{y,t}\end{bmatrix}, \begin{bmatrix}\Sigma_{xx,t} & \Sigma_{xy,t}\\ \Sigma_{yx,t} & \Sigma_{yy,t}\end{bmatrix}\right), \\ &= \mathcal{N}\left(\begin{bmatrix}\check{x}_t\\ \bar{y}_t + H_t(\check{x}_t - \bar{x}_t)\end{bmatrix}, \begin{bmatrix}\check{P}_t & \check{P}_t H_t^\top\\ H_t\check{P}_t & H_t\check{P}_t H_t^\top + M_t R_t M_t^\top\end{bmatrix}\right). \end{aligned} \tag{A-1}$$

In (A-1), the joint Gaussian over a pair of variables (x,y) is written as

$$p(x, y) = \mathcal{N}\left(\begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix}, \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}\right). \tag{A-2}$$

Observing that, $\Sigma_{yx} = \Sigma_{xy}^\top$ and that it has the same exponential form as the multivariate Gaussian PDF $p(x|\mu, \Sigma)$ where $x \in \mathbb{R}^N$ is given as

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^N \det \Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right). \tag{A-3}$$

Here $\mu \in \mathbb{R}^N$ is the mean and $\Sigma \in \mathbb{R}^{N\times N}$ is the symmetric positive definite covariance matrix.

Subsequently, the Gaussian conditional density for x_t, i.e. *the posterior*, is to be determined where it is assumed that y_t is known. To determine this posterior, note that it is always possible to break a joint density into the product of two factors: $p(x, y) = p(x|y)p(y)$. The key here is to factor out the term $\begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}$ for the joint Gaussian case by making use of

the Schur complement [43] as

$$\begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix} = \begin{bmatrix}\mathcal{I} & \Sigma_{xy}\Sigma_{yy}^{-1}\\ 0 & \mathcal{I}\end{bmatrix}\begin{bmatrix}\Sigma_{xx}-\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} & 0\\ 0 & \Sigma_{yy}\end{bmatrix}\begin{bmatrix}\mathcal{I} & 0\\ \Sigma_{yy}^{-1}\Sigma_{yx} & \mathcal{I}\end{bmatrix}. \tag{A-4}$$

As in (A-3), the inverse of Σ is taken, inverting both sides of (A-4) yields

$$\begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}^{-1} = \begin{bmatrix}\mathcal{I} & 0\\ -\Sigma_{yy}^{-1}\Sigma_{yx} & \mathcal{I}\end{bmatrix}\begin{bmatrix}(\Sigma_{xx}-\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx})^{-1} & 0\\ 0 & \Sigma_{yy}^{-1}\end{bmatrix}\begin{bmatrix}\mathcal{I} & -\Sigma_{xy}\Sigma_{yy}^{-1}\\ 0 & \mathcal{I}\end{bmatrix}. \tag{A-5}$$

Subsequently, looking at the exp-term $(x-\mu)^\top\Sigma^{-1}(x-\mu)$, then $p(x,y)$ can be written using the Schur complement as

$$\begin{aligned}
&\left(\begin{bmatrix}x\\ y\end{bmatrix}-\begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix}\right)^\top\begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}^{-1}\left(\begin{bmatrix}x\\ y\end{bmatrix}-\begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix}\right)\\
&=\left(\begin{bmatrix}x\\ y\end{bmatrix}-\begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix}\right)^\top\begin{bmatrix}\mathcal{I} & 0\\ -\Sigma_{yy}^{-1}\Sigma_{yx} & \mathcal{I}\end{bmatrix}\begin{bmatrix}(\Sigma_{xx}-\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx})^{-1} & 0\\ 0 & \Sigma_{yy}^{-1}\end{bmatrix}\begin{bmatrix}\mathcal{I} & -\Sigma_{xy}\Sigma_{yy}^{-1}\\ 0 & \mathcal{I}\end{bmatrix}\left(\begin{bmatrix}x\\ y\end{bmatrix}-\begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix}\right),\\
&=\Big(x-\mu_x-\Sigma_{xy}\Sigma_{yy}^{-1}(y-\mu_y)\Big)^\top\Big(\Sigma_{xx}-\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\Big)^{-1}\Big(x-\mu_x-\Sigma_{xy}\Sigma_{yy}^{-1}(y-\mu_y)\Big)\\
&+(y-\mu_y)^\top\Sigma_{yy}^{-1}(y-\mu_y).
\end{aligned} \tag{A-6}$$

Notice that this is the sum of two quadratic terms. Moreover, observe that the exponential of a sum is the product of two exponentials, hence

$$p(x,y) = p(x|y)p(y), \tag{A-7a}$$

$$p(x|y) = \mathcal{N}\left(\mu_x+\Sigma_{xy}\Sigma_{yy}^{-1}(y-\mu_y), \Sigma_{xx}-\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\right), \tag{A-7b}$$

$$p(y) = \mathcal{N}(\mu_y,\Sigma_{yy}). \tag{A-7c}$$

As it was assumed that y_t is known, the Gaussian conditional density for x_t, *the posterior*, is given as

$$p(x_t|\check{x}_0,y_{0:t}) = \mathcal{N}\Big(\underbrace{\mu_{x,t}+\Sigma_{xy,t}\Sigma_{yy,t}^{-1}(y_t-\mu_{y,t})}_{\hat{x}_t}, \underbrace{\Sigma_{xx,t}-\Sigma_{xy,t}\Sigma_{yy,t}^{-1}\Sigma_{yx,t}}_{\hat{P}_t}\Big), \tag{A-8}$$

here $\hat{x}_t$ denotes the mean and $\hat{P}$ its covariance. This concludes the derivation of (3-50). Continuing by observing (A-8) closely, the reader can see that the Kalman gain is given by $\Sigma_{xy,t}\Sigma_{yy,t}^{-1}$. Furthermore, $\mu_{x,t}$ and $\Sigma_{xx,t}$ are set to $\mu_{x,t}=\check{x}_t$ and $\Sigma_{xx,t}=\check{P}_t$. Thus deriving from (A-8), this yields the generalized Gaussian update/correction equations

$$K_t = \Sigma_{xy,t}\Sigma_{yy,t}^{-1}, \tag{A-9a}$$

$$\hat{P}_t = \check{P}_t - K_t\Sigma_{xy,t}^\top, \tag{A-9b}$$

$$\hat{x}_t = \check{x}_t + K_t(y_t-\mu_{y,t}). \tag{A-9c}$$

Contrary to the general KF, for the IEKF case, the moments of $\mu_{y,t},\Sigma_{yy,t}$ and $\Sigma_{xy,t}$ as shown in (A-1) are now substituted which concludes this procedure and yields the IEKF equations

$$K_t = \check{P}_tH_t^\top(H_t\check{P}_tH_t^\top+M_tR_tM_t^\top)^{-1}, \tag{A-10a}$$

$$\hat{P}_t = (\mathcal{I}-K_tH_t)\check{P}_t, \tag{A-10b}$$

$$\hat{x}_t = \check{x}_t + K_t\Big(y_t-\bar{y}_t-H_t(\check{x}_t-\bar{x}_t)\Big). \tag{A-10c}$$

Appendix B

Implementation in OpenSim

When a user-defined system, modeled as an `.osim` model is created, the first step to running a simulation is to initialize this just created model. For convenience, this `.osim` model will be named `osimModel`. The components of the model need to be interconnected and a Simbody Multibody System needs to be created to represent the `osimModel` computationally. Then this model needs to be finalized. These steps can be accomplished by running the OpenSim command `osimModel.initSystem` on the `osimModel` which realizes the following steps

1. Generate the Equations of Motion (EoM) governing the dynamics of the model.
2. Assembles the model to satisfy position constraints.
3. Returns the initial state as a `state` object.

When no initial state is specified, OpenSim will set the state equal to 0. In OpenSim, simulations for a user-defined system can then be performed by first constructing a `Manager`. This class within OpenSim manages the execution of a simulation and by default uses a Runge-Kutta-Merson integrator. First, the `osimModel` can be passed to this `Manager` constructor to create an object of class `Manager` named `manager` as

$$\texttt{manager} = \texttt{Manager}(\texttt{osimModel}). \tag{B-1}$$

The next step is to initialize the `manager` at the current `state` as

$$\texttt{manager.initialize}(\texttt{state}). \tag{B-2}$$

The model can then be simulated from start till a specified final time $fTime$ in one go by integrating the EoM for the specified model at the current state as

$$\texttt{manager.integrate}(fTime). \tag{B-3}$$

If, however, users also want to extract values during simulation, like is the idea for this thesis, a for-loop $t = 1 : n$ can be defined such that the `state` is iteratively updated per time step as

$$\texttt{state} = \texttt{manager.integrate}(t * dTime), \tag{B-4}$$

where $dTime$ is the time step and n the total simulation time. For this thesis, this $dTime$ value will be equal to the maximal available sampling frequency of the used Xsens MTw Awinda IMUs. This sampling frequency, however, depends on the size of the set of MTw's used and is depicted in Table G-4. Using this for-loop approach, information such as the values of the generalized coordinates q and generalized velocities u can be called at the current time t as

$$q = \texttt{state.getQ}, \tag{B-5a}$$

$$u = \texttt{state.getU}, \tag{B-5b}$$

$$\dot{q} = \texttt{state.getQDot}, \tag{B-5c}$$

$$\dot{u} = \texttt{state.getUDot}. \tag{B-5d}$$

Note that if one is interested in the derivatives of q and u, being $\dot{q}$ and $\dot{u}$ respectively, one must realize the OpenSim acceleration stage prior to calling these `state.getQDot` and `state.getUDot` functions. This realization can be accomplished as

$$\texttt{osimModel.realizeAcceleration(state)}. \tag{B-6}$$

Appendix C

Simulation of an actuated double pendulum

Prior to testing the IEKF-OS motion reconstruction algorithm on the KUKA robot manipulator, the novel algorithm was first verified on a rather simple actuated double pendulum system. First of all, a schematic overview of the verification process is shown in Figure C-1.

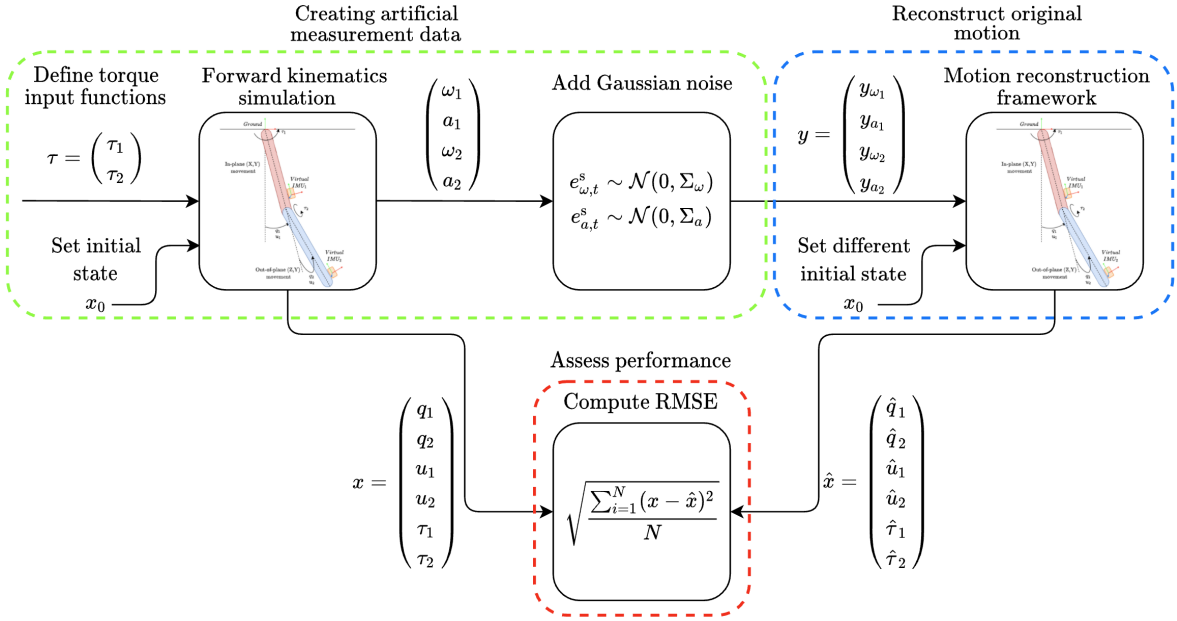


Figure C-1: Overview of the performed steps. First, as shown in the green box, artificial measurement data is created from a forward kinematics performed simulation after which Gaussian noise is added. With the simulated artificial measurement data, the motion is then reconstructed using the developed motion reconstruction algorithm as shown in the blue box. Lastly, the performance of the IEKF-OS algorithm is assessed in the red box by means of the computed RMSE error between the actual state variables x and the estimated state variables $\hat{x}$.

As this verification process is simulation-based, the original values of the state vector x can be extracted from this forward kinematics simulated system. The values obtained, can then be compared against the predicted state vector $\hat{x}$ computed by the IEKF-OS algorithm. The reconstruction performance is then assessed by means of the Root-Mean-Square Error (RMSE) between the actual state variables x and predicted state variables $\hat{x}$ as will be outlined in Section C-3. With the overview of the procedure detailed, the model of the simulated double pendulum with the two hinge joints allowing for 3D motion is presented next.

C-1 The double pendulum OpenSim model

The upper link of the double pendulum, shown in red in Figure C-2a, is attached to the ground by means of a pin joint allowing this link to rotate in the (X, Y) plane. To have the lower link of the double pendulum, shown in blue in Figure C-2a, perform motions out of its 2D plane spanned by the axes (X, Y), a second pin joint was modeled. This joint is located between the upper link and the lower link and is rotated by 90 [deg] with respect to the first joint around the Y-axes. This allows the lower link to perform motions in the (Z, Y) plane. Combined with the first link, the lower link will perform motions in a 3D space. As to each body, one virtual IMU frame is attached, the three angular velocities ω and three linear accelerations a can be measured by each virtual attached IMU. A schematic overview of the double pendulum and the generalized coordinates is shown below in Figure C-2a. Next to this schematic, the double pendulum as modeled in the OpenSim environment is shown in Figure C-2b.

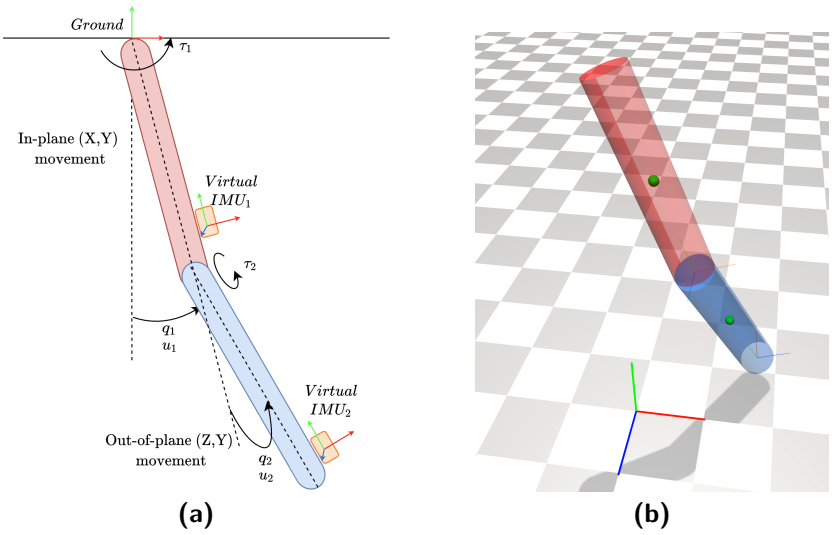


Figure C-2: (a) Schematic overview of the double pendulum and its generalized coordinates. Moreover, each link has a virtual IMU attached, which both are modeled as reference frames. (b) The modeled double pendulum system shown in the OpenSim environment together with two reference frames functioning as virtual IMUs. The center of mass locations are shown by green spheres at the end of each link. The OpenSim reference frame can be seen in the left corner.

To create a model in OpenSim, for each body, the mass, the center of mass locations in the body frame, and the elements of the inertia tensor measured about the mass center need to be defined. The double pendulum system was modeled as two solid cylinders each having a length of 1 [m], a radius of 0.1 [m], and a mass of 1 [kg]. The center of mass was located at the center of the cylinder as can be seen by the green spheres in Figure C-2b. For a solid cylinder with radius r, length l and mass m, the inertia tensor was equal to

$$I_1 = I_2 = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} [\text{kg m}^2], \tag{C-1a}$$

$$= \begin{pmatrix} \frac{1}{12}m(3r^2+h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2+h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{pmatrix} = \begin{pmatrix} 0.085833 & 0 & 0 \\ 0 & 0.085833 & 0 \\ 0 & 0 & 0.005 \end{pmatrix}. \tag{C-1b}$$

With these inertial properties described, attention will now be turned to the state vector of this double pendulum system.

C-1-1 State vector

The system dynamics are described using the generalized coordinate formulation. For each Degree of Freedom (DoF), three state variables are used. Here, the first state variable describes its joint angle, denoted by q, the second state variable describes the joint angular velocity, u, and lastly, the augmented state variable τ denotes the joint torque applied to that joint. For instance, for two linked rigid-body segments, each subsequent generalized coordinate is described with respect to its previous one as can be seen in Figure C-2a. There, the joint angle q_2 is defined relative to its linked upper body. The joint angle q_1 is then again defined relative to the vertical of the ground frame. As the system has two DoFs, the total state vector x describing the system will consist of six state variables resulting in

$$x = \begin{pmatrix} q_1 \\ q_2 \\ u_1 \\ u_2 \\ \tau_1 \\ \tau_2 \end{pmatrix}. \tag{C-2}$$

C-1-2 Actuating the double pendulum model

To actuate the double pendulum model with the torque values as will be computed by the IEKF-OS algorithm, coordinate actuators and controller objects were defined in the OpenSim model. Each coordinate actuator is only allowed to actuate the respective generalized coordinate it has access to. Subsequently, a controller object needs to be made which defines the amount of control torque that this coordinate actuator needs to supply to its joint. This controller gets the computed IEKF-OS joint torque value as its input. Lastly, prior to running a forward simulation, virtual IMUs need to be attached which is detailed in the next section.

C-1-3 Attaching the virtual IMUs

With the system properties defined, the virtual IMUs which are modeled as reference frames will now be detailed. These frames are visualized in Figure C-2b as orthogonal XYZ-frames, with the X-axes in red, the Y-axes in green, and the Z-axes in blue. The first virtual IMU is located at the center of the end of the first link. Similarly, the second virtual IMU is located at the center of the end of the second link. For the double pendulum's default configuration of $q_1 = q_2 = 0$ [rad], the first virtual IMU has its X-axis, Y-axis, and Z-axis pointing along the positive OpenSim ground reference frame X-, Y-, and Z-axes respectively. The joint between the upper and lower link is rotated 90 [deg] counterclockwise around the Y-axis with respect to the upper joint. Hence, the second virtual IMU for this pendulum's default configuration has its X-axis pointing along the negative OpenSim ground reference frame Z-axis, and its Z-axis pointing along the positive OpenSim ground reference frame X-axis while having its Y-axis pointing along the positive OpenSim ground reference frame Y-axis. With the created model in place, the next section describes the contents as shown in the green box of the schematic overview as was given in Figure C-1.

C-2 Creating artificial IMU measurements

As for this verification process, no real IMU measurements are available, they first have to be created. Hence, the idea is to initialize the double pendulum system at a known initial state x_0 and perform a forward kinematics simulation. Moreover, various known joint torque step functions will be applied to the joints. The aim is here to eventually estimate these torques using the IEKF-OS algorithm and compare the actual and predicted torques later on.

Compared to the current motion tracking technique of [15], this is an additional benefit as in this algorithm, the system's segment masses and lengths, center of mass locations, and moments of inertia are taken into account, allowing for this joint torque estimation. A current limitation of the method presented in this thesis, however, is that no constraints are invoked on the computed joint torques. Next to that, for the algorithm's eventual use for human motion reconstruction in the field, instead of the currently modeled coordinate actuators, muscle objects need to be included in the model of the human. These muscle objects would then allow for accurate muscle energy predictions.

Nonetheless, as the focus of this thesis is limited to the reconstruction of motions of mechanical systems, future work could look towards reconstruction of human motions by instead modeling a human body with muscles as actuators. Hence, for the rest of this chapter, it will now be assumed that there are no limits on the to be supplied control torques to minimize the error between the predicted and actual motion. This mainly since the system at hand does not represent an actual modeled double pendulum setup, but was just modeled by the author himself for the algorithm's verification purposes. Attention will now be turned to the double pendulum system where first the known torque input functions and the pendulum's initialization will be specified.

Torque input functions and initialization

Instead of a forward kinematics simulation, where the system is just let go from its initial state, the system will be actuated by two different supplied control torques yielding a non-autonomous system. Here, τ_1 is the joint torque supplied to joint angle q_1 and similarly, τ_2 the joint torque actuating joint angle q_2. A visualization of these joint torque step functions, $f(t)_{\tau_1}$ and $f(t)_{\tau_2}$, with their starting and end times is shown in Figure C-3. The initial state of the pendulum was set at rest meaning that both joint angles were equal to q_1=q_2=0 [deg] and both joint angular speeds were also set to zero: u_1=u_2=0 [deg/s].

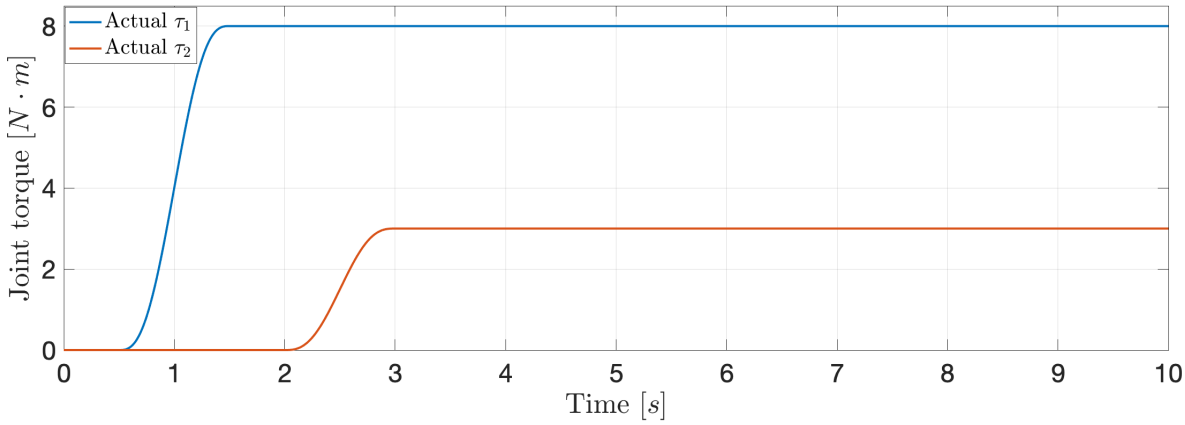


Figure C-3: The two control torque step functions $f(t)_{\tau_1}$ and $f(t)_{\tau_2}$ applied to the joint angles q_1 and q_2 respectively.

As these input torque functions are known at each time, this allows to also assess the performance of the IEKF-OS algorithm in estimating these various control inputs. The results obtained will be detailed in Section C-3. With the initialization defined, the next step will be to add Gaussian noise to the obtained measurements to mimic experimentally logged IMU data.

Adding Gaussian noise to the artificially created IMU measurements

The system is simulated at 100 [Hz] using the initial conditions and given input torque step functions as shown in the previous section. The sampling frequency of 100 [Hz] was chosen as this is the maximum sampling frequency of the eventually used Xsens MTw IMUs when a set of six IMUs are used [11]. This for the reason as during the experiments conducted on the KUKA robot manipulator, as outlined in Chapter 5, a set of six Xsens IMUs will be used. Looking at (3-52c), it can be seen that for each measurement, also a predicted measurement is needed. For this reason, the sampling frequency of the motion reconstruction algorithm should always be set similar to the sampling frequency of the experimental Xsens IMUs.

From the attached OpenSim frames, representing the virtual IMUs, during the simulation the desired angular velocity ω and linear acceleration a of each pendulum link can be obtained. This as the idea of the IEKF-OS algorithm is based on tracking raw angular velocities and linear accelerations. Note, however, that in a real experiment, these quantities are measured by body attached IMUs and are typically affected by noise, for example see Chapter 2 of [10].

Hence, to get the simulation in agreement with an actual motion tracking experiment, noise should be added. It is assumed that the noise, corrupting these measured angular velocities and linear accelerations in real experiments, is of Gaussian form. To create noise that is similar to real noise values of the eventually used Xsens MTw IMUs [11], first, the noise covariances of two calibrated stationary IMU sensors were found experimentally at 100 [Hz] as

$$\Sigma_{\omega_1} = \begin{pmatrix} 2.357 & 0 & 0 \\ 0 & 3.364 & 0 \\ 0 & 0 & 3.048 \end{pmatrix} \cdot 10^{-6},\ \Sigma_{\omega_2} = \begin{pmatrix} 3.901 & 0 & 0 \\ 0 & 2.103 & 0 \\ 0 & 0 & 2.550 \end{pmatrix} \cdot 10^{-6}.$$
$$\Sigma_{a_1} = \begin{pmatrix} 1.957 & 0 & 0 \\ 0 & 1.840 & 0 \\ 0 & 0 & 3.102 \end{pmatrix} \cdot 10^{-4},\ \Sigma_{a_2} = \begin{pmatrix} 1.932 & 0 & 0 \\ 0 & 1.817 & 0 \\ 0 & 0 & 2.792 \end{pmatrix} \cdot 10^{-4}.$$

where Σ_ω is the noise covariance matrix obtained from the gyroscope and Σ_a the noise covariance matrix of the accelerometer. Using these obtained noise covariance matrices, the noise vectors to be added to the gyroscope simulated angular velocity ω and accelerometer simulated linear acceleration a were created as $e_\omega \sim \mathcal{N}(0, \Sigma_\omega)$ and $e_a \sim \mathcal{N}(0, \Sigma_a)$ using the `MATLAB` function `mvnrnd`. The artificial measurements were then constructed by adding these obtained noise values to the already obtained simulated ω and a signals as

$$y = \begin{pmatrix} y_{\omega_1} \\ y_{a_1} \\ y_{\omega_2} \\ y_{a_2} \end{pmatrix} = \begin{pmatrix} \omega_1 + e_{\omega_1} \\ a_1 + e_{a_1} \\ \omega_2 + e_{\omega_2} \\ a_2 + e_{a_2} \end{pmatrix}. \tag{C-3}$$

Note that no plots of this obtained motion will be presented here as these obtained results will be shown together with the IEKF-OS estimated motion in the next section. This for the reason of easier comparison. With the forward kinematics simulation detailed, the motion's reconstruction as estimated by the IEKF-OS algorithm as shown in the blue box of Figure C-1 will now be presented.

C-3 Double pendulum motion reconstruction and results

The motion tracking performance of the IEKF-OS algorithm will now be assessed. Here, the results of an idealized simulated scenario will be presented. With idealized, it is meant that the locations and orientations of the virtual IMUs in the IEKF-OS system, as shown in the blue box in Figure C-1, are identical to the locations and orientations of the virtual IMUs in the forward kinematics simulated system, as shown in the green box in Figure C-1. Mainly for the reason, as it was first key to perform some sanity checks if the developed IEKF-OS method was able to reconstruct the original motion. Moreover, the aim of this double pendulum simulation was also to check the convergence of the IEKF-OS algorithm. This as the IEKF-OS pendulum system will be initialized differently compared to the configuration of the forward kinematics simulated pendulum system.

Initialization

Compared to the initial state of the double pendulum system generating the measurements, the initial state of the IEKF-OS double pendulum system was initialized with both joint angles q_1 and q_2 set at 10 [deg]. The initial joint angular velocities u_1 and u_2 were also set at 10 [deg/s]. For an unactuated system, this would result in a totally different obtained simulation. However, the IEKF-OS algorithm will try to minimize this difference by applying joint torques to the two joints. As such, trying to reduce this error and reconstruct the original motion.

Number of measurement iterations ϵ

There are various parameters in the IEKF-OS algorithm to be tuned. For instance, the number of measurement iterations ϵ can be defined as shown in Algorithm 3. This is essentially a trade-off that needs to be made. More measurement update iterations allow for estimating the actual underlying state value more accurately, for an example see [50]. The downside of choosing a lot of these iterations translates to an increase in computational cost and time. This then diverges from the desire of applying this algorithm for eventual real-time motion reconstruction. For this reason, three measurement update iterations were chosen, as from simulations it was shown that the performance gained for more iterations was negligible.

Process and measurement covariances Q and R

As the IEKF-OS simulated system is the same as the forward kinematics simulated system, the process noises for both q and u were set to zero. This yields $Q_q = Q_u = \mathcal{O}_3$ which represents that there is no process noise and thus no errors that are expected in the dynamical equations giving the updates of these variables. On the other hand, looking back to Section 3-5-2, $w_\tau(t)$ in (3-37) should be tuned such that the best performance in terms of RMSE between the actual state vector x and predicted state vector $\hat{x}$ is obtained. Iteratively, it was found that, for this system and simulation, Q_τ set equal to $1 \cdot \mathcal{I}_3$ [Nm] yielded the lowest RMSE.

The measurement covariance matrix R was set to 100 times the previously obtained Xsens IMU covariance matrices of Σ_{ω_1}, Σ_{a_1}, and Σ_{ω_2}, Σ_{a_2} yielding more variance than the actual computed noise covariance matrices. The resulting R matrix is of the form

$$R = 100 \cdot \begin{pmatrix} \Sigma_{\omega_1} & \mathcal{O}_3 & \mathcal{O}_3 & \mathcal{O}_3 \\ \mathcal{O}_3 & \Sigma_{a_1} & \mathcal{O}_3 & \mathcal{O}_3 \\ \mathcal{O}_3 & \mathcal{O}_3 & \Sigma_{\omega_2} & \mathcal{O}_3 \\ \mathcal{O}_3 & \mathcal{O}_3 & \mathcal{O}_3 & \Sigma_{a_2} \end{pmatrix}, \tag{C-4}$$

where $\mathcal{O}_3$ denotes a matrix consisting of zeros with size $\mathbb{R}^{3\times3}$. With all the to be tuned parameters discussed, the results obtained will now be presented.

Results

To evaluate the motion tracking performance, the system was simulated for 10 [s]. From the simulation obtained, it was observed that the IEKF-OS estimations already converged within

one iteration towards a steady-state error. The state estimates of the IEKF-OS converged to the actual state vector x after the system became more dynamic when the first joint torque on the original system was applied. Hence, the RMSE between the IEKF-OS estimated state vector $\hat{x}$ and the actual state vector x is calculated from 0.75 [s] till the end of the simulation. The obtained RMSE values are presented in Table C-1.

Table C-1: Obtained RMSE values between the actual and IEKF-OS estimated state variables.

Joint angle	RMSE [deg]	Joint angular velocity	RMSE [deg/s]	Joint torque	RMSE [Nm]
q_1	0.063	u_1	0.059	τ_1	0.023
q_2	0.091	u_2	0.134	τ_2	0.009

From this table, it can be concluded that the IEKF-OS algorithm presented in this thesis, is able to reconstruct the original motion accurately. The obtained RMSE values are all in the order of magnitude of 10^{-1} or lower. One must note, however, that this is an idealized scenario where no sensor placement errors have been introduced between modeled virtual IMU sensors. As such the errors presented in this table, are likely due to the noise that was added to the artificially created IMU measurements. Next to these RMSE values, the responses and computed joint torques are depicted in Figures C-4 and C-5.

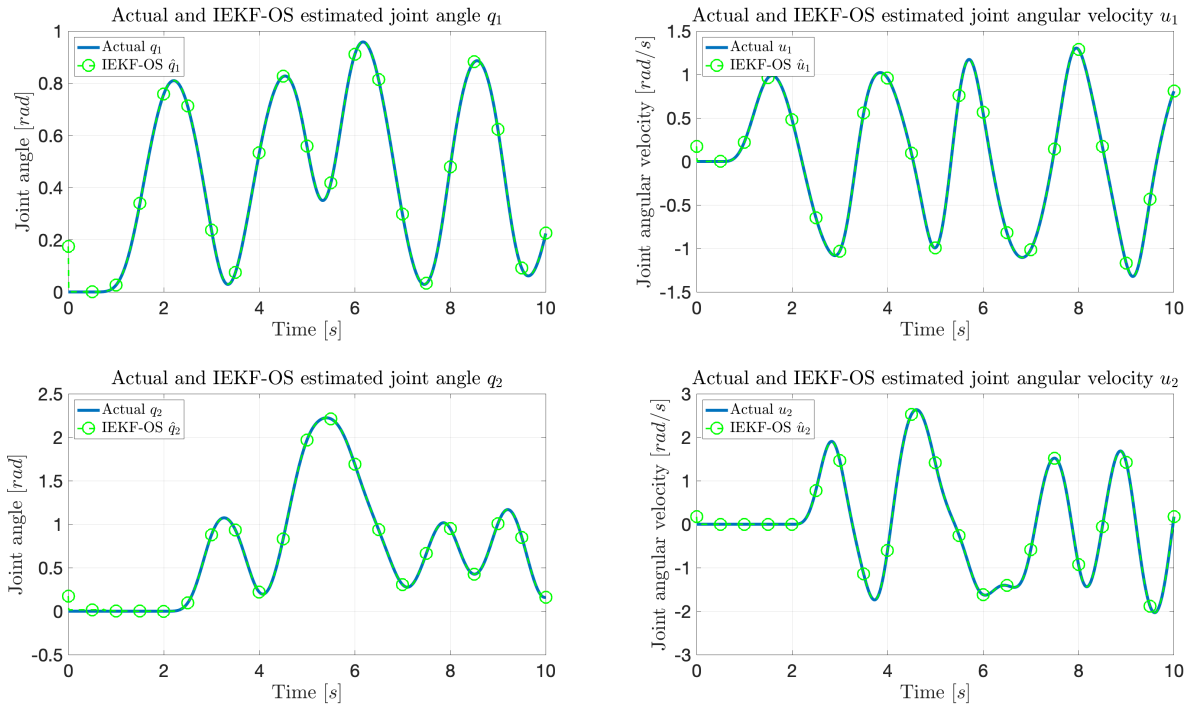


Figure C-4: The results obtained for the reconstructed motion. The left column of plots illustrates the two joint angles, q_1 and q_2, and the right column of plots shows the joint angular velocities, u_1 and u_2. From these plots, it can be seen that the estimated state variables, dashed lines, are completely following the actual state evolutions illustrating near-perfect motion reconstruction.

From Figure C-5, it can be seen that the idea of modeling the joint torque τ as a random-walk model, as defined in Section 3-5-2, works neatly. In the reconstructed joint torque signals,

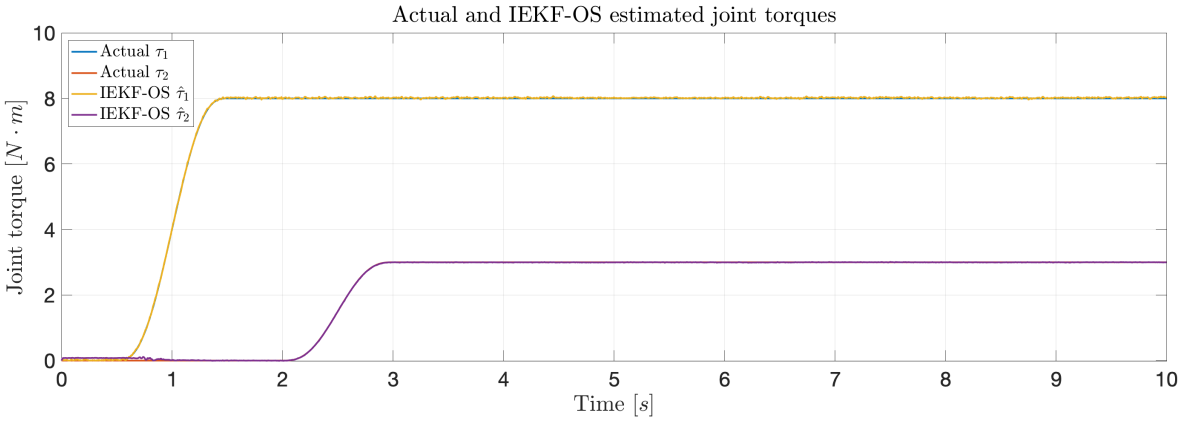


Figure C-5: The two control torque step functions $f(t)_{\tau_1}$ and $f(t)_{\tau_2}$ applied to the joint angles q_1 and q_2 respectively and their corresponding IEKF-OS estimated joint torques.

the overall form of the originally applied step function is clearly visible indicating accurate reconstruction.

Appendix D

OpenSim model of the KUKA LBR iiwa 7 R800

D-1 Inertial properties of the links of the KUKA

As outlined in Section 4-2, the OpenSim model of the KUKA LBR iiwa 7 R800 robot was modeled in `XML` code using the inertial properties as given in the GitHub repository of Chatzilygeroudis et al. (2019) [47]. Subsequently, the joint between two rigid bodies was modeled as an OpenSim `PinJoint`. This joint allows the child body to rotate with respect to the parent joint it is connected to. Geometry files of all the links were included which are also available on the repository of Chatzilygeroudis et al. (2019). Table D-1, adapted from this GitHub file, illustrates the inertial properties of the KUKA robot.

Table D-1: The inertial properties of the KUKA LBR iiwa 7 R800 robot adapted from [47].

Link	0	1	2	3	4	5	6	7
CoM (XYZ) [m]	(-0.1 0 0.07)	(0 -0.03 0.12)	(0.0003 0.059 0.042)	(0 0.03 0.13)	(0 0.067 0.034)	(0.0001 0.021 0.076)	(0 0.0006 0.0004)	(0 0 0.02)
Mass [kg]	5	3.4525	3.4821	4.05623	3.4822	2.1633	2.3466	3.129
I_{xx} [kg m^2]	0.05	0.02183	0.02076	0.03204	0.02178	0.01287	0.006509	0.01464
I_{xy} [kg m^2]	0	0	0	0	0	0	0	0.0005912
I_{xz} [kg m^2]	0	0	-0.003626	0	0	0	0	0
I_{yy} [kg m^2]	0.06	0.007703	0.02179	0.00972	0.02075	0.005708	0.006259	0.01465
I_{yz} [kg m^2]	0	-0.003887	0	0.006227	-0.003625	-0.003946	0.00031891	0
I_{zz} [kg m^2]	0.03	0.02083	0.00779	0.03042	0.007785	0.01112	0.004527	0.002872

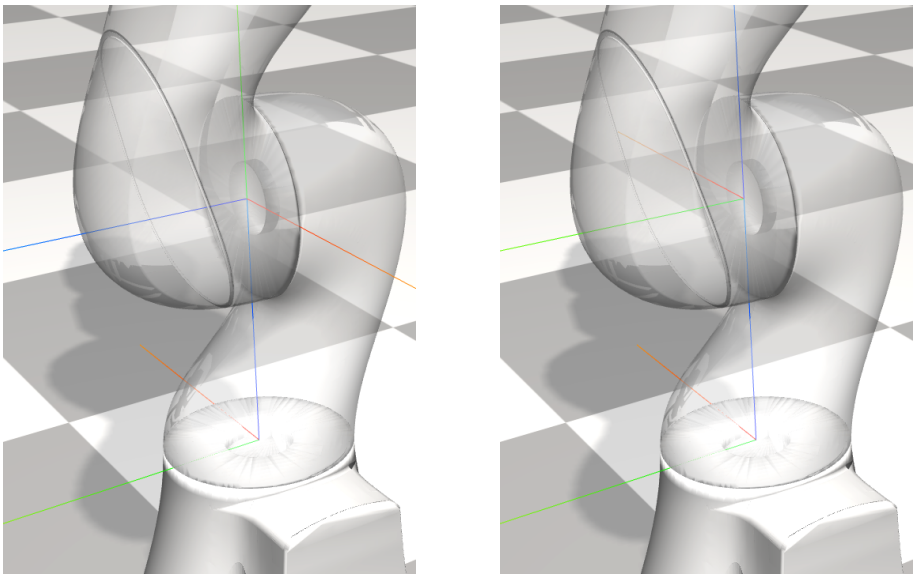
For the interested reader, the ranges of motion and angular velocities of each individual joint are shown in Table D-2 This table was adapted from [55].

Table D-2: The ranges of motion and the angular velocities of each KUKA link.

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7
Range of motion [deg]	±170	±120	±170	±120	±170	±120	±175
Speed with rated payload [deg/s]	98	98	100	130	140	180	180

D-2 Locations and orientations of the virtual IMU frames on the KUKA OpenSim model for the simulation-based verification

The virtual IMU frame, modeled as a `PhysicalOffsetFrame`, is created by defining an offset translation vector and an offset rotational vector with respect to its parent coordinate frame and expressed in the parent coordinate frame. For the actual experiment, it is important that these two offset vectors are accurately defined. Determining these offset values is performed using the calibration procedure as outlined in Appendix F for calibrating Xsens MTw IMUs to an OpenSim model. This procedure relies on having all the parent coordinate frames, from which virtual IMU frames are defined, aligned in the same orientation with respect to an initially chosen base frame. Moreover, during this calibration, it is also important that the KUKA robot is fully extended in its upright position, all joint angles $q = 0$ [rad]. Of course, when one models an OpenSim model, these parent coordinate frames, located in the body's joint center, will not be aligned in the same orientation with this base frame. For that reason, intermediate frames, also modeled as `PhysicalOffsetFrame`s, were defined with respect to the misaligned body frame b. As such that their orientation matches the orientation of the base frame. This is illustrated in Figures D-1a and D-1b.



(a) **(b)**

Figure D-1: (a) The lower frame is the initial base frame chosen. It has its X-axis (red) pointing forwards, its Y-axis (green) pointing to the left, and its Z-axis (blue) pointing up. The body frame b_2, located in the joint center of link 2, instead has its X-axis pointing backward, its Y-axis pointing up, and its Z-axis pointing left. Thus for this joint, an intermediate frame needs to be defined which is shown in Figure (b).
(b) The upper frame, the intermediate frame modeled, has its orientation congruent to the initially chosen base frame as desired. To obtain this congruent orientation of this intermediate frame, the intermediate frame was configured as such using `set_orientation(Vec(`$-\pi/2, 0, \pi$`))` with respect to its parent frame b_2, located in the joint center of link 2.

For each joint, one needs to check if its corresponding joint center frame has its orientation

congruent to the base frame. If not, one needs to create an intermediate frame as shown in Figures D-1a and D-1b. With this idea illustrated, Table D-3 depicts the orientations and translations of the virtual IMUs, as well as the defined KUKA required intermediate frames.

Table D-3: The translation and rotations of the defined frames all with respect to and expressed in the corresponding parent coordinate frames.

	Virtual IMU 1	Intermediate frame 2	Virtual IMU 2	Virtual IMU 3	Intermediate frame 4
Translation [m]	(0, -0.1, 0.07)	(0, 0, 0)	(0, 0.095, 0.12)	(0, 0.095, 0.09)	(0, 0, 0)
Rotation [rad]	(π/2+0.3, 0, π/2)	($-\pi$/2, 0, π)	($-\pi$/2+0.3, 0, -π/2)	($-\pi$/2-0.4, 0, -π/2)	($-\pi$/2, 0, 0)
Parent frame	Body frame b_1	Body frame b_2	Intermediate frame 2	Body frame b_3	Body frame b_4
	Virtual IMU 4	**Intermediate frame 5**	**Virtual IMU 5**	**Intermediate frame 6**	**Virtual IMU 6**
Translation [m]	(0, -0.09, 0.13)	(0, 0, 0)	(0, -0.095, 0.11)	(0, 0, 0)	(0, 0.12, 0.07)
Rotation [rad]	(π/2-0.3, 0, π/2)	(0, 0, π)	(π/2+0.3, 0, π/2)	($-\pi$/2, 0, π)	($-\pi$/2+0.15, 0, -π/2)
Parent frame	Intermediate frame 4	Body frame b_5	Intermediate frame 5	Body frame b_6	Intermediate frame 6

Although this was a simulation, still these locations and orientations were chosen such that they were not obstructing the ranges of motion of the links of the KUKA. Figure D-2 shows all these virtual IMUs modeled as frames.

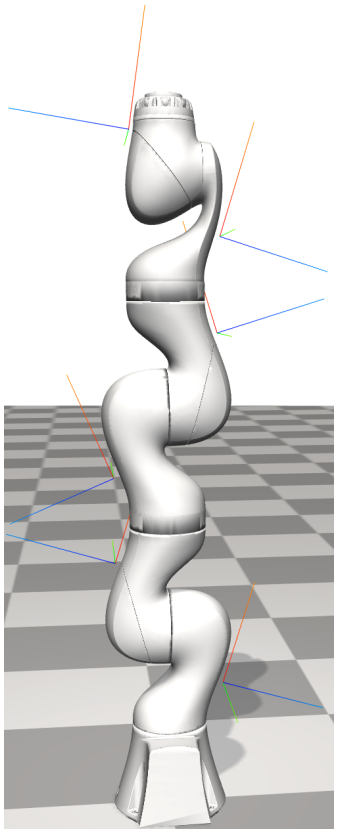


Figure D-2: The OpenSim KUKA model with all the virtual IMUs being modeled as frames.

Appendix E

Experiment results

E-1 Results for the other IEKF-OS validation trials

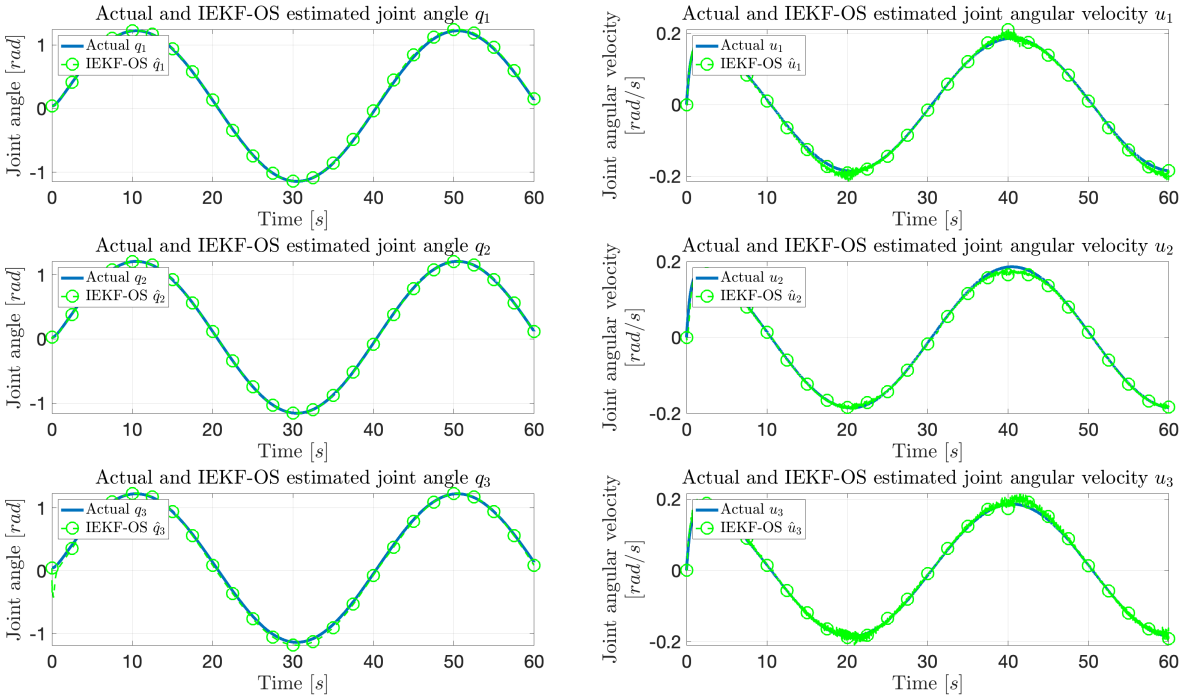


Figure E-1: Trial 1 with sine waves applied to each joint with a frequency of 0.025 [Hz] and an amplitude of 1.2 [rad]. Depicted are the estimated and actual joint angles q_1, q_2, and q_3 (left column), and estimated and actual joint angular velocities u_1, u_2, and u_3(right column).

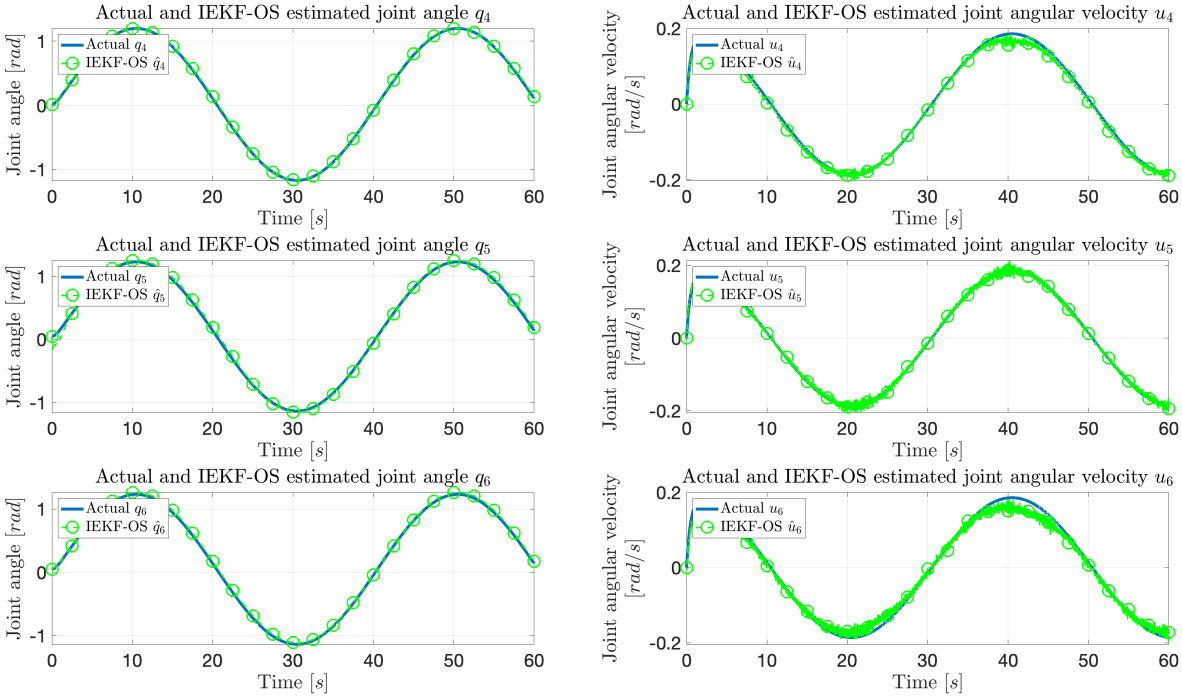


Figure E-2: Trial 1 with sine waves applied to each joint with a frequency of 0.025 [Hz] and an amplitude of 1.2 [rad]. Depicted are the estimated and actual joint angles q_4, q_5, and q_6 (left column), and estimated and actual joint angular velocities u_4, u_5, and u_6 (right column).

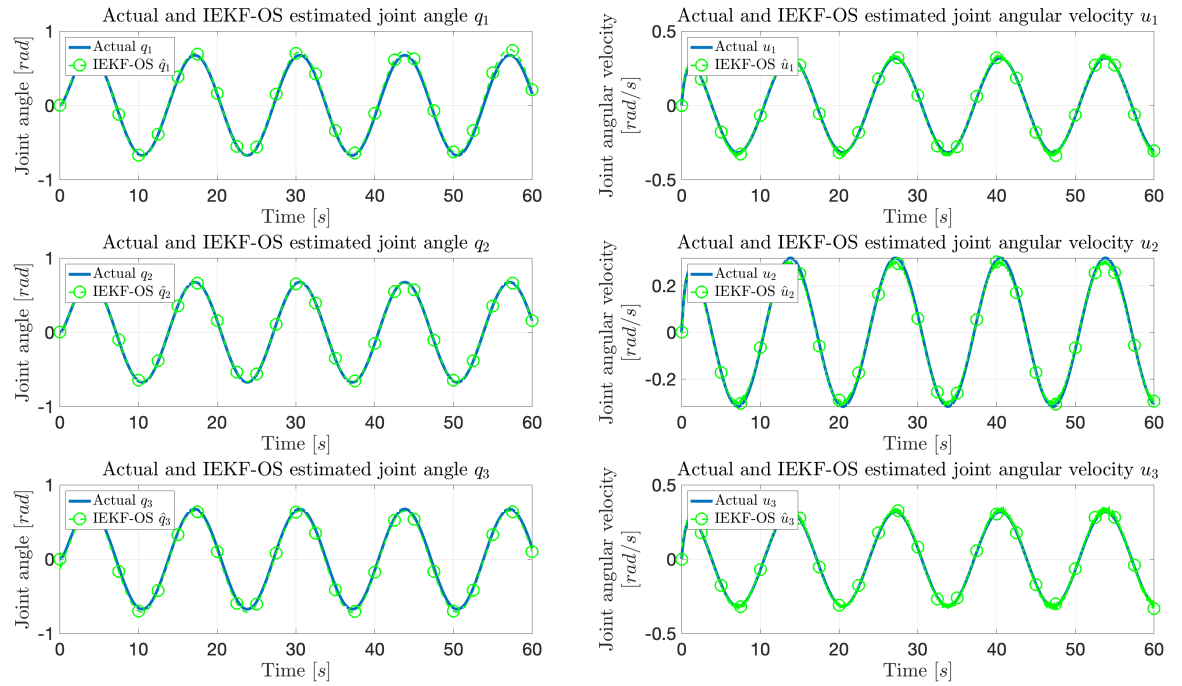


Figure E-3: Trial 3 with sine waves applied to each joint with a frequency of 0.075 [Hz] and an amplitude of 0.7 [rad]. Depicted are the estimated and actual joint angles q_1, q_2, and q_3 (left column), and estimated and actual joint angular velocities u_1, u_2, and u_3 (right column).

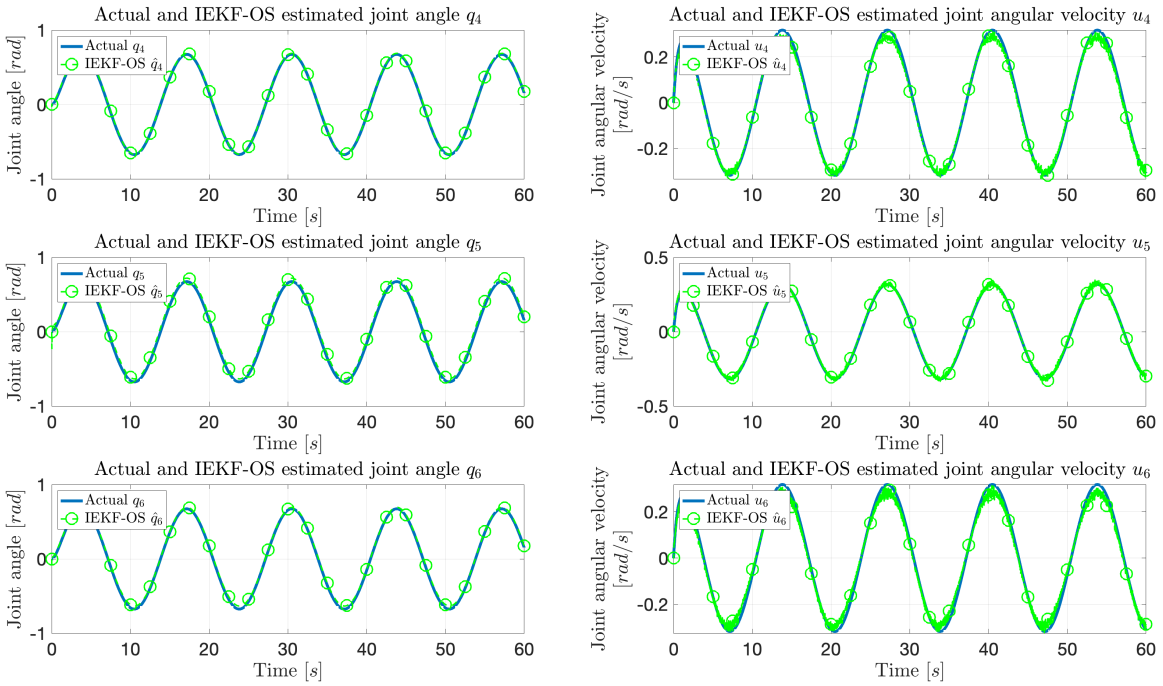


Figure E-4: Trial 3 with sine waves applied to each joint with a frequency of 0.075 [Hz] and an amplitude of 0.7 [rad]. Depicted are the estimated and actual joint angles q_4, q_5, and q_6 (left column), and estimated and actual joint angular velocities u_4, u_5, and u_6 (right column).

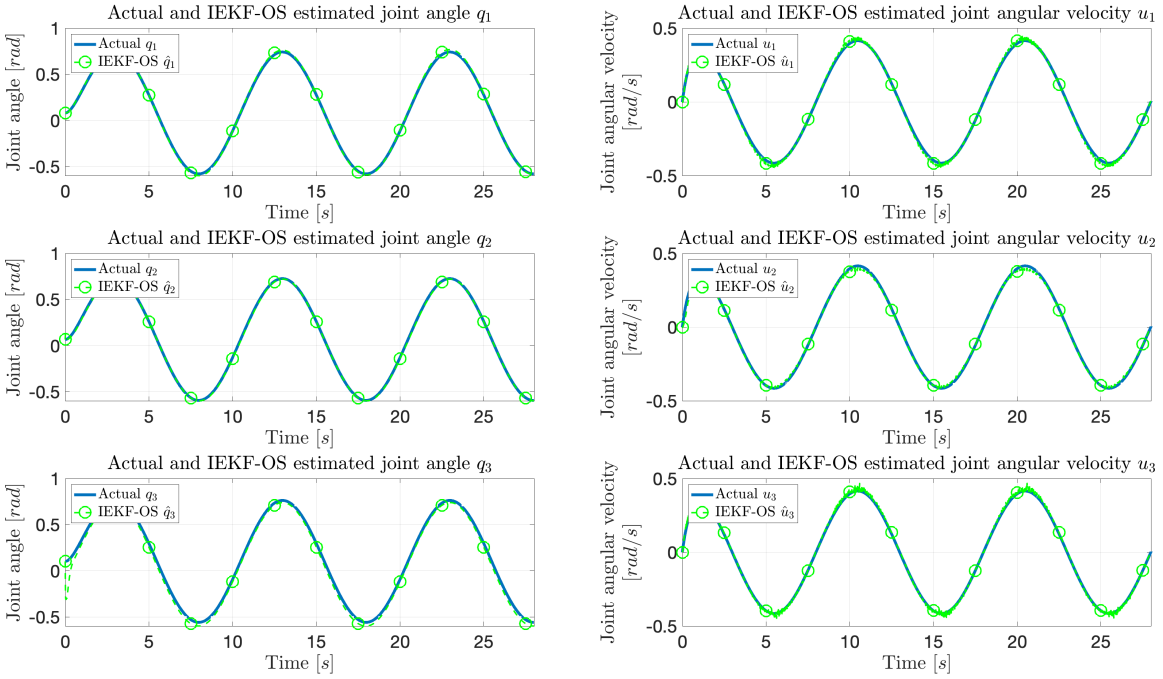


Figure E-5: Trial 4 with sine waves applied to each joint with a frequency of 0.1 [Hz] and an amplitude of 0.7 [rad]. Depicted are the estimated and actual joint angles q_1, q_2, and q_3 (left column), and estimated and actual joint angular velocities u_1, u_2, and u_3 (right column).

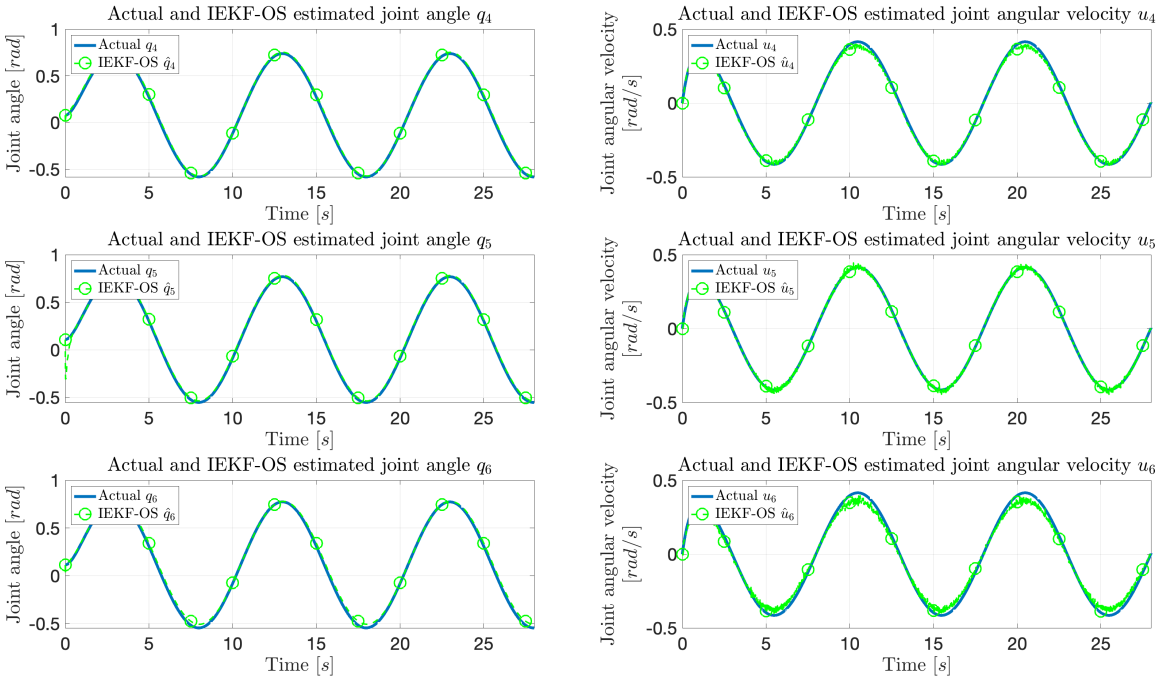


Figure E-6: Trial 4 with sine waves applied to each joint with a frequency of 0.1 [Hz] and an amplitude of 0.7 [rad]. Depicted are the estimated and actual joint angles q_4, q_5, and q_6 (left column), and estimated and actual joint angular velocities u_4, u_5, and u_6 (right column).

E-2 Results for the other trials comparing the IEKF-OS algorithm with OpenSense-Madgwick

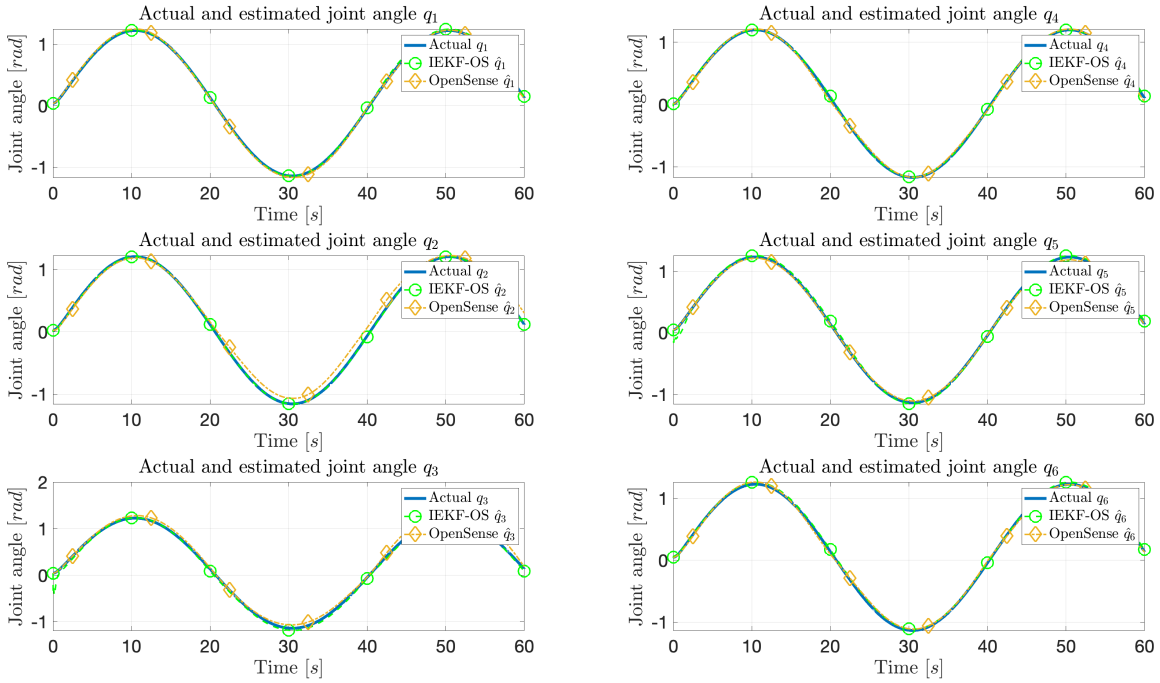


Figure E-7: Trial 1 with sine waves applied to each joint with a frequency of 0.025 [Hz] and an amplitude of 1.2 [rad]. Comparison between actual joint encoder values shown in blue, IEKF-OS joint angle estimations shown in green, and OpenSense joint angle estimations shown in orange.

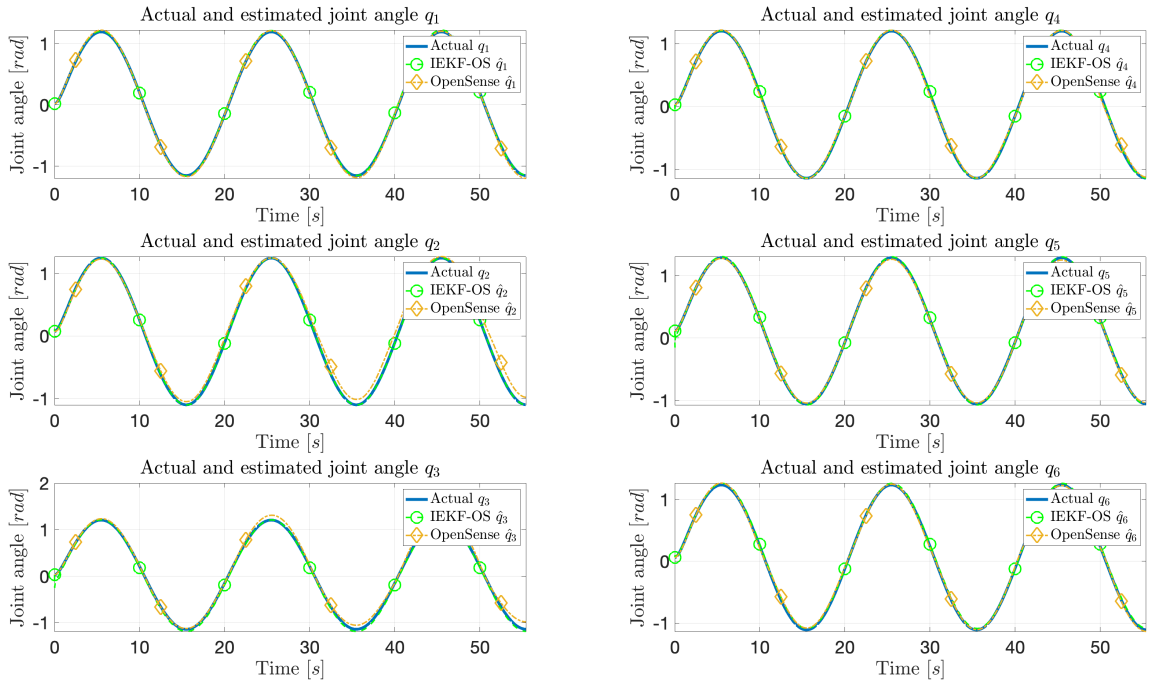


Figure E-8: Trial 2 with sine waves applied to each joint with a frequency of 0.05 [Hz] and an amplitude of 1.2 [rad]. Comparison between actual joint encoder values shown in blue, IEKF-OS joint angle estimations shown in green, and OpenSense joint angle estimations shown in orange.

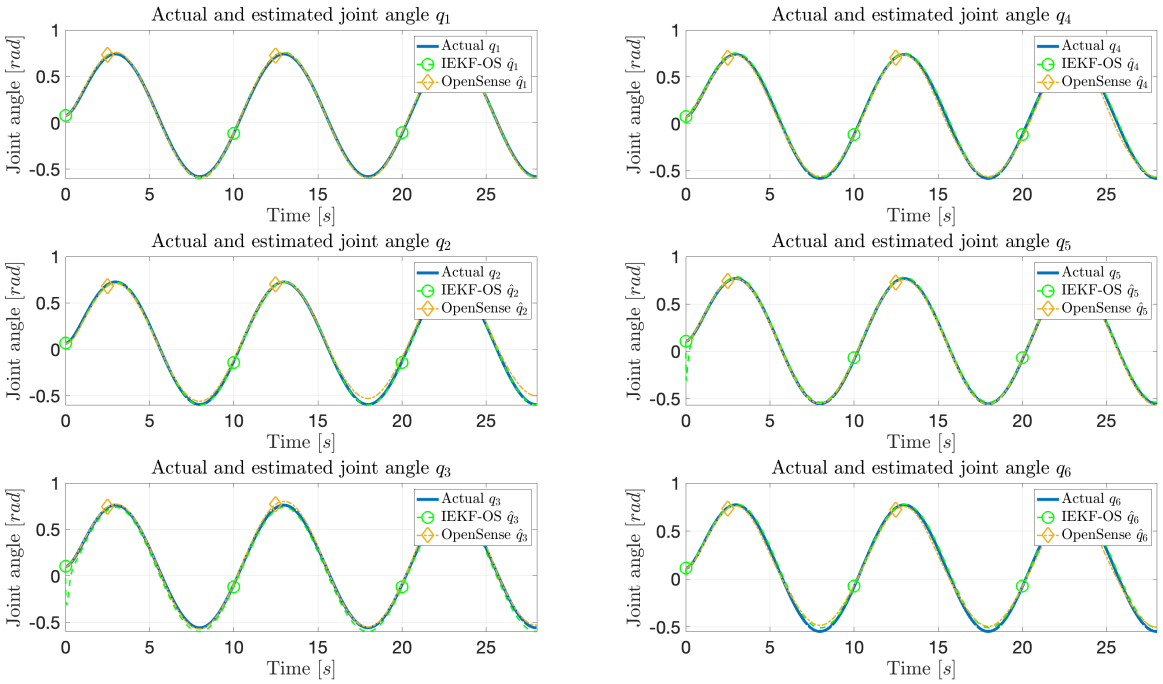


Figure E-9: Trial 4 with sine waves applied to each joint with a frequency of 0.01 [Hz] and an amplitude of 0.7 [rad]. Comparison between actual joint encoder values shown in blue, IEKF-OS joint angle estimations shown in green, and OpenSense joint angle estimations shown in orange.

Appendix F

Calibration of IMUs

F-1 Detailed workflow of calibration procedure

For future users who want to use the IEKF-OS algorithm, it is important to accurately register and calibrate Xsens IMUs to the desired OpenSim model representing the system at hand. In Section 3-2-3, the mathematical principles of calibrating the gyroscope and accelerometer are presented. In this section, a more practical workflow is presented consisting of four to five steps depending on the system from which the motion needs to be reconstructed. In case of a system that consists of ferromagnetic materials such as iron or steel, the accuracy of the heading estimates of the Xsens Kalman Filter for 3D human motion (XKF3hm) may decrease [48]. In such a case, when one attaches Xsens IMUs to ferromagnetic materials, one may use the Xsens Magnetic Field Mapper to perform calibration of the on-board magnetometer [51]. As this procedure is extensively outlined in [51], it won't be detailed here. Hence, it is assumed from here on that calibration of the magnetometers has been accounted for. Lastly, it is also assumed that the user has access to Xsens MTw Awinda wireless IMU sensor [11]. Besides, it is assumed that the user also has access to the Xsens MT Manager software which ensures time syncing between connected IMUs, sensor fusion, and data interpolation for missing entries.

Detailed list of the workflow for capturing Xsens IMU measurements

Gathering the data necessary to reconstruct the desired motion consists of four steps. Assuming that the magnetometer has been calibrated, these steps are summarized in the list below.

1. Log stationary data to estimate the gyroscope bias.
2. Log rotational data to estimate the accelerometer bias and misalignment matrix.
3. Perform Xsens IMU to rigid body registration and orientation calibration.
4. Log the raw inertial data during the motion of interest.

Hence, before one wants to gather data of reconstructing the motion of interest, three steps need to be performed. It is recommended, prior to doing these calibration steps, to create an organized folder structure. This facilitates performing these calibration steps and minimizes the possibility of making mistakes. An example could look like

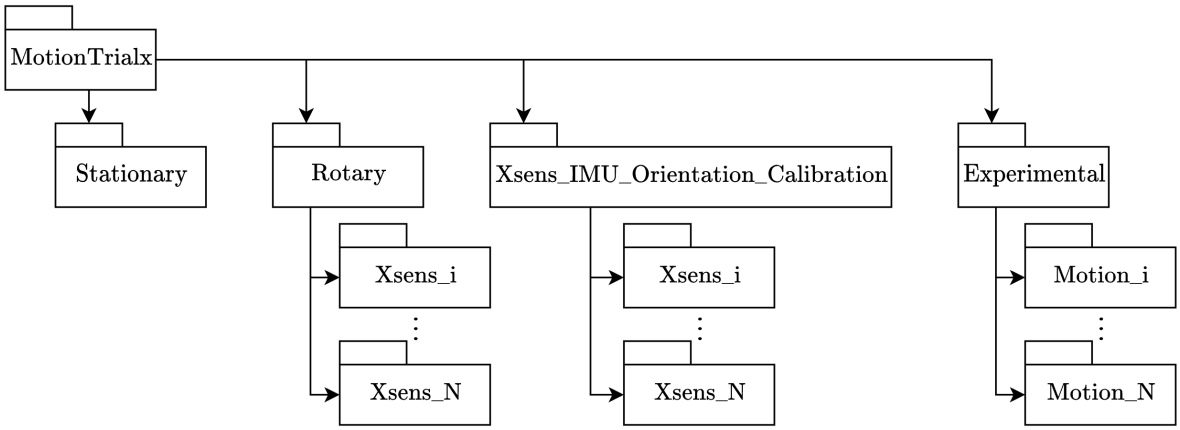


Figure F-1: Recommended folder structure for performing the calibration procedure.

Moreover, the author is convinced that having a clear detailed list like is presented here will for a large part avoid making mistakes. Especially when a large set of sensors is used for which it will become more difficult to perform all necessary calibration steps for each sensor. Therefore, all the performed steps, as were also carried out when actual experiments for this thesis were conducted, will be outlined in a systematic fashion. Lastly, it is also recommended to label each IMU and clearly have in mind which IMU will eventually be attached to which body.

Log stationary data to estimate the gyroscope bias

With this folder structure created, the first step is to perform the stationary calibration to determine the gyroscope bias.

1. In the MT manager software, choose the folder *Stationary*.
2. Connect the set of MTw's to the Xsens MT manager software.
3. Place all the IMUs on a flat table.
4. Start the measurement.
5. Log the stationary data for about 60 seconds.
6. Stop the measurement and disconnect the IMUs.
7. Write the data to the folder *Stationary*.
    (a) Make sure to select the angular velocity option when writing data to a file as this measurement will be used for determining the gyroscope bias.
8. Perform the gyroscope calibration for all IMUs as is outlined in Section 3-2-3.

Log rotary data to estimate the accelerometer bias and calibration matrix

With the calibration of the gyroscope performed, the next step is to determine the accelerometer bias and calibration matrix for each IMU's accelerometer individually. Whereas in the previous calibration, all the sensors were connected simultaneously, in this calibration, each IMU needs to be connected solely and then calibrated individually. As was already outlined in Section 3-2-3, this calibration makes use of the ellipsoid fitting method [32]. For this calibration, it is essential to configure the sensor in as many orientations as possible by making sure to rotate the sensor between orientations using approximately a constant velocity. To that aim, one can make an orientation prism in which the sensor is kept stationary for a certain orientation. This also facilitates the requirement of slowly rotating the sensor between orientations. With these details in mind, the procedure for capturing the data to calibrate the accelerometer is presented in the list below.

For each sensor in the set of sensors

1. In the MT manager software, choose the folder *Rotary/Xsens_i* where i denotes one of the sub-folders corresponding to the i-th sensor.
2. Place the i-th sensor in the orientation prism.
3. Start the measurement.
4. Perform the rotations in the orientation cube as slowly as possible. Each session should last about 2-3 minutes to capture enough data and in different orientations.
5. Stop the measurement and disconnect the i-th sensor.
6. Write the data to the folder *Rotary/Xsens_i*.
    (a) Make sure to select the linear acceleration option when writing data to a file as this measurement will be used for determining the accelerometer bias and the calibration matrix.
7. Perform the i-th IMU accelerometer calibration as is outlined in Section 3-2-3.

Perform Xsens IMU to rigid body registration and orientation calibration.

Having collected the data necessary to calibrate both the gyroscope and the accelerometer, the most important step of this calibration workflow will now be detailed. The IEKF-OS algorithm assumes that the orientation of the i-th virtual IMU frame, attached to the i-th body of the OpenSim model of the system at hand, matches the orientation of the i-th Xsens IMU attached to the i-th body of the actual system. When for a system's given configuration, a large difference in orientation between the modeled virtual IMU frame and the Xsens IMU frame is present, the motion reconstruction performance using the IEKF-OS algorithm is deteriorated. Hence, it is key to estimate the orientation of the Xsens IMU on the rigid body of the system as best as possible. The XKF3hm orientation estimation algorithm of Xsens [48] can be used to determine the orientation of the sensor with respect to the world frame. This world frame, however, does not coincide with the ground frame of OpenSim.

For that reason, an inclination and heading reset will be performed when the Xsens IMU is oriented similar to the chosen base frame. The notion of this base frame is illustrated in Section D-2. With this inclination and heading reset performed, the Xsens IMU will now, instead of computing its orientation with respect to the world frame, estimate its orientation with respect to this base frame. Note that, prior to performing this inclination and heading reset, one must carefully align the Xsens IMU's axes with the axes of the chosen base frame. Another important assumption is that the system has all its generalized coordinates equal to zero, $q = 0$ [rad]. For the system at hand in this thesis, the KUKA robot, this means that the robot should be fully extended in its upright position. This requirement was validated by reading of the values of the joint encoders.

With these important assumptions outlined, the procedure to log the required data for performing Xsens IMU orientation calibration is shown in the list below. Again, this procedure needs to be performed for each Xsens IMU individually.

1. In the MT manager software, choose the folder *Xsens_IMU_Orientation_Calibration/Xsens_i* where i denotes one of the sub-folders corresponding to the i-th sensor and establish a connection with this i-th IMU.
2. Carefully align the axes of the i-th Xsens IMU sensor with the axes of the chosen base frame.
3. Perform an inclination reset.
   (a) Roll and pitch estimates should be approximately zero now.
4. Perform a heading reset.
   (a) Yaw estimate should be approximately zero now.
5. Start recording.
6. Pick up the Xsens IMU and attach the IMU to the desired i-th link.
7. Stop recording when the Xsens IMU is attached to the corresponding link.
8. Write the orientation, parametrized as rotation matrix data of the sensor, to the folder: *Xsens_IMU_Orientation_Calibration/Xsens_i* where i denotes one of the sub-folders corresponding to the i-th sensor.
9. **Important**, disconnect the Xsens IMU, as otherwise all quantities logged by the IMU will be measured with respect to this base frame!
   (a) For the actual inertial data logging during the experiment, the IMUs should measure with respect to the world frame again, not with respect to the self-defined base frame.
10. Open the last nine values of the logged rotation matrix and store these values in a matrix in `MATLAB` as shown in F-1

$$RotMatrix\_Link\_i = \begin{pmatrix} Mat[1][1] & Mat[1][2] & Mat[1][3] \\ Mat[2][1] & Mat[2][2] & Mat[2][3] \\ Mat[3][1] & Mat[3][2] & Mat[3][3] \end{pmatrix}. \tag{F-1}$$

To obtain the space fixed Euler angles from this orientation needed to configure the virtual IMU on the OpenSim model of the system at hand, the function `rotm2eul(RotMatrix_Link_i, 'XYZ')` was used. The sequence XYZ needs to be set as the option here, as this is the space fixed Euler angle sequence from IMU space to OpenSim space.

As an example of this whole procedure, the following code can be used to obtain the required Euler angles that define how the virtual IMU should be configured on the OpenSim model.

```
%% Create a struct to store all the rotation and translation parameters.
Orientation = struct;
% ------------------------------------------------------------------------

%% KUKA Link 1
% Look at the last row of the file:
% Xsens_IMU_Orientation_Calibration/Xsens_1/OrientationData1.txt

% Determine the rotation parameters of Xsens IMU 1.
RotMatrix_Link_1 = [-0.009910  -0.999508  -0.029753
                    -0.219128   0.031202  -0.975197
                     0.975646  -0.003145  -0.291330];
eul_Link1 = rotm2eul(RotMatrix_Link_1, 'XYZ');

Orientation.Xsens_1_Rot_X = eul_Link1(1,1);
Orientation.Xsens_1_Rot_Y = eul_Link1(1,2);
Orientation.Xsens_1_Rot_Z = eul_Link1(1,3);
% ------------------------------------------------------------------------
```

The first step of this calibration procedure has now been completed. The last step is to determine the XYZ translations from the center of the coordinate frame of the parent body b towards the center of the coordinate frame of the Xsens IMU. This latter frame corresponds to the position of the accelerometer in the Xsens MTw Awinda IMU as shown in Figure F-2. These translations need to be expressed in the coordinate frame of the i-th parent body b to which this i-th Xsens IMU will be attached. As in this schematic, the accelerometer position of the MTw Awinda has been defined with respect to the lower-left corner, the user needs to manually measure, e.g., using a tape measure, the translations from the parent body coordinate frame towards the lower-left corner of the MTw Awinda.

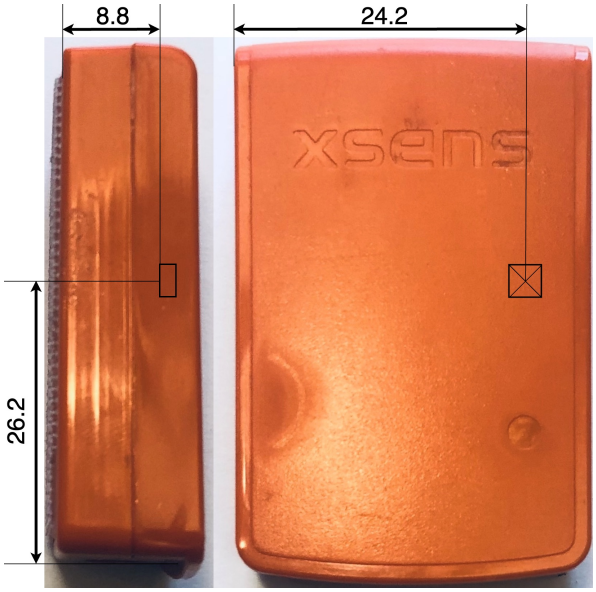


Figure F-2: Schematics of the accelerometer position in the Xsens MTw Awinda IMU. Distances are shown in [mm]. Figure adapted from [48].

Again, an example is illustrated in the following code to define these translational offsets expressed in the coordinate frame of the i-th body.

```
%% Determine the translation parameters of Xsens IMU 1.
% XYZ translation measurements are approximately and measured toward
% the left down corner of the IMU and expressed in the base frame
% coordinates.

% Add the constant offset parameters from the left down corner to the
% accelerometer triad location. This information is obtained from
% the Xsens MTw manual page 64, section 11.1.5.
% Offset X expressed in OpenSim base frame with
% IMU pointing as follows: X upwards, Y to the left, and Z towards you.
% Hence, for the actual translations, check if these constants need to be
% added or subtracted!
% Express in meters!
Orientation.Xoffset = 0.0242; % Meters
Orientation.Yoffset = 0.0088; % Meters
Orientation.Zoffset = 0.0262; % Meters

% Translations must be expressed in meters in OpenSim:
Orientation.Xsens_1_Trans_X = -0.015 + Orientation.Xoffset;
Orientation.Xsens_1_Trans_Y = -0.071 - Orientation.Yoffset;
Orientation.Xsens_1_Trans_Z =  0.005 + Orientation.Zoffset;
% ------------------------------------------------------------------------
```

Finally, with both the rotational and translational parameters defined for the i-th IMU, the virtual IMU can be created using a combination of OpenSim and MATLAB code as is illustrated below.

```
%% Import OpenSim Libraries into Matlab.
import org.opensim.modeling.*
% ------------------------------------------------------------------------


%% Create the virtual IMU modeled as a PhysicalOffsetFrame.
IMU1 = PhysicalOffsetFrame();
% Set the name of the IMU.
IMU1.setName('FirstIMU');
% Select the body to which it is attached.
IMU1.setParentFrame(Link1);
% Set the (XYZ) translation vector expressed in parent coordinate frame.
IMU1.set_translation(Vec3(Orientation.Xsens_1_Trans_X,
                          Orientation.Xsens_1_Trans_Y,
                          Orientation.Xsens_1_Trans_Z)); % Translate
% Set the (XYZ) rotation vector expressed in parent coordinate frame.
IMU1.set_orientation(Vec3(Orientation.Xsens_1_Rot_X,
                          Orientation.Xsens_1_Rot_Y,
                          Orientation.Xsens_1_Rot_Z));   % Rotate
% Add the virtual IMU to the rigid body.
Link1.addComponent(IMU1);
% ------------------------------------------------------------------------
```

Log the raw inertial data during the motion of interest

When all the calibration steps have been performed as described in the previous sections, the desired motion can be logged. It is very important that prior to this experiment, all the Xsens IMUs are disconnected from the MT Manager. Failure in doing so results in the fact that the MTw's will log all raw inertial data of interest, angular velocities, and linear accelerations in the previously set base frame. The MTw Awinda does not store the heading and inclination resets on board [48]. Hence, disconnecting the Xsens IMUs and then connecting them again or performing a re-scan ensures that the MTw Awinda stores the inertial data again according to its body-fixed sensor coordinate system. The procedure of logging data is again listed below.

1. Select the folder *Experimental/Motion_i*.
2. Start up the Xsens MTw Awinda IMUs.
    (a) Start moving the system at hand such that the IMUs become excited hence findable again.
3. Make sure that all the desired IMUs are connected to the MT Manager.
4. Start recording.
5. Perform the motion that needs to be reconstructed later using the IEKF-OS algorithm.

6. Stop recording when the motion is completed.
7. Write the logged data to the folder *Experimental/Motion_i*, where i denotes the number of the trial.
   (a) Store all the data.
      - Angular velocity.
      - Linear acceleration.
      - Magnetometer data.
      - Orientation data: rotation matrix, quaternion, Euler angles.
8. Possibly, perform another trial and start at the top of this list again.

Appendix G

Technical Specifications Xsens MTw Awinda IMU

MTw performance

Table G-1: The technical specifications of the Xsens MTw Awinda gyroscope, accelerometer, and magnetometer. Table adapted from [48].

	Angular velocity	Linear acceleration	Magnetic field
Dimensions	3 axes	3 axes	3 axes
Full scale	± 2000 [deg/s]	± 160 [m/s^2]	± 1.9 [Gauss]
Nonlinearity	0.1% of full scale	0.1% of full scale	0.1% of full scale
Bias stability	10 [deg/hr]	0.1 [mg]	-
Noise	0.01 [deg/s/$\sqrt{Hz}$]	200 [$\mu g\sqrt{Hz}$]	0.2 [mGauss/$\sqrt{Hz}$]
Alignment error	0.1 [deg]	0.1 [deg]	0.1 [deg]
Bandwidth	180 [Hz]	180 [Hz]	10-60 [Hz]

MTw orientation performance

Table G-2: Orientation performance of the Xsens MTw Awinda in a magnetically undisturbed environment. Table adapted from [48].

Dynamic range	All angles in 3D
Static accuracy (Roll/Pitch)	0.5 deg RMS
Static accuracy (Heading)	1 deg RMS
Dynamic accuracy (Roll/Pitch)	0.75 deg RMS
Dynamic accuracy (Heading)	1.5 deg RMS

MTw physical properties

Table G-3: Physical properties of the Xsens MTw Awinda. Table adapted from [48].

Accelerometers	MEMS solid-state, capacitive readout.
Rate gyroscope	MEMS solid-state, capacitive readout.
Magnetometer	Magneto-Impedance sensor elements.
Barometer	Piezo-resistive sensor element.
Weight	16 [g]
Housing dimensions	$47 \times 30 \times 13$ [mm]

MTw update rates and retransmission slots

Table G-4: Update rates and available retransmission slots. Table adapted from [48].

Amount of Xsens MTw Awinda's	Update rate [Hz]	Available retransmission slots
1-5	120	2
6-9	100	2
10	80	3
11-20	60	4
21-32	40	1

Bibliography

[1] A. K. Ingale and D. U. J., "Real-time 3D reconstruction techniques applied in dynamic scenes: A systematic literature review," *Computer Science Review*, vol. 39, p. 100338, 2021.

[2] Vicon Motion Systems Ltd UK, "What is Motion Capture?." https://www.vicon.com/about-us/what-is-motion-capture/. Accessed: 13-01-2021.

[3] N. Barbour and G. Schmidt, "Inertial sensor technology trends," *IEEE Sensors Journal*, vol. 1, no. 4, pp. 332–339, 2001.

[4] Xsens Technologies B.V., "Home - Xsens 3D motion tracking." https://www.xsens.com/. Accessed: 13-01-2021.

[5] A. Cappozzo, U. Della Croce, A. Leardini, and L. Chiari, "Human movement analysis using stereophotogrammetry. Part 1: Theoretical background," *Gait and Posture*, vol. 21, no. 2, pp. 186–196, 2005.

[6] E. Dorschky, M. Nitschke, A. K. Seifer, A. J. van den Bogert, and B. M. Eskofier, "Estimation of gait kinematics and kinetics from inertial sensor data using optimal control of musculoskeletal models," *Journal of Biomechanics*, vol. 95, p. 109278, 2019.

[7] B. H. Koning, M. M. van der Krogt, C. T. Baten, and B. F. Koopman, "Driving a musculoskeletal model with inertial and magnetic measurement units," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 18, no. 9, pp. 1003–1013, 2015.

[8] L. Tagliapietra, L. Modenese, E. Ceseracciu, C. Mazzà, and M. Reggiani, "Validation of a model-based inverse kinematics approach based on wearable inertial sensors," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 21, no. 16, pp. 834–844, 2018.

[9] K. Aminian and B. Najafi, "Capturing human motion using body-fixed sensors: Outdoor measurement and clinical applications," *Computer Animation and Virtual Worlds*, vol. 15, no. 2, pp. 79–94, 2004.

[10] M. Kok, J. D. Hol, and T. B. Schön, "Using Inertial Sensors for Position and Orientation Estimation," *Foundations and Trends in Signal Processing*, vol. 11, no. 1-2, pp. 1–153, 2017.

[11] M. Paulich, M. Schepers, N. Rudigkeit, and G. Bellusci, "Xsens MTw : Miniature Wireless Inertial Motion Tracker for Highly Accurate 3D Kinematic Applications," *Xsens Technologies*, no. April, pp. 1–9, 2013.

[12] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[13] D. Roetenberg, H. Luinge, and P. Slycke, "Xsens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors," *Xsens Motion Technologies BV*, pp. 1–7, 2009.

[14] M. Kok, J. D. Hol, and T. B. Schön, "An optimization-based approach to human body motion capture using inertial sensors," in *Proceedings of the 19th World Congress The International Federation of Automatic Control*, vol. 19, pp. 79–85, 2014.

[15] I. Weygers, M. Kok, H. De Vroey, T. Verbeerst, M. Versteyhe, H. Hallez, and K. Claeys, "Drift-Free Inertial Sensor-Based Joint Kinematics for Long-Term Arbitrary Movements," *IEEE Sensors Journal*, vol. 20, no. 14, pp. 7969–7979, 2020.

[16] S. J. Hall, *Basic Biomechanics*. New York: McGraw-Hill Higher Education, eighth ed., 2019.

[17] A. Karatsidis, M. Jung, H. M. Schepers, G. Bellusci, M. de Zee, P. H. Veltink, and M. S. Andersen, "Musculoskeletal model-based inverse dynamic analysis under ambulatory conditions using inertial motion capture," *Medical Engineering and Physics*, vol. 65, pp. 68–77, 2019.

[18] T. Seel, T. Schauer, and J. Raisch, "Joint Axis and Position Estimation from Inertial Measurement Data by Exploiting Kinematic Constraints," *Proceedings of the IEEE International Conference on Control Applications*, pp. 45–49, 2012.

[19] F. Olsson and K. Halvorsen, "Experimental evaluation of joint position estimation using inertial sensors," in *Proceedings of the 20th International Conference on Information Fusion*, 2017.

[20] F. Olsson, T. Seel, D. Lehmann, and K. Halvorsen, "Joint Axis Estimation for Fast and Slow Movements Using Weighted Gyroscope and Acceleration Constraints," in *Proceedings of the 22nd International Conference on Information Fusion*, 2019.

[21] J. K. Lee and T. H. Jeon, "Magnetic Condition-Independent 3D Joint Angle Estimation Using Inertial Sensors and Kinematic Constraints Jung," *Sensors*, vol. 19, no. 24, 2019.

[22] A. J. Van Den Bogert, D. Blana, and D. Heinrich, "Implicit methods for efficient musculoskeletal simulation and optimal control," *Procedia IUTAM*, vol. 2, pp. 297–316, 2011.

[23] A. Seth, J. L. Hicks, T. K. Uchida, A. Habib, C. L. Dembia, J. J. Dunne, C. F. Ong, M. S. DeMers, A. Rajagopal, M. Millard, S. R. Hamner, E. M. Arnold, J. R. Yong, S. K.

Lakshmikanth, M. A. Sherman, J. P. Ku, and S. L. Delp, "OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement," *PLoS Computational Biology*, vol. 14, no. 7, pp. 1–20, 2018.

[24] A. Seth, M. Dong, R. Matias, and S. Delp, "Muscle Contributions to Upper-Extremity Movement and Work From a Musculoskeletal Model of the Human Shoulder," *Frontiers in Neurorobotics*, vol. 13, no. November, pp. 1–9, 2019.

[25] "OpenSense - Kinematics with IMU Data - OpenSim Documentation." https://simtk-confluence.stanford.edu/display/OpenSim/OpenSense+-+Kinematics+with+IMU+Data. Accessed: 01-09-2020.

[26] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," *IEEE International Conference on Rehabilitation Robotics*, 2011.

[27] J. Hol, *Sensor fusion and calibration of inertial sensors, vision, ultrawideband and GPS," Linköping Studies in Science and Technology.* PhD thesis, Linköping University, 2011.

[28] U. Qureshi and F. Golnaraghi, "An Algorithm for the In-Field Calibration of a MEMS IMU," *IEEE Sensors Journal*, vol. 17, no. 22, pp. 7479–7486, 2017.

[29] A. B. Chatfield, *Fundamentals Of High Accuracy Inertial Navigation.* American Institute of Aeronautics and Astronautics, 1997.

[30] O. J. Woodman, "An introduction to inertial navigation," Tech. Rep. 696, University of Cambridge, 2007.

[31] M. Kok, J. D. Hol, T. B. Schön, F. Gustafsson, and H. Luinge, "Calibration of a magnetometer in combination with inertial sensors," *Proceedings of the 15th International Conference on Information Fusion, FUSION 2012*, pp. 787–793, 2012.

[32] M. Kok and T. B. Schon, "Magnetometer Calibration Using Inertial Sensors," *IEEE Sensors Journal*, vol. 16, no. 14, pp. 5679–5689, 2016.

[33] F. Olsson, M. Kok, K. Halvorsen, and T. B. Schon, "Accelerometer calibration using sensor fusion with a gyroscope," *Proceedings of the IEEE Workshop on Statistical Signal Processing*, vol. 2016, pp. 660–664, 2016.

[34] G. Panahandeh, I. Skog, and M. Jansson, "Calibration of the Accelerometer Triad of an Inertial Measurement Unit, Maximum Likelihood Estimation and Cramér-Rao bound," *2010 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2010 - Conference Proceedings*, no. September, pp. 15–17, 2010.

[35] I. Skog and P. Handel, "Calibration of A MEMS Inertial Measurement Unit," *18th IMEKO World Congress 2006: Metrology for a Sustainable Development*, vol. 2, no. January 2006, pp. 1445–1450, 2006.

[36] W. Gander, G. H. Golub, and R. Strebel, "Least-Squares Fitting of Circles and Ellipses," *Bit*, vol. 34, no. 4, pp. 558–578, 1994.

[37] M. Grant and S. Boyd, *Recent Advances in Learning and Control: Graph implementations for nonsmooth convex programs. Lecture Notes in Control and Information Sciences*. Springer-Verlag Limited, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[38] M. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming version 2.1." http://cvxr.com/cvx, 2014.

[39] M. Sherman, "Simbody Theory Manual," Tech. Rep. March, Stanford University, 2013.

[40] A. Jain and G. Rodriguez, "Recursive Dynamics Algorithm for Multibody Systems with Prescribed Motion," *Journal of Guidance, Control and Dynamics*, vol. 16, no. 5, pp. 830–837, 1993.

[41] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: Multibody dynamics for biomedical research," *Procedia IUTAM*, vol. 2, pp. 241–261, 2011.

[42] G. Rodriguez, A. Jain, and K. Kreutz-Delgado, "Spatial Operator Algebra for Multibody System Dynamics," *Journal of the Astronautical Sciences*, vol. 40, no. 1, pp. 27–50, 1992.

[43] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach*. Cambridge University Press, 2007.

[44] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. New Jersey: Prentice Hall, Englewood, third ed., 2010.

[45] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2020.

[46] J. L. Torres-Moreno, J. L. Blanco-Claraco, A. Giménez-Fernándeze, E. Sanjurjo, and M. Á. Naya, "Online Kinematic and Dynamic-State Estimation for Constrained Multibody Systems Based on IMUs," *Sensors (Switzerland)*, vol. 16, no. 3, 2016.

[47] K. Chatzilygeroudis, B. Fichera, W. Amanhoud, and Y. Mollard, "iiwa_ros." https://github.com/epfl-lasa/iiwa_ros, 2019. Accessed: 05-10-2020.

[48] Xsens, "MTw Awinda User Manual MTw," Tech. Rep. May, Xsens Technologies B.V., 2018.

[49] J. L. McGinley, R. Baker, R. Wolfe, and M. E. Morris, "The reliability of three-dimensional kinematic gait measurements: A systematic review," *Gait and Posture*, vol. 29, no. 3, pp. 360–369, 2009.

[50] M. A. Skoglund, G. Hendeby, and D. Axehill, "Extended Kalman Filter Modifications Based on an Optimization View Point," in *Proceedings of the 18th International Conference of Information Fusion*, pp. 1856–1861, 2015.

[51] Xsens, "Magnetic Calibration Manual," tech. rep., Xsens Technologies B.V., 2019.

[52] R. M. Murray, Z. Li, and S. Shankar Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 2017.

[53] A. Jouybari, H. Amiri, A. A. Ardalan, and N. K. Zahraee, "Methods comparison for attitude determination of a lightweight buoy by raw data of IMU," *Measurement: Journal of the International Measurement Confederation*, vol. 135, pp. 348–354, 2019.

[54] A. Seth, R. Matias, A. P. Veloso, and S. L. Delp, "A Biomechanical Model of the Scapulothoracic Joint to Accurately Capture Scapular Kinematics during Shoulder Movements," *PLoS ONE*, vol. 11, no. 1, pp. 1–18, 2016.

[55] KUKA, "KUKA iiwa Lightweight Robot Specification," tech. rep., KUKA Roboter GmbH, Augsburg, 2015.