

École polytechnique de Louvain

Sensor fusion for three-dimensional movement of human beings on an Internet of Things network

Author: **Hadrien SONNET**
Supervisor: **Peter VAN ROY**
Readers: **Ramin SADRE, Peer STRITZINGER**
Academic year 2023–2024
Master [120] in Data Sciences Engineering

Abstract

The purpose of this master's thesis is to apply the Kalman filter to the problem of human motion tracking. The goal is to develop a fault-tolerant sensor fusion application that can accurately track human movement.

One of the primary challenges of this thesis will be creating a physical model of the human body to improve motion tracking accuracy. By integrating this model, we aim to significantly enhance the precision of our tracking system.

Acknowledgement

I would like to start by expressing my heartfelt gratitude to my family and friends for their unwavering support and invaluable advice throughout the year.

I am also deeply appreciative of the GRiSP team for their prompt assistance whenever I encountered bugs while using NIFs with the GRiSP board, and for their help in resolving installation issues.

Lastly, I would like to extend my sincere thanks to Professor Van Roy for his continuous support and availability over the past year. His guidance on the various steps I needed to take has been instrumental in my progress.

Contents

1	Introduction	1
2	Materials	3
2.1	GriSP	3
2.2	Pmod Nav	4
3	Kalman Filter	5
3.1	Model definition	5
3.2	Kalman filter algorithm	6
3.2.1	Prediction	6
3.2.2	Update	6
3.3	Properties of the Kalman filter algorithm	7
3.4	Extended Kalman filter	8
3.4.1	Prediction	8
3.4.2	Update	8
3.4.3	Main differences with the Kalman filter	9
4	Presentation of the model	10
4.1	Kalman filter for attitude	10
4.1.1	Quaternions	10
4.1.2	Initialization of the quaternion	12
4.1.3	Quaternion prediction	13
4.1.4	Measurement of the quaternion	15
4.1.5	Definition of the Kalman	18
4.1.6	Conclusion of the Kalman for attitude	19
4.2	Kalman filter for position	19
4.2.1	Introduction	19
4.2.2	Prediction step	20
4.3	Constraints on the model	20
4.3.1	Sensor placement	21
4.3.2	Constraint between upper chest and abdomen	21

4.3.3	Constraint between upper chest and arm	23
4.3.4	Constraints between arm and forearm	23
4.3.5	Consequences of the constraints on the model	24
4.4	Definition of the model	24
4.4.1	Definition of the state variables	24
4.4.2	Process of the extended Kalman filter	25
4.4.3	Definition of the Jacobian matrix	27
4.4.4	Measurement process	29
4.5	Conclusion on the presentation of the model	30
5	Experiment and implementation	31
5.1	Hera Framework	31
5.1.1	Description of the Hera framework	31
5.1.2	Detail of implementation	32
5.2	Naive implementation	33
5.2.1	The naive implementation	33
5.2.2	First model	34
5.2.3	Second model	35
5.2.4	Third model	38
5.3	Implementation taking advantage of the fast computing time	39
5.4	Conclusion on the model	45
6	Parameter settings	46
6.1	Variance for each variable	46
6.1.1	Angular velocity	46
6.1.2	Quaternions	46
6.1.3	Acceleration of the Upper Chest	47
6.1.4	Position and speed variances	47
7	What are the next steps ?	48
7.1	Decreasing the time interval between two steps	48
7.1.1	Why reducing the time interval	48
7.1.2	Using fewer GRiSP cards	49
7.1.3	Changing the solver	50
7.2	Optimizing the hyperparameters	51
7.2.1	Attitude tracking	51
7.2.2	Position tracking	51
8	Conclusion	52
	Bibliography	53

9 Bibliography	54
A Jacobian matrix of the prediction function	56

Chapter 1

Introduction

The current master's thesis builds upon two previous studies concerning sensor fusion technologies and their application within the context of IoT networks. The first master thesis is "The Hera framework for fault-tolerant sensor fusion on an Internet of Things network with application to inertial navigation and tracking" and the second is "Low-cost high-speed sensor fusion with GRiSP and Hera".[13][8]

The point of this master thesis is to leverage the work done previously by applying it to the sensor fusion for three-dimensional movement of human beings on an Internet of things network.

All the work done in these master is possible thanks to the GRiSP card. GRiSP is an hardware and software solution tailored for Erlang and Elixir developer. Traditionally, embedded solution use low-level language where properties like fault tolerance, debugging and tracing, test and iterate are hard to implement. That's where Erlang comes handy to create this kind of solution, with its fault-tolerance and its supervision structure.

The first master thesis created the hera framework. Hera, as described in the master thesis is "A Kalman filter-based sensor fusion framework for Erlang which offers a high-level approach for dynamic, soft real-time, and fault-tolerant sensor fusion directly at the edge of an IoT network". [13]

They built an attitude and heading reference system (AHRS) as well as a 6 degrees of freedom (DOF) inertial measurement unit (IMU). The main conclusion given by this master thesis is that AHRS experiment gave good result while the IMU experiment still needed improvement.

The second master thesis tried to improve the performance of the first master thesis.

Two major improvement were done. First, the Hera module has been upgraded to work with the GRiSP2 board. This significantly improved the performance.

Second, is the use of the Native Implemented Functions (NIFs) to do matrix computation. NIFs is simple and efficient way of calling C-code in Erlang. To improve the calculation time for a Kalman filter the numerl library was imported to do all the matrix computation.[1]

These two improvement gave a real step up in what is possible using the GRiSP card. Building on this foundation, the current thesis aims to improve the resolution of the AHRS problem as it will be described in section 4.1. It will also improve the IMU by leveraging a model of a human body. The model use constraint to represent a human body, and thus gives information on how the sensor should be positioned relative to one an other. This will be further discussed in section 4.3.

Chapter 2

Materials

This chapter aims at referencing all the material used for the following experiences. We will also explain some of the specification of these equipments that we will have to deal with.

2.1 GRiSP

The GRiSP platform is a development environment designed to simplify the creation of embedded systems with a particular emphasis on real-time capabilities. It integrates both hardware and software components to streamline the development process, making it particularly suited for applications where precise timing and performance are critical.

GRiSP hardware includes a development board equipped with essential peripherals to support a wide range of embedded applications. This typically includes a processor capable of real-time embedded “operating system” (RTEMS), memory (SRAM and flash storage) for code and data, and input/output interfaces such as GPIO, SPI, I2C, UART, and more specialized interfaces like Ethernet.[7]

The GRiSP software stack is built around the Erlang programming language, renowned for its concurrency, fault tolerance, and distributed computing capabilities.[2] The key software components include RTEMS, a minimal Erlang runtime environment tailored for embedded systems, and a collection of libraries and tools specifically designed to leverage Erlang’s strengths in real-time and embedded contexts.

One of the main benefits of GRiSP is the ease of development it offers by abstracting much of the complexity involved in real-time and embedded systems

programming. This makes the platform accessible even to those who may not be experts in low-level system design. Additionally, GRiSP's real-time capabilities are enhanced by Erlang's lightweight process model and robust concurrency support, enabling developers to create systems that can handle multiple tasks simultaneously without compromising performance. Erlang's design principles, including process isolation and automatic error recovery, contribute to the reliability and fault tolerance of GRiSP-based systems, ensuring they are resilient to failures.[2]

2.2 Pmod Nav

The Pmod Nav module features a suite of sensors, including an accelerometer, gyroscope, magnetometer, and barometer. The accelerometer measures linear acceleration in three axes, providing data on movement and orientation changes. The gyroscope offers angular velocity measurements, essential for tracking rotational movements. The magnetometer detects magnetic field strength, which can be used for compass heading and orientation relative to the Earth's magnetic field. Additionally, the barometer measures atmospheric pressure, enabling altitude estimation and environmental pressure monitoring. [3]

One of the key properties of an accelerometer, which will be utilized later in this thesis, is its ability to determine the direction of gravity when it is stationary. This capability is crucial for accurately establishing the orientation of the sensor.

Chapter 3

Kalman Filter

This chapter is largely inspired by the book Introduction and Implementations of the Kalman Filter by Felix Govaers. The same notation will be used in this work. [6]

The focus of this work is on the Kalman filter, and this chapter aims to elucidate its operational principles. The Kalman filter is employed to estimate the state of a system described by linear dynamical systems in a state-space format. The Kalman filter operates in two main steps, which will be elaborated upon.

3.1 Model definition

The process model describes the state evolution from time $k - 1$ to k as follows:

$$\mathbf{s}_k = \mathbf{F} * \mathbf{s}_{k-1} + \mathbf{B} * \mathbf{u}_{k-1} + \mathbf{w}_{k-1} \quad (3.1)$$

- $\mathbf{s}_k$ is the state vector at time k .
- $\mathbf{u}_{k-1}$ is the control vector at time $k-1$.
- $\mathbf{w}_{k-1}$ is the process noise, $\mathbf{w}_{k-1} \sim \mathcal{N}(0, Q)$.
- Q is the variance of the noise
- $\mathbf{F}$ is the state transition matrix
- $\mathbf{B}$ is the control-input matrix

This formulation is known as the process model. It is important to note that the matrices $\mathbf{F}$ and $\mathbf{B}$ are defined for the transition from $\mathbf{s}_{k-1}$ to $\mathbf{s}_k$ and may not be the same for the transition from $\mathbf{s}_k$ to $\mathbf{s}_{k+1}$.

The measurement model at the current step is defined as:

$$\mathbf{m}_k = \mathbf{H}\mathbf{s}_k + \mathbf{v}_k \quad (3.2)$$

where:

- $\mathbf{m}_k$ is the measurement vector
- $\mathbf{v}_k$ is the measurement noise, $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R})$
- $\mathbf{R}$ is the variance of the measurement noise
- $\mathbf{H}$ is the measurement matrix

The Kalman filter assumes the initial estimate $\mathbf{s}_0$ and the series of measurements $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_k$, as well as the matrices $\mathbf{F}$, $\mathbf{B}$, $\mathbf{H}$, $\mathbf{Q}$, and $\mathbf{R}$, are known. The following sections will explain how these parameters were chosen to fit the model.

3.2 Kalman filter algorithm

The Kalman filter algorithm is divided into two main stages: prediction and update.

3.2.1 Prediction

The prediction step is defined by the following two equations:

$$\hat{\mathbf{s}}_k^- = \mathbf{F}\hat{\mathbf{s}}_{k-1}^+ + \mathbf{B}\mathbf{u}_{k-1} \quad (\text{Predicted state estimate})$$

$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}^+\mathbf{F}^T + \mathbf{Q} \quad (\text{Predicted error covariance})$$

3.2.2 Update

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}\hat{\mathbf{s}}_k^- \quad (\text{Measurement residual})$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{R} + \mathbf{H}\mathbf{P}_k^- \mathbf{H}^T)^{-1} \quad (\text{Kalman gain})$$

$$\hat{\mathbf{s}}_k^+ = \hat{\mathbf{s}}_k^- + \mathbf{K}_k \tilde{\mathbf{y}}_k \quad (\text{Updated state estimate})$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_k^- \quad (\text{Updated error covariance})$$

3.3 Properties of the Kalman filter algorithm

In Bayesian statistics, the Kalman filter can be viewed as a maximum a posteriori estimator. This implies the following. Let us define the error:

$$\mathbf{e}_k^- = \mathbf{s}_k - \hat{\mathbf{s}}_k^- \quad (3.3)$$

$$\Sigma_k^- = \mathbb{E}(\mathbf{e}_t^- (\mathbf{e}_t^-)^T) \quad (3.4)$$

$$\mathbf{e}_t^+ = \mathbf{s}_t - \hat{\mathbf{s}}_t^+ \quad (3.5)$$

$$\Sigma_k^+ = \mathbb{E}(\mathbf{e}_t^+ (\mathbf{e}_t^+)^T) \quad (\text{Posterior covariance})$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{R} + \mathbf{H} \mathbf{P}_k^- \mathbf{H}^T)^{-1} \quad (3.6)$$

The Kalman gain equation (Kalman gain) is designed to minimize the posterior covariance. We note that as:

$$\mathbf{R} \rightarrow 0 \text{ then } \mathbf{K}_k = \mathbf{H}^{-1} \text{ and } \hat{\mathbf{s}}_k^+ = \hat{\mathbf{s}}_k^- + \mathbf{H}^{-1}(\mathbf{m}_k - \mathbf{H}\hat{\mathbf{s}}_k^-) \quad (3.7)$$

$$\hat{\mathbf{s}}_k^+ = \mathbf{H}^{-1} \mathbf{m}_k \quad (3.8)$$

This shows that as the measurement noise covariance $\mathbf{R}$ approaches zero, the updated state estimate $\hat{\mathbf{s}}_k^+$ relies solely on the measurement $\mathbf{m}_k$, disregarding the predicted state estimate.

This is valid only if the matrix $\mathbf{H}$ is invertible, implying $\mathbf{H}$ is full rank and square. Consequently, from $\mathbf{m}_k$, we can deduce every component of $\mathbf{s}_k$. However, $\mathbf{H}$ is not always invertible.

Similarly, for $\mathbf{P}_k^- \rightarrow 0$:

$$\mathbf{P}_k^- \rightarrow 0 \implies \mathbf{K}_t = 0 \text{ and } \hat{\mathbf{s}}_k^+ = \hat{\mathbf{s}}_k^- \quad (3.9)$$

Thus, as $\mathbf{P}_k^- \rightarrow 0$ approaches zero, the posterior state is equal to the predicted state estimate.

This indicates that $\mathbf{Q}$ and $\mathbf{R}$ can be perceived as measures of the model's reliance on the process and measurement equations (3.1) and (3.2), respectively. A smaller component wise norm of $\mathbf{Q}$ implies greater reliance on the process model, since it has a direct impact on $\mathbf{P}_k^-$. While a smaller component wise norm of $\mathbf{R}$ implies greater reliance on the measurement model.

3.4 Extended Kalman filter

In this chapter, we have discussed the standard Kalman filter, which is effective for linear systems. However, many real-world systems, including the one we are addressing, are nonlinear.

The extended Kalman filter (EKF) extends the Kalman filter to nonlinear systems by linearizing the model around the current estimate. Suppose we have the following nonlinear models for state transition and measurement [6]:

$$\hat{\mathbf{s}}_k^- = f(\mathbf{s}_{k-1}, \mathbf{u}_{k-1}) + \omega_{k-1} \quad (3.10)$$

$$\mathbf{m}_k = h(\mathbf{s}_k) + \mathbf{v}_k \quad (3.11)$$

where f is a nonlinear function of the previous state $\mathbf{s}_{k-1}$, and the control input, $\mathbf{u}_{k-1}$, providing the current state $\mathbf{s}_k$. h is the measurement function that relates the current state, $\mathbf{s}_k$, to the measurement $\mathbf{m}_k$. ω_{k-1} and $\mathbf{v}_k$ are Gaussian noises for the process model and the measurement model with covariance $\mathbf{Q}$ and $\mathbf{R}$, respectively.

The Jacobians of the functions f and h at a point are defined as:

$$\mathbf{F}_{k-1} = \frac{\partial f}{\partial x} \big|_{\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1}} \quad (3.12)$$

$$\mathbf{H}_k = \frac{\partial h}{\partial x} \big|_{\hat{\mathbf{x}}^+} \quad (3.13)$$

By injecting these equality in the stages defined in section 3.2. We obtain the following prediction and measurement steps.

3.4.1 Prediction

$$\hat{\mathbf{s}}_k^- = f(\hat{\mathbf{s}}_{k-1}^+, \mathbf{u}_{k-1}) \quad (3.14)$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1} \mathbf{P}_{k-1}^+ \mathbf{F}_{k-1}^T + \mathbf{Q} \quad (\text{Predicted error covariance ekf})$$

3.4.2 Update

$$\tilde{\mathbf{y}}_k = \mathbf{m}_k - h(\hat{\mathbf{s}}_k^-) \quad (\text{Measurement residual ekf})$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{R} + \mathbf{H} \mathbf{P}_k^- \mathbf{H}^T)^{-1} \quad (\text{Kalman gain ekf})$$

$$\hat{\mathbf{s}}_k^+ = \hat{\mathbf{s}}_k^- + \mathbf{K}_k \tilde{\mathbf{y}}_k \quad (\text{Updated state estimate ekf})$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_k^- \quad (\text{Updated error covariance ekf})$$

3.4.3 Main differences with the Kalman filter

The primary difference between the extended Kalman filter (EKF) and the standard Kalman filter lies in the computation of $\hat{\mathbf{s}}_k^+$. The EKF uses nonlinear functions, while the standard Kalman filter relies on matrix multiplication. Additionally, the measurement function in the EKF can also be nonlinear.

Another significant difference is the need to compute the Jacobians of the nonlinear functions in the EKF. This is necessary for both the process and measurement functions if they are nonlinear. The computational complexity can be high depending on the function and the number of state variables.

Chapter 4

Presentation of the model

4.1 Kalman filter for attitude

One of the main steps in solving the human motion tracking problem is to create an Attitude and Heading Reference System (AHRS). This reference system will enable the transformation from the sensor frame to the global frame, allowing us to reinterpret the acceleration and rotational acceleration data provided by the sensors in a global context. Consequently, this transformation is essential for navigating within a global space.

The Kalman filter for attitude will be utilized to represent the data and state of each sensor in the {North, Up, East} frame. This will provide a coherent understanding of the relative positions and orientations of the different sensors with respect to one another.

This step is also mandatory to be able to isolate the acceleration due to the gravity. In fact, the accelerometer always measures the accelerations of the gravity even when sitting still. It is thus mandatory to know what is the direction of the gravity to remove it when trying to interpret how the sensors is moving.

4.1.1 Quaternions

In this chapter, quaternions will play a crucial role. This section provides a brief introduction to quaternions and establishes the notation used throughout the chapter.

Quaternions offer a way to represent rotations in three-dimensional space. They are based on the principle that any 3D rotation can be described as a rotation around a specific axis by a certain angle. Therefore, it is essential to define this axis and the angle of rotation.

A quaternion is defined as follows:

$$\mathbf{q} = w + xi + yj + zk \quad (4.1)$$

To understand quaternion operations, we need to define the relationships between the axis i,j,k:

$$\begin{aligned} i^2 &= -1, & ij &= k, & ji &= -k, \\ j^2 &= -1, & jk &= i, & kj &= -i, \\ k^2 &= -1, & ki &= j, & ik &= -j, \end{aligned} \quad (4.2)$$

In the quaternion space, we can define a rotation as:

$$\mathbf{F}(\mathbf{p}) = \mathbf{q}\mathbf{p}\mathbf{q}^{-1} \quad (4.3)$$

- $\mathbf{q}$ is a unit quaternion $\leftrightarrow ||\mathbf{q}|| = 1$
- $\mathbf{F}(\mathbf{p}) = \begin{bmatrix} w & x & y & z \end{bmatrix}$, x, y, z are the coordinates of $\mathbf{p}$ after the rotation.

As mentioned earlier, a quaternion is based on the idea that a rotation can be specified by a direction and an angle. This direction and angle can be derived from a quaternion. It can be shown that the following equation holds

$$\mathbf{q} = w + xi + yj + zk = \cos(\alpha/2) + \mathbf{u}\sin(\alpha/2) \quad (4.4)$$

- $\mathbf{u}$ is a unit vector

Here, $\mathbf{u}$ is a unit vector indicating the direction around which we rotate by an angle α .

This representation is particularly useful for the physical interpretations that will be explored in this chapter.

Some of the most important properties of rotation quaternions are :

- rotation quaternion are unit quaternion
- the inverse of a unit quaternion is defined by

$$\begin{aligned} \mathbf{q} &= w + xi + yj + zk \\ \mathbf{q}^{-1} &= w - xi - yj - zk \end{aligned} \quad (4.5)$$

Since a quaternion defines a rotation, we can from a rotation matrix define an equivalent quaternion. Let's define a rotation matrix:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (4.6)$$

We can define a quaternion corresponding to $\mathbf{R}$ as: [10]

$$\mathbf{q} = \frac{1}{2} \begin{pmatrix} (1 + r_{11} + r_{22} + r_{33})^{\frac{1}{2}} \\ \frac{r_{32} - r_{23}}{(1 + r_{11} + r_{22} + r_{33})^{\frac{1}{2}}} \\ \frac{r_{13} - r_{31}}{(1 + r_{11} + r_{22} + r_{33})^{\frac{1}{2}}} \\ \frac{r_{21} - r_{12}}{(1 + r_{11} + r_{22} + r_{33})^{\frac{1}{2}}} \end{pmatrix} \quad (4.7)$$

Inversely, we can, from a quaternion define a rotation matrix:

$$\begin{aligned} \mathbf{F}(\mathbf{p}) &= \mathbf{q}\mathbf{p}\mathbf{q}^{-1} \\ &= \begin{pmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2(q_0^2 + q_3^2) - 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \end{aligned} \quad (4.8)$$

4.1.2 Initialization of the quaternion

To ensure accurate quaternion updates at each step, our initial task is to calibrate the starting quaternion. This calibration process will be outlined in the following section. The point of the quaternion that we will define is to go from local frame data, i.e. the sensor frame, to global frame. The global frame chosen is the {North, Up, East}.

During the calibration process, the sensor will be placed still. The accelerometer will then give an acceleration of $-\hat{\mathbf{g}}$. The hat notation indicates that this is the data read from the accelerometer, acknowledging that there is some noise and it is an approximation of the gravity vector $\mathbf{g}$.

$$-\hat{\mathbf{g}} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T \quad (4.9)$$

$$\mathbf{mag} = \frac{\begin{bmatrix} m_x & m_y & m_z \end{bmatrix}^T}{\| \begin{bmatrix} m_x & m_y & m_z \end{bmatrix}^T \|} \quad (4.10)$$

$$\mathbf{up} = \frac{-\hat{\mathbf{g}}}{\|\hat{\mathbf{g}}\|} \quad (4.11)$$

$$\mathbf{down} = -\mathbf{up} = \frac{\hat{\mathbf{g}}}{\|\hat{\mathbf{g}}\|} \quad (4.12)$$

$$\mathbf{east} = \mathbf{down} \times \mathbf{mag} \quad (4.13)$$

$$\mathbf{north} = \mathbf{east} \times \mathbf{down} \quad (4.14)$$

$-\hat{\mathbf{g}}$ is defined by the acceleration given by the accelerometer while the sensor is standing still. This allows us to define $\mathbf{up}$ which is the vector $\hat{\mathbf{g}}$ normalized.

$\mathbf{mag}$ is defined as the magnetometer data while the sensor stands still and normalized to be a unit vector.

The $\mathbf{mag}$ is not directly used for the north direction because the magnetometer points to the magnetic north and not the geographic north.[4]

The vector $\mathbf{north}, \mathbf{up}, \mathbf{east}$ define a rotation matrix that allows us to go from the local to the global frame:

$$\mathbf{R} = [\mathbf{north}^T \quad \mathbf{up}^T \quad \mathbf{east}^T] \quad (4.15)$$

From this rotation matrix we can deduce a quaternion that gives an equivalent rotation using equation 4.7. We will call this quaternion q_0^{-1} , it gives the rotation from the local to the global frame. In the following section we will work with quaternion going from global to local frame. Thus we will take the inverse of this quaternion, q_0 , as the initial state.

4.1.3 Quaternion prediction

To determine the sensor's rotation at a specific time, we continuously update the quaternion using gyroscope's rotational velocity data.

In this section we will elaborate the equation that gives us the possibility to do so. Let's first start with the derivative of a quaternion: [11]

$$\boldsymbol{\omega}(t) \triangleq \frac{d\boldsymbol{\phi}_{\mathcal{L}}(t)}{dt} \triangleq \lim_{\Delta t \rightarrow 0} \frac{\Delta \boldsymbol{\phi}_{\mathcal{L}}}{\Delta t} \quad (4.16)$$

If the perturbation angle $\Delta \Phi_{\mathcal{L}}$ is small then the perturbation can be approximated by the Taylor expansions of equation 4.4 to the linear term,

$$\Delta \mathbf{q}_{\mathcal{L}} \approx \begin{bmatrix} 1 \\ \Delta \Phi_{\mathcal{L}}/2 \end{bmatrix} \quad (4.17)$$

$$\begin{aligned}
\dot{\mathbf{q}} &\triangleq \lim_{\Delta t \rightarrow 0} \frac{\mathbf{q}(t + \Delta t) - \mathbf{q}(t)}{\Delta t} \\
&= \lim_{\Delta t \rightarrow 0} \frac{\mathbf{q} \otimes \Delta \mathbf{q}_{\mathcal{L}} - \mathbf{q}}{\Delta t} \\
&= \lim_{\Delta t \rightarrow 0} \frac{\mathbf{q} \otimes \left(\begin{bmatrix} 1 \\ \Delta \Phi_{\mathcal{L}}/2 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)}{\Delta t} \\
&= \frac{1}{2} \mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{\mathcal{L}} \end{bmatrix}
\end{aligned} \tag{4.18}$$

It is important to note that in this development the angular rate $\boldsymbol{\omega}_{\mathcal{L}}$ is expressed in the local frame defined by $\mathbf{q}$. Thus implying that the quaternion $\mathbf{q}$ defines a rotation from global to local frame. We can develop the Taylor series:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \dot{\mathbf{q}}_n \Delta t + \frac{1}{2!} \ddot{\mathbf{q}}_n \Delta t^2 + \frac{1}{3!} \dddot{\mathbf{q}}_n \Delta t^3 + \dots \tag{4.19}$$

Using equation 4.18 with $\ddot{\boldsymbol{\omega}} = 0$,

$$\begin{aligned}
\dot{\mathbf{q}}_n &= \frac{1}{2} \mathbf{q}_n \boldsymbol{\omega}_n \\
\ddot{\mathbf{q}}_n &= \frac{1}{2^2} \mathbf{q}_n \boldsymbol{\omega}_n^2 + \frac{1}{2} \mathbf{q}_n \dot{\boldsymbol{\omega}} \\
\ddot{\mathbf{q}}_n &= \frac{1}{2^3} \mathbf{q}_n \boldsymbol{\omega}_n^3 + \frac{1}{4} \mathbf{q}_n \dot{\boldsymbol{\omega}} \boldsymbol{\omega}_n + \frac{1}{2} \mathbf{q}_n \boldsymbol{\omega}_n \dot{\boldsymbol{\omega}} \\
\mathbf{q}_n^{(i \geq 4)} &= \frac{1}{2^i} \mathbf{q}_n \boldsymbol{\omega}_n^i + \dots
\end{aligned} \tag{4.20}$$

If we take as constant the angular rate over the period $[t_n, t_{n+1}]$, we have $\dot{\boldsymbol{\omega}} = 0$. Then :

$$\mathbf{q}_{n+1} = \mathbf{q}_n \otimes \left(1 + \frac{1}{2} \boldsymbol{\omega}_n \Delta t + \frac{1}{2!} \left(\frac{1}{2} \boldsymbol{\omega}_n \Delta t \right)^2 + \frac{1}{3!} \left(\frac{1}{2} \boldsymbol{\omega}_n \Delta t \right)^3 + \dots \right) \tag{4.21}$$

We can identify the Taylor series of the exponential, $\exp(\boldsymbol{\omega}_n \Delta t / 2)$. It can be shown from the quaternion properties that such a quaternion represents the incremental rotation of $\boldsymbol{\omega}_n \Delta t$ around the axis defined by $\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}$.

$$\exp(\boldsymbol{\omega} \Delta t / 2) = \begin{bmatrix} \cos(\|\boldsymbol{\omega}\| \Delta t / 2) \\ \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|} \sin(\|\boldsymbol{\omega}\| \Delta t / 2) \end{bmatrix} \tag{4.22}$$

We assume really small Δt and we will only take into account the first two element of the Taylor development:

$$\mathbf{q}_{n+1} = \mathbf{q}_n \otimes \left(1 + \frac{1}{2} \boldsymbol{\omega}_n \Delta t \right) \tag{4.23}$$

We can define the matrix:

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & -\omega_z & \omega_y \\ \omega_y & \omega_z & 0 & -\omega_x \\ \omega_z & -\omega_y & \omega_x & 0 \end{bmatrix} \quad (4.24)$$

This matrix is equivalent to doing a quaternion product from the right for $\boldsymbol{\omega} = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix}$. Thus giving us the equivalent of the equation 4.23 under the matrix form:

$$\begin{aligned} \mathbf{q}_{n+1} &= (\mathbb{I} + \frac{1}{2}\Omega(\boldsymbol{\omega})\Delta t)\mathbf{q}_n \\ &= \begin{bmatrix} 1 & \frac{-\omega_x\Delta t}{2} & \frac{-\omega_y\Delta t}{2} & \frac{-\omega_z\Delta t}{2} \\ \frac{\omega_x\Delta t}{2} & 1 & \frac{-\omega_z\Delta t}{2} & \frac{\omega_y\Delta t}{2} \\ \frac{\omega_y\Delta t}{2} & \frac{\omega_z\Delta t}{2} & 1 & \frac{-\omega_x\Delta t}{2} \\ \frac{\omega_z\Delta t}{2} & \frac{-\omega_y\Delta t}{2} & \frac{\omega_x\Delta t}{2} & 1 \end{bmatrix} \mathbf{q}_n \end{aligned} \quad (4.25)$$

This matrix is the state transition matrix in equation 4.4. The matrix is the $\mathbf{F}$ in the equation 3.1, while $\mathbf{q}_n$ is the state. $\mathbf{q}_n$ in our case is the rotation quaternion from global to the local frame. The w_x, w_y, w_z are respectively x, y, z angular rate in rad/s derived from the gyroscope.

4.1.4 Measurement of the quaternion

As depicted in chapter 3, the Kalman algorithm is defined by two equations. The model definition 3.1 and the measurement equation 3.1. We just defined the model we are now defining the measurement equation.

The orientation quaternion for the measurement, will be derived from the accelerometer and magnetometer data. The accelerometer is approximated to represent the direction of gravity. However, we acknowledge that the accelerometer data will not solely reflect gravitational direction and will be influenced by other accelerations. To address this, we introduce a parameter to adjust the weight given to the accelerometer data. This approach is elaborated in [5], which we will briefly discuss in this chapter. The algorithm presented in this paper operates in two steps.

Correct the estimated direction of gravity using accelerometer readings

The first step is described in Figure 4.1. In this section, the vector $\mathbf{v}_g$ stands for the estimated direction of the gravity using the accelerometer data in the local frame.

$$\begin{aligned}
\mathbf{g}^n &= \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^T \\
\hat{\mathbf{g}}^b &= \mathbf{C}(\mathbf{q})\mathbf{g}^n \\
\hat{\mathbf{v}}_g &= \frac{\hat{\mathbf{g}}^b}{\|\hat{\mathbf{g}}^b\|}
\end{aligned} \tag{4.26}$$

Where $\hat{\mathbf{g}}^b$ is the normalized gravity expressed in the local frame and $\mathbf{g}^n$ is the gravity expressed in the global frame. As a recall, the global frame is defined as {North,Up,East} frame. $\mathbf{C}(\mathbf{q})$ is the rotation matrix equivalent to the rotation defined by the quaternion $\mathbf{q}$. $\mathbf{q}$ denotes the rotation quaternion from the global frame to the local frame.

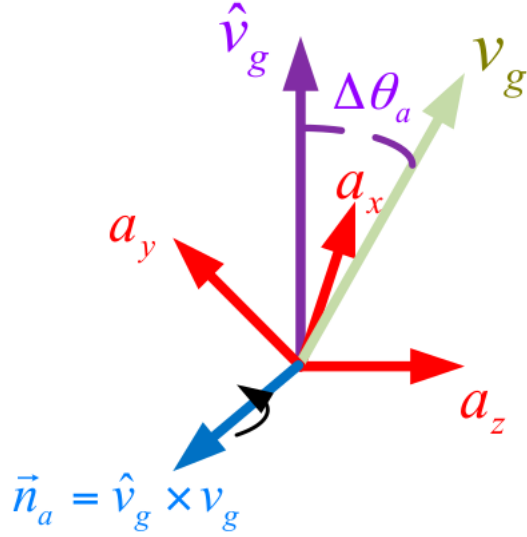


Figure 4.1: Correction of the estimated gravity vector

As shown in Figure 4.1, we expect a difference between the direction of the gravity estimated by the quaternion and the one estimated by using the accelerometer data. The point of this section is to adjust our quaternion to take into consideration the accelerometer data.

In the Figure 4.1, $\mathbf{v}_g$ is defined by the data given by the accelerometer. The

point is to find the angle $\Delta\theta_a$. Which is defined as:

$$\begin{aligned}\mathbf{g}^L &= [a_x, a_y, a_z] \\ \mathbf{v}_g &= \frac{\mathbf{g}^L}{\|\mathbf{g}^L\|}\end{aligned}\tag{4.27}$$

$$\Delta\theta_a = \arccos(v_g \cdot \hat{v}_g)\tag{4.28}$$

$$\mathbf{n}_a = \hat{\mathbf{v}}_g \times \mathbf{v}_g\tag{4.29}$$

We can now define the quaternion $\mathbf{q}_{ae}$:

$$\begin{aligned}\mathbf{q}_{ae} &= \sin\left(\frac{\mu_a \Delta\theta_a}{2}\right) + \mathbf{n}_a \cos\left(\frac{\mu_a \Delta\theta_a}{2}\right) \\ \mathbf{q}_a &= \mathbf{q}_{ae} \otimes \mathbf{q}\end{aligned}\tag{4.30}$$

$\mathbf{q}_{ae}$ defines a rotation of an angle $\mu_a \Delta\theta_a$ around the axis $\mathbf{n}_a$. It is important to notice the introduction of μ_a parameter. This parameter will take value in the $[0,1]$ interval. The higher the μ_a the more we take into account the acceleration data. Inversely the smaller it is the less we take into account the acceleration data.

Correct the estimated direction of the magnetic field using magnetometer readings

First let's define $\mathbf{v}_{mxyz}^G$, the magnetometer data in the frame defined by $\mathbf{q}_a^{-1}$, i.e. the rotation from local to global frame:

$$\mathbf{v}_{mxyz}^G = [\omega, b_x, b_y, b_z] = \mathbf{q}_a^{-1} \otimes [0, m_x, m_y, m_z] \otimes \mathbf{q}_a\tag{4.31}$$

$[m_x, m_y, m_z]$ is the magnetometer data. Equation 4.31 uses $\mathbf{q}_a^{-1}$ to rotate from the local frame to the adjusted {North,Up,East} frame. In order to find the North direction, we omit the vertical component of the data, we need a direction orthogonal to the up direction defined by quaternion $\mathbf{q}_a^{-1}$:

$$\mathbf{v}_{mxz}^G = [0, \frac{b_x}{\sqrt{b_x^2 + b_z^2}}, 0, \frac{b_z}{\sqrt{b_x^2 + b_z^2}}]\tag{4.32}$$

The equation 4.32 gives us a unit vector expressed in the global frame with no Up component. We can now use the rotation defined by $\mathbf{q}_a$ to put this vector in the local frame.

$$\mathbf{v}_{mxz} = \mathbf{q}_a \otimes [0, \frac{b_x}{\sqrt{b_x^2 + b_z^2}}, 0, \frac{b_z}{\sqrt{b_x^2 + b_z^2}}] \otimes \mathbf{q}_a^{-1}\tag{4.33}$$

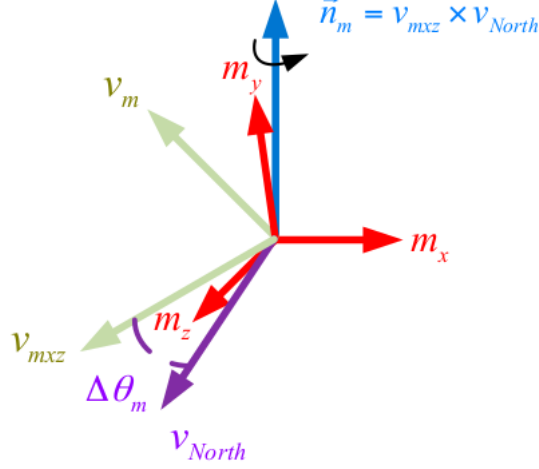


Figure 4.2: Correcting the estimated direction of the magnetic field magnetometer.

$$\mathbf{v}_{North} = \mathbf{q}_a \otimes [0, 1, 0, 0] \otimes \mathbf{q}_a^{-1} \quad (4.34)$$

$$\mathbf{n}_m = v_{mxz} \times v_{North} \quad (4.35)$$

We can define $\Delta\theta_m = \arccos(\mathbf{v}_{mxz} \cdot \mathbf{v}_{north})$ and define:

$$\mathbf{q}_{me} = \sin\left(\frac{\Delta\theta_m}{2}\right) + \mathbf{n}_m \cos\left(\frac{\Delta\theta_m}{2}\right) \quad (4.36)$$

$\mathbf{n}_m$ is the direction around which we need to rotate as we can see on Figure 4.2. Finally we can define the estimation of the quaternion as:

$$\mathbf{q} = \mathbf{q}_{me} \otimes \mathbf{q}_a \quad (4.37)$$

The estimated quaternion will be used as a measurement in the Kalman filter.

4.1.5 Definition of the Kalman

Finally we can define a Kalman filter using the equation 3.1 and 3.2:

$$\mathbf{F} = \begin{bmatrix} 1 & \frac{-\omega_x \Delta t}{2} & \frac{-\omega_y \Delta t}{2} & \frac{-\omega_z \Delta t}{2} \\ \frac{\omega_x \Delta t}{2} & 1 & \frac{-\omega_z \Delta t}{2} & \frac{\omega_y \Delta t}{2} \\ \frac{\omega_y \Delta t}{2} & \frac{\omega_z \Delta t}{2} & 1 & \frac{-\omega_x \Delta t}{2} \\ \frac{\omega_z \Delta t}{2} & \frac{-\omega_y \Delta t}{2} & \frac{\omega_x \Delta t}{2} & 1 \end{bmatrix} \quad (4.38)$$

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

The measurement is defined in the section 4.1.4. While the initial state, $\mathbf{q}_0$ is defined in the section 4.1.2. There is no control vector, so no need to define $\mathbf{B}$ and $\mathbf{u}$.

This geometrical construction allows us to create a measurement for the quaternion. We can observe that by setting $\mu_a = 1$, we replicate the process outlined in section 4.1.2. The key advantage of this method is that it enables us to adjust μ_a based on the specific utilization of the sensor. For instance, if the sensor is deployed in environments where accelerations other than gravity are predominant, we can obtain more accurate results by minimizing the influence of gravity correction. Conversely, in settings where gravity is the primary acceleration, increasing the gravity correction will enhance the measurement accuracy.

4.1.6 Conclusion of the Kalman for attitude

One of the main, advantage of this implementation is its non drifting property. In fact, the gyroscope provides a high resolution but drifting solution, while the magnetometer combined with the accelerometer gives a non drifting but noisy estimation.

This property extends to the whole attitude tracking part of our model. Attitude meaning orientation of each sensor in the global frame.

To achieve accurate tracking, we will implement a Kalman filter for the attitude of each sensor in our model. This allows us to monitor the position of each sensor relative to the others and understand the subject's movements more precisely.

4.2 Kalman filter for position

4.2.1 Introduction

In this section, we introduce the prediction step for estimating position using a Kalman filter. We assume the transformation from the local frame to the global frame is complete, meaning the position, speed, and acceleration vectors are defined in the {North, Up, East} coordinate frame.

4.2.2 Prediction step

First, we define the state of a sensor at a given time with the following state vector:

$$\mathbf{s} = [x, v_x, y, v_y, z, v_z] \quad (4.39)$$

The prediction of the next state follows these equations. Assuming the position $x(t)$ can be predicted as:

$$x(t) = x(t_k) + \int_{t_k}^t v(t) dt \quad (4.40)$$

and assuming constant acceleration over the interval $[t_k, t_{k+1}]$:

$$v(t) = v(t_k) + a * (t - t_k) \quad t_{k+1} \geq t \quad (4.41)$$

Using equations 4.40 and 4.41, and integrating with respect to t , we obtain:

$$x(t) = x(t_k) + v(t_k) * \Delta t + \frac{a * \Delta t^2}{2} \quad (4.42)$$

where:

- $\Delta t = t - t_k$

We will generalize equation 4.42 in 3D to create a Kalman process, i.e. a formulation like equation 3.1.

4.3 Constraints on the model

The primary objective of this study is to enhance prediction accuracy by utilizing a model of the human body. By establishing a model that considers the anatomical positions of various body parts, we aim to improve the precision of our predictions. Specifically, by knowing the exact location of each sensor on the body, we can ascertain their relative positions. This approach aligns with the methodologies presented in the paper "Synthesis of human motion using Kalman filter" and "A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package" adapted to our specific problem. [12] [14]

We will develop a hierarchical model where the position of each sensor is defined relative to a reference sensor in our cases the upper chest sensor. The orientation of each body segments will give information about the positions of others sensors. This section relies on the work done in the previous section, it will leverage the fact that we have an estimation of the orientation of each sensor.

4.3.1 Sensor placement

Four sensors will be placed on the subject.

- one sensor on the upper chest
- one sensor on the abdomen
- one sensor on the right biceps
- one sensor on the right forearm

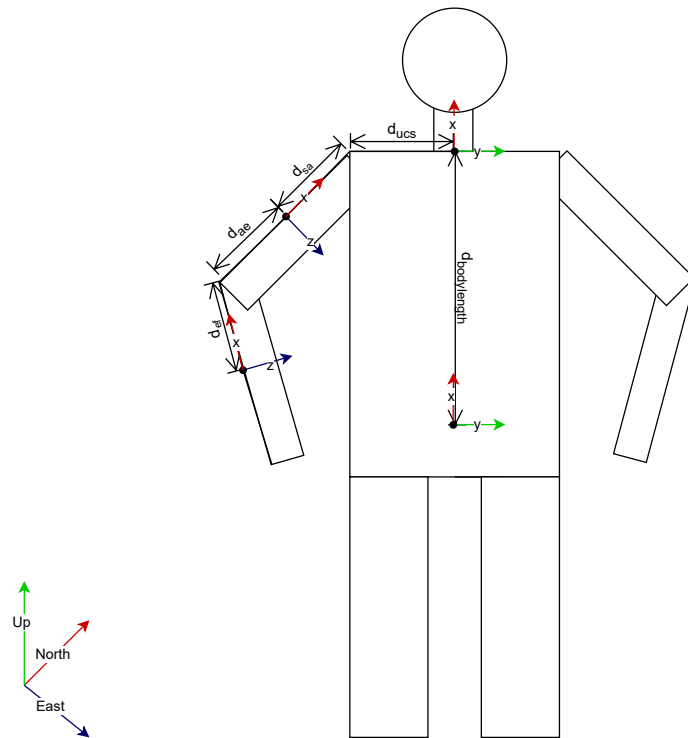


Figure 4.3: Representation of the distances on a human diagram

4.3.2 Constraint between upper chest and abdomen

The goal of this section is to explain how to take advantage of the redundant information given by the upper chest and abdomen sensor. In fact, it is easy to calculate the relative position of each sensor with respect to the other. As we can see on Figure 4.3 the abdomen sensor is in the negative x direction from the upper

chest sensor, while the upper chest is in positive x direction of the abdomen sensor. This distance will be referred as the body length.

$$d_{bodylength} = \text{Distance between abdomen and upper chest sensor} \quad (4.43)$$

Although this distance is not perfectly constant in reality, we assume it to be, for the purposes of the model.

From Figure 4.3, we see that if we know the orientation of each card we can deduce their position, with respect to the others.

The fact that we can deduce the position of one card w.r.t. the other will be used in the process step of the Kalman filter, equation 3.1.

Position of the upper chest

We use a hierarchical model, meaning that the position of each card is deduced from a reference card, here the upper chest card. The position of the upper chest card is deduced using the equations described in the previous section.

One of the upside of this choice is the fact that the attitude tracking part of the problem will have a non drifting solution. The error of the relative position of one sensor with respect to the other will not increase over time. Since, it depends only on the attitude tracking part of the Kalman filter.

On the downside, if the upper chest sensor comes to stop. The remaining card will have a good orientation with respect to each other but won't have a good absolute position in the space.

Position of the abdomen

The idea here, is to use the position of the upper chest sensor and the fact that we know its orientation to deduce the position of the other sensors, using the distance defined above:

$$\mathbf{p}_a = \mathbf{p}_{uc} + \mathbf{C}(\mathbf{q}_{uc}) \begin{bmatrix} -d_{bodylength} & 0 & 0 \end{bmatrix}^T \quad (4.44)$$

$\mathbf{p}_a$ represents the position of the abdomen. $\mathbf{p}_{uc}$ the position of the upper chest sensor. $\mathbf{C}(\mathbf{q}_{uc})$ is the quaternion defined by the upper chest sensor. We use $-d_{bodylength}$ because the abdomen card is in the minus direction.

We take the the position of the abdomen sensor to be $d_{bodylength}$ away from the upper chest sensor in the sensor's x direction. Since the vector is in the sensor

frame we need to put it in the global frame by using the rotation matrix. The rotation matrix we use is the one defined by the upper chest sensor's quaternion

4.3.3 Constraint between upper chest and arm

In this section we want to give the position of the arm relative to the upper chest position. We assume that the arm moves on a sphere centered at the shoulder.

Given this assumptions, the first step will be to find the center's coordinate. By following the same, idea as in the previous subsection:

$$d_{ucs} = \text{Distance between upper chest sensor and shoulder} \quad (4.45)$$

$$\mathbf{p}_s = \mathbf{p}_{uc} + \mathbf{C}(\mathbf{q}_{uc}) \begin{bmatrix} 0 & -d_{ucs} & 0 \end{bmatrix}^T \quad (4.46)$$

The shoulder is $-d_{ucs}$ away in the y direction. The minus comes from the orientation of the card. Here we take the right shoulder, thus we need to go in the negative direction. $\mathbf{q}_{uc}$ is the orientation quaternion of the upper chest sensor. $\mathbf{p}_{uc}$ is in the position of the upper chest sensor. $\mathbf{p}_s$ gives the position of the shoulder.

Once we have the position of the right shoulder, we need to know the orientation of the arm card. Thanks to the code described in the section 4.1, we have the quaternion of the orientation.

$$d_{sa} = \text{Distance between the shoulder and the arm sensor.} \quad (4.47)$$

$$\mathbf{p}_{arm} = \mathbf{p}_s + \mathbf{C}(\mathbf{q}_{arm}) \begin{bmatrix} -d_{sa} & 0 & 0 \end{bmatrix}^T \quad (4.48)$$

4.3.4 Constraints between arm and forearm

In this section we will define the constraints between the arm and the forearm. First let's define some distances:

$$d_{ae} = \text{distance between the arm sensor and the elbow} \quad (4.49)$$

First we want to find the position of the elbow following the same reasoning as before we get:

$$\mathbf{p}_{elbow} = \mathbf{p}_{arm} + \mathbf{C}(\mathbf{q}_{arm}) \begin{bmatrix} -d_{ae} & 0 & 0 \end{bmatrix}^T \quad (4.50)$$

We take $-d_{ae}$, because we need to go in the negative direction in order to find the position of the elbow. Once we have the elbow's position we can find the forearm position:

$$d_{ef} = \text{distance between the elbow and the forearm sensor} \quad (4.51)$$

$$\mathbf{p}_{forearm} = \mathbf{p}_{elbow} + \mathbf{C}(\mathbf{q}_{forearm}) \begin{bmatrix} -d_{ef} & 0 & 0 \end{bmatrix}^T \quad (4.52)$$

We go in the $-x$ direction in the forearm frame for a distance of d_{ef} .

4.3.5 Consequences of the constraints on the model

In this subsection, we defined some constraint that define the way our model work. These constraint need to be respected and will be implemented in the process part of the Kalman filter algorithm.

If we look more closely at the constraint, we see that there is a rotation that needs to be performed for the distance vector. For the sake of the notation we denoted it $\mathbf{C}(\mathbf{q})$. In practice, the rotation is applied using the quaternion, not by creating a rotation matrix, but by applying the rotation as in equation 4.3.

4.4 Definition of the model

4.4.1 Definition of the state variables

The model variables are defined as follows:

$$\mathbf{p}_{uc} = \text{coordinates of the upper chest sensor.} \quad (4.53)$$

$$\mathbf{q}_{uc} = \text{quaternion representing the orientation of the upper chest sensor.} \quad (4.54)$$

$$\mathbf{v}_{uc} = \text{speed of the upper chest sensor.} \quad (4.55)$$

$$\mathbf{a}_{uc} = \text{acceleration in the local frame of the upper chest sensor.} \quad (4.56)$$

$$\mathbf{g}_{uc} = \text{angular rate of the upper chest sensor.} \quad (4.57)$$

$$\mathbf{p}_a = \text{coordinates of the abdomen sensor.} \quad (4.58)$$

$$\mathbf{q}_a = \text{quaternion representing the orientation of the abdomen sensor.} \quad (4.59)$$

$$\mathbf{g}_a = \text{angular rate of the upper chest sensor.} \quad (4.60)$$

$$\mathbf{p}_{arm} = \text{coordinates of the arm sensor.} \quad (4.61)$$

$$\mathbf{q}_{arm} = \text{quaternion representing the orientation of the arm sensor} \quad (4.62)$$

$$\mathbf{g}_{arm} = \text{angular rate of the arm sensor} \quad (4.63)$$

$$\mathbf{p}_{forearm} = \text{coordinates of the forearm sensor.} \quad (4.64)$$

$$\mathbf{q}_{forearm} = \text{quaternion representing the orientation of the forearm sensor.} \quad (4.65)$$

$$\mathbf{g}_{forearm} = \text{angular rate of the forearm sensor.} \quad (4.66)$$

It is noteworthy to mention that each sensor possesses its respective quaternion, position, and angular velocity as state variables. An exception is made for the upper chest sensor, which also includes speed and acceleration in the local frame as a state variable. This distinction arises due to the hierarchical structure of the model.

4.4.2 Process of the extended Kalman filter

Since we define the acceleration in the local frame, a simple Kalman filter cannot adequately model the system. Instead, an Extended Kalman Filter (EKF) is required. To update the position of the head sensor, a rotation must be applied to the acceleration variable.

As previously discussed in Equation 4.8, this rotation introduces nonlinearity into the model due to the multiplication of the rotation quaternion with the acceleration vector. Consequently, this necessitates the use of an extended Kalman filter.

In this section we will thus write for every variable defined above the function to predict it.

For the variable $\mathbf{p}_{uc}$, equation 4.42 should be used. But we need to make some changes, in fact the state acceleration of the upper chest sensor is in the local frame. By changing that using equation 4.8, we will get the following equation for motion equation. Let us define:

$$\mathbf{p}_{uc} = [x, y, z] \quad (4.67)$$

$$\mathbf{q}_{uc} = [q_0, q_1, q_2, q_3] \quad (4.68)$$

$$x_k = x_{k-1} + v_x * \Delta t + (2(q_0^2 + q_1^2) - 1) * a_x * \frac{\Delta t^2}{2} + 2 * (q_1 q_2 - q_0 q_3) * a_y * \frac{\Delta t^2}{2} + 2 * (q_1 q_3 + q_0 q_2) * a_z * \frac{\Delta t^2}{2} \quad (4.69)$$

$$y_k = y_{k-1} + v_y * \Delta t + 2(q_1 q_2 + q_0 q_3) a_x * \frac{\Delta t^2}{2} + (2(q_0^2 + q_2^2) - 1) a_y * \frac{\Delta t^2}{2} + 2(q_2 q_3 - q_0 q_1) a_z * \frac{\Delta t^2}{2} \quad (4.70)$$

$$z_k = z_{k-1} + v_z * \Delta t + 2(q_1 q_3 - q_0 q_2) a_x * \frac{\Delta t^2}{2} + (q_2 q_3 + q_0 q_1) a_y * \frac{\Delta t^2}{2} + (2(q_0^2 + q_3^2) - 1) a_z * \frac{\Delta t^2}{2} \quad (4.71)$$

These equations introduce nonlinearity into the function f , thereby necessitating the use of an extended Kalman filter. The Kalman filter process for computing a quaternion has been detailed in Section 4.1.

In the process model, the angular rate for each sensor is assumed to be constant, as is the acceleration of the upper chest sensor. These variables will be updated during the measurement phase, as described later.

For the position of the abdomen, arm and forearm sensor the process was shown in section 4.3. Here, we provide a more detailed explanation to incorporate the temporal evolution of these variables:

$$\mathbf{p}_a^{(k)} = \mathbf{p}_{uc}^{(k)} + \mathbf{C}(\mathbf{q}_{uc}^{(k-1)}) \begin{bmatrix} -d_{bodylength} & 0 & 0 \end{bmatrix} \quad (4.72)$$

$$\mathbf{p}_s^{(k)} = \mathbf{p}_{uc}^{(k)} + \mathbf{C}(\mathbf{q}_{uc}^{(k-1)}) \begin{bmatrix} 0 & -d_{ucs} & 0 \end{bmatrix} \quad (4.73)$$

$$\mathbf{p}_{arm}^{(k)} = \mathbf{p}_s^{(k)} + \mathbf{C}(\mathbf{q}_{arm}^{(k-1)}) \begin{bmatrix} -d_{sa} & 0 & 0 \end{bmatrix} \quad (4.74)$$

$$\mathbf{p}_{elbow}^{(k)} = \mathbf{p}_{arm}^{(k)} + \mathbf{C}(\mathbf{q}_{arm}^{(k-1)}) \begin{bmatrix} -d_{ae} & 0 & 0 \end{bmatrix} \quad (4.75)$$

$$\mathbf{p}_{forearm}^{(k)} = \mathbf{p}_{elbow}^{(k)} + \mathbf{C}(\mathbf{q}_{forearm}^{(k-1)}) \begin{bmatrix} -d_{ef} & 0 & 0 \end{bmatrix} \quad (4.76)$$

If the position of one sensor depends on the position of another, the required positions must be computed at time k before calculating the position at time k for the sensor in question.

While for the position we use the estimate at time k , it is not the case for the quaternion. This approximation is made to avoid too much computing. In fact, if we wanted to use the quaternion at time k , we would need to first do a measurement for the quaternion and then process to update positions.

We approximate that change in orientation between two step is marginal. As we will see later given the time between two steps and the error due to other part in the model. This approximation doesn't seem to degrade the model performance.

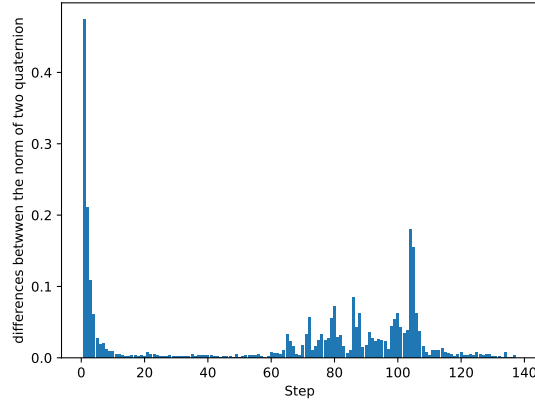


Figure 4.4: Norm of the difference between two consecutive quaternions

Figure 4.4 shows how much the quaternion change between two steps of the filter. As we can see there is a big change in the beginning, it is the difference between the initial calibration and the moment we launch the measurement.

The experiment here was a subject walking 2m, on a straight line. The experiment inherently should not have a lot of change in the quaternions. This shows that the approximation is good for this kind of experiment.

Nonetheless, it is mandatory to check that it holds when doing an experiment. If we do an experiment with a lot of changes in the quaternions, it is important to know that an error will be induced.

This error will be reduced as the time between two iterations decreases. We will see later that the time between iteration is in the order of a $100[ms]$, thus our approximation, may induce a delay of this time between what is predicted and what is actually happening.

4.4.3 Definition of the Jacobian matrix

As previously discussed, the nature of the prediction function necessitates the use of an Extended Kalman Filter (EKF). A key distinction between the EKF and the standard Kalman filter lies in the requirement to compute the Jacobian matrix. In our specific application, only the process model function exhibits non-linearity, thereby obviating the need to compute the Jacobian for the measurement function.

The calculation of the Jacobian matrix has been completed, and the detailed

results are presented in the appendix A.

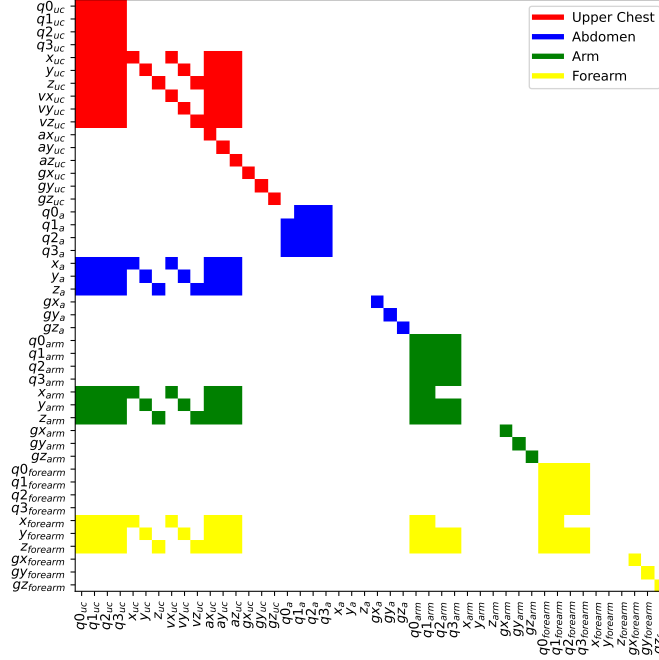


Figure 4.5: Color map of the Jacobian matrix

Figure 4.5, shows a color map of the Jacobian of the process function defined in the previous subsection. The y axis shows which function we derivative, while the x-axis shows with respect to which variable we derivative.

This Jacobian matrix can be interpreted as the F matrix in a regular Kalman filter, described in section 3. In fact, when the function is linear, the Jacobian matrix and the F matrix are equivalent. The color shows to which sensor's variable the derivative belongs. Every line in red is the derivative for a variable relating to the upper chest, every line in blue is the derivative for the position of the abdomen,..

The matrix is notably sparse and exhibits an almost lower triangular form, which is a direct consequence of the hierarchical nature of the model. The figure clearly demonstrates that the derivatives of variables related to the upper chest sensor are exclusively dependent on the corresponding sensor variables.

The abdomen sensor's derivatives are depending on the abdomen and the up-

per chest sensor. The derivative of the position are depending on the orientation, the position, the speed and the acceleration of the upper chest sensor. Interestingly, the quaternion of the abdomen sensor does not affect its position; rather, only the orientation of the upper chest sensor has an impact. This observation aligns with the equations presented in Section 4.3.2. The orientation of the abdomen sensor's only purpose is to have a better representation of the subject being tracked but it won't influence any other variable.

The arm sensor position's derivative alike the abdomen sensor depend on the position, speed and acceleration of the upper chest sensor. It also depend on its own quaternion.

The forearm sensor position's derivative has exactly the same value as the arm sensor derivative but has more. In fact since we have a hierarchical model to have the position of the forearm we first need to calculate the position of the arm and from there we can deduce the position of the forearm using its quaternion, as described in the previous section.

Each quaternion derivative depends on its own value and the value of its angular velocity, which is in harmony with the equation described in section 4.1.3.

4.4.4 Measurement process

In this subsection, we detail the measurement process. Upon receiving data from a sensor, a distinct measurement step is required depending on the sensor.

Starting with the upper chest sensor, measurements include the quaternion($\mathbf{q}_{uc}$), acceleration ($\mathbf{a}_{uc}$), and angular velocity ($\mathbf{g}_{uc}$). The acceleration and angular velocity are directly obtained from the sensor data, while the quaternion vector is computed as explained in Section 4.1. No other variables are measured when processing data from the upper chest sensor.

For all the data from the other sensors, the same apply with their respective quaternion and angular velocity. The upper chest sensor is the only one whose acceleration is a state variable.

4.5 Conclusion on the presentation of the model

In this section we have introduced the model in itself and some of its properties. The consequences of the hierarchical properties have been highlighted.

One significant consequence of the model's implementation is its non-drifting property concerning the attitude estimation. This is partly due to the hierarchical structure of the model and the inherent non-drifting characteristic of the Kalman filter when applied to attitude estimation.

The non-drifting property implies that the error in the relative positions of the sensors remains constant over time, rather than increasing. This contrasts with the behavior of absolute position errors, which may accumulate over time.

Chapter 5

Experiment and implementation

This chapter aims to explain how the preceding chapter was implemented using the Hera framework. Additionally, the experiments conducted revealed some weaknesses in the initial code, which will be highlighted along with proposed solutions.

5.1 Hera Framework

5.1.1 Description of the Hera framework

Hera, as described in the master thesis "The Hera framework for fault-tolerant sensor fusion on an Internet of Things network with application to inertial navigation and tracking," is an Erlang/OTP framework designed for handling asynchronous and dynamic measurements. This framework enables robust communication and data transfer between nodes in an IoT network, with a particular focus on fault tolerance and computational efficiency. [13]

A key feature of Hera is its ability to facilitate the connectivity of GRiSP boards via WiFi, enabling data exchange between nodes. Each GRiSP board is capable of both retrieving data from other nodes and performing local computations, a crucial functionality for maintaining system reliability in the event of node failures. In such cases, unaffected nodes continue their operations seamlessly, while the failed node can restart and reintegrate into the network without causing disruption.

The current implementation integrates a Kalman filter into the Hera framework using Native Implemented Functions (NIFs), alongside migration to the GRiSP2 board, which collectively reduces computation time substantially.[8]

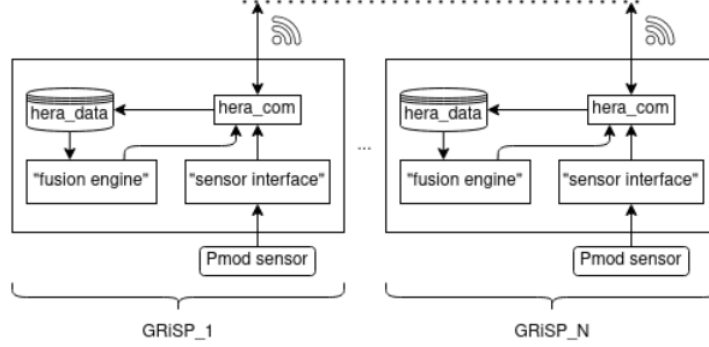


Figure 5.1: Data flow

Figure 5.1 illustrates the interaction between the different Hera modules and the data flow among the GRiSP boards. This figure helps visualize how data is transferred and processed within the network, highlighting the decentralized nature of the computations and the fault-tolerant capabilities of the system.

In Hera, each measurement initiates a supervisor and a child process, ensuring that if the child process fails, the supervisor can restart it. The supervisor can be configured to define how many times and at what frequency the child process should be restarted. The child process is responsible for executing computations (e.g., the Kalman filter) and distributing the results to all connected GRiSP boards, as demonstrated in Figure 5.1.

5.1.2 Detail of implementation

The implementation described in this thesis differs from previous approaches used in earlier related works .

In prior implementations, each GRiSP board initiated two measurements within the Hera framework: one to transmit data from the Pmod Nav, and another to send the calculated state. [13][8].

As we will see later in this chapter and as it was shown in the preceding master

thesis, a majority of the time taken between two step of the Kalman filter is due to communication, or other steps but not computation. [8]

By eliminating one of the measurement processes, we effectively halved the frequency at which each GRiSP board must transmit data to other GRiSP boards. Although the precise time savings from this adjustment were not quantified, it was retained as a strategic implementation choice to enhance system efficiency.

5.2 Naive implementation

In the section, the first implementation of the model as described in the chapter 4 will be discussed. We will Further discuss, the flaw and strength of this implementation and the necessity of a different implementation.

5.2.1 The naive implementation

The idea behind this implementation is to follow exactly the model as described in the chapter 4 and changing the way measurement is done in the preceding master thesis.[8][13] As it was discussed in the preceding section.

At each measurement executed by the Pmod Nav, we will send this data to the network of connected card. Upon receiving the data, each card will check its timeliness. If the data is newer than the last received and not older than 500 milliseconds, the model will use it to make a prediction and update accordingly, as specified in Chapter 4. The 500-millisecond threshold ensures the data remains relevant for real-time prediction, as older data would likely be ineffective.

In this section, we will measure how good different model do. It is thus really important to be able to judge the accuracy of a model. The nature of the problem, we are trying to solve, make it hard to replicate many times the same experiment. We choose to do one time the experiment and keep the data registered to calculate the result for all the models. We in fact only need the data from the accelerometer, gyro and magnetometer at each time step to replicate the experiment with an other model. The experiment consist of making a subject walk 2m in a straight line wearing the upper chest and abdomen sensor and standing still at the 2m mark.

This way of operating while having the advantage of evaluating all the model on the same data is not totally fair. In fact when experimenting, we decided to make a model run at the same time so that we don't bias our result. Having a model run make the time between registering data longer since we have to make

computation for the model to run. We decided in this case to run the biggest model of the following experiments while registering the data. This will certainly negatively affect the result of the computationally less demanding model even if this effect might be marginal as we will see later in this section.

5.2.2 First model

The first model is the lightest computation wise. It consists of putting only the upper chest sensor and only using the prediction step described in section 4.2.

The experiment consist of a subject 2m walking on a straight line, we will use the notation as described in the Figure 4.3 implying that the x axis should be aligned with the gravity. The combined distance walked along the z and y axis is supposed to equal 2m meter, while the distance traveled along the x axis, aligned with the gravity should be null.

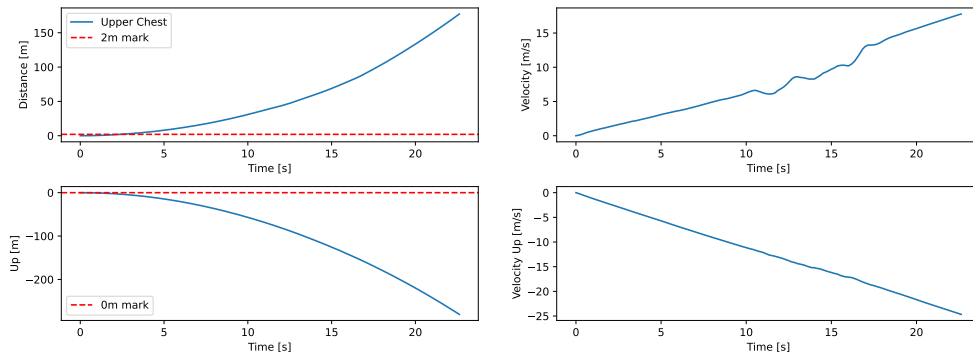


Figure 5.2: Above: distance traveled and velocity along the East and North axis. Under: position and velocity on the up axis

As we can see on the the Figure 5.2, there is a huge drift on the position and the velocity using this model. It seems like the model detects that the subjects start moving but doesn't quiet understand when the subject stops. The velocity stays high even at the end when the subject stopped moving, creating a big drift on the position. Also we can see that the velocity is getting way too high for someone walking at a slow pace.

The result by themselves are not really relevant since the experiment is not replicable, (it is impossible to have someone walking at exactly the same speed) only the comparison of the different model is interesting here.

This results have many explanation, the first one being that the sensor is not totally perpendicular to the ground. In this experiment, we make the hypothesis that the x direction is aligned with the gravity. In practice, the x direction is not perfectly aligned with the gravity. This approximation doesn't work properly, a component of the gravity is actually measured in the y and z direction and harms the prediction. This will lead us to test this code by adding the AHRS system detailed in section 4.1. Also, the measurement error play a role in this error. An error on the gravity has a big impact on the position and the velocity. Since the position and the velocity accumulate from one state to the next.

This experiment's point was not to give good result but to have a starting point for the comparison of results.

5.2.3 Second model

In the second experiment, an AHRS was incorporated to assess its impact on the model's performance. We don't approximate the x direction of the upper chest sensor to be aligned with gravity. We use the process explained in section 4.1, to express all the variable in the global frame.

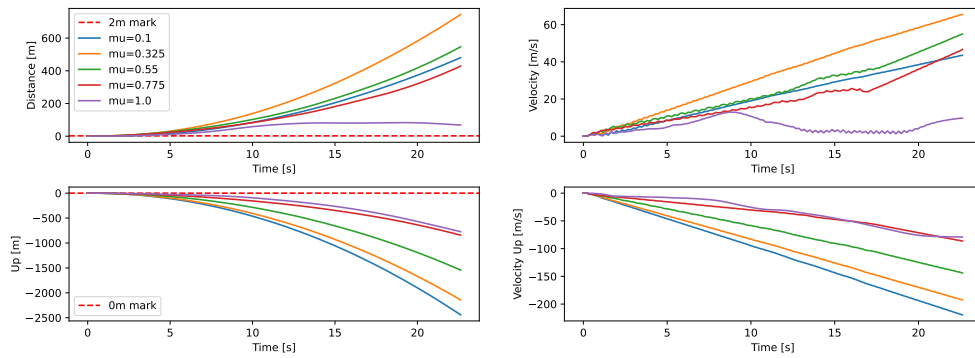


Figure 5.3: Above: distance traveled and velocity along the East and North axis. Under: position and velocity along the up axis

Figure 5.3 shows the comparison of the result with the AHRS for different value

μ_a defined in section 4.1. We can see that for this experiment the value $\mu_a = 1$ is the most appropriate. The experiment's settings is such that the gravity is the principal acceleration which explain this value. The subject traveling with the sensors keeps a low speed and acceleration.

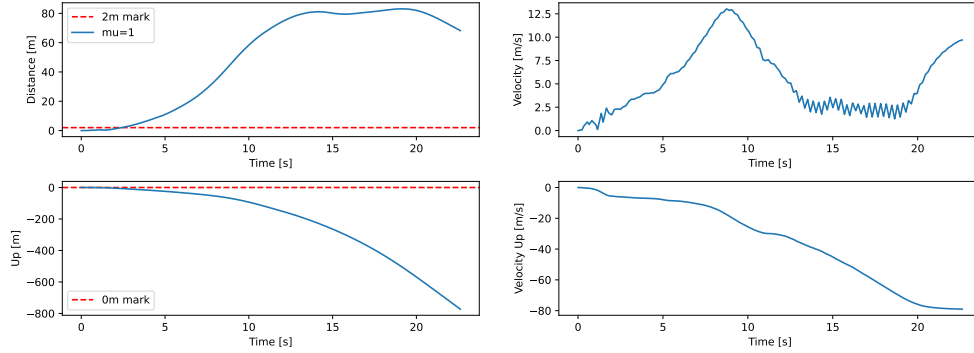


Figure 5.4: Above: distance traveled and velocity along the East and North axis with $\mu_\alpha = 1$. Under: position and velocity along the up axis with $\mu_\alpha = 1$

As shown in Figure 5.4, adding an attitude and heading reference system (AHRS), drastically improved the result. Having a more accurate estimation of the up direction improved the error on the distance traveled, in comparison with last experiment which used the x direction as the up direction.

However we can see that the error on the up direction got worse. Although the total distance traveled is far from being accurate it is better than the last one.

While this experiment gives better result than the first one, it is important to note that the velocity of the sensor is pretty high at the end. This error on the velocity will continue to induce error on the position even if there is no more acceleration. One of the main improvement that we can see on Figure 5.4 is that there is a deceleration. In the first experiment the velocity only increased.

While the distance traveled gave better result it is important to note that the distance on the Up axis has gotten worse. In conclusion the AHRS enabled us to isolate the gravity better. The error due to the gravity that was not well isolated (taken as the up axis) is now reduced. Since the gravity is more isolated it enables us to detect the subject slowing its pace.

One of the main source of error remaining is still the gravity not well isolated. It

seems that when we let the sensor on a still position there is an offset between the gravity it detects and the real gravity, i.e. the gravity measured is not 9.81. This will induce an error on speed and a position.

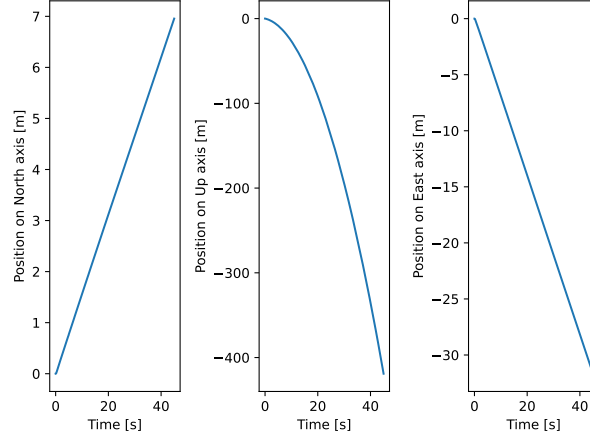


Figure 5.5: Position of the sensor when placed still on North, Up and East axis over time

Figure 5.5 shows the evolution of the position when a sensor is placed still using this model. On these graph we can see that it seems like there is an initial speed on North and East axis and then speed stays constant, meaning there is no more acceleration.

The Up axis graph confirms what we thought earlier, the acceleration measured is not exactly $9.81[m/s^2]$, when subtracting $9.81[m/s^2]$ on the up axis it seems we subtract more than what the accelerometer detects. Resulting in what seems like a constant acceleration on the Up axis.

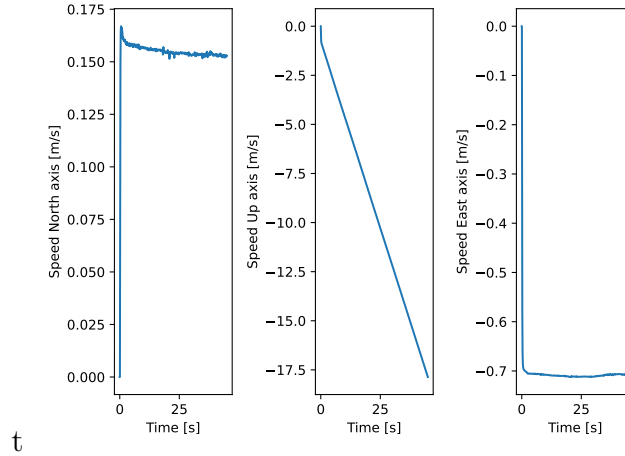


Figure 5.6: Speed of the sensor when place still on North, Up and East axis over time

Figures 5.6 confirms what we thought when interpreting the position graph. It seems that after getting the initial speed the model doesn't perform that bad on the North and East axis. While the speed on the Up axis increases until the end.

5.2.4 Third model

In the last model, we used the AHRS and added constraint between the upper chest and the abdomen sensor

The point of this constraint is to have to sensor taking the same measure. In fact, for this model we will assume that the upper chest and abdomen sensor move at the same speed and that they are place as in Figure 4.3.

If we update the state of the upper chest sensor, we will use this new information to update the abdomen sensor and vice versa. We will here make the assumptions that these two sensors have the same speed.

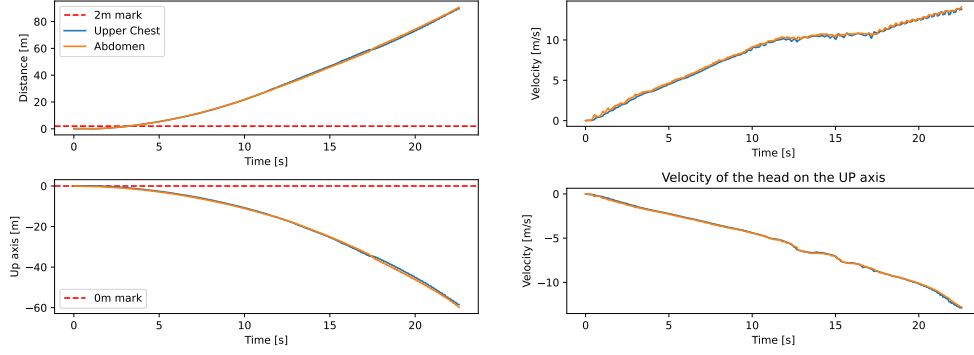


Figure 5.7: Above distance traveled and norm of the velocity along the East and North axis. Under: position and velocity along the up axis. $\mu_a = 1$

It is not explicitly depicted on the Figure 5.7, but we tried once again many values for the parameter μ_a . It is as in the first experiment the value, $\mu_a = 1$ that gave the best result.

Figure 5.7, clearly shows the effect of these constraint on the position and velocity of the upper chest and abdomen sensors.

We can see that adding these constraints have a negative effect on the distance tracking, achieving better result than the first experiment but worst than the second experiment.

On the Up axis tracking, it performed better than the two previous experiment. Regarding the speed, it seems that it doesn't do good. It fails to detect that the subject decelerate at the end.

5.3 Implementation taking advantage of the fast computing time

The previous experiment has uncovered a new challenge, which was also highlighted in a previous master's thesis.[8] Figure 5.8 illustrates the computation times for the Hora framework and various GRiSP cards. The results indicate a significant reduction in computation time with the implementation of NIFs and the GRiSP 2. However, the reading time did not decrease as substantially, and the time labeled as "other" now represents a majority of the total time. This "other" time includes the duration required for sending and fetching data between cards.

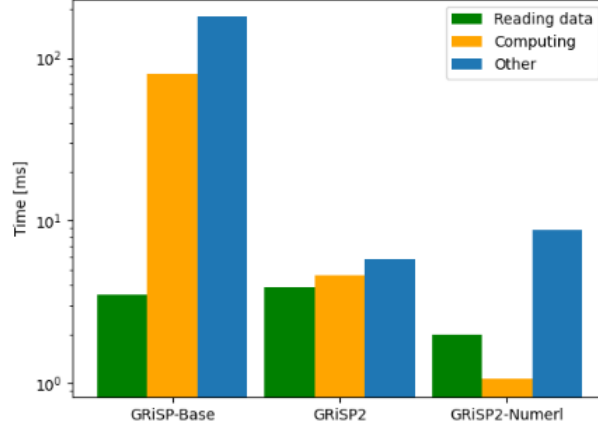


Figure 5.8: Computation time using the Hora framework for an AHRS [8]

With this understanding, we revised our implementation to capitalize on the high-speed processing available for the calculation phase.

In the earlier section, we observed suboptimal results when attempting to predict a sensor's position, partially due to the time elapsed between prediction steps.

Equation 4.42 assumes that acceleration is constant over a Δt interval. This assumption is valid only when Δt is sufficiently small. In the previous experiment, the average Δt was 0.16[s].

In this section, we aim to leverage the reduced computation time to achieve smaller Δt intervals between iterations. This is achieved by performing multiple Kalman filtering locally on a sensor before sending the state to the others sensors. Specifically, we will execute a hundred Kalman filter process and update for each sensor before transmitting the calculated state to other components. This transmission contributes to the "other" time category, as shown in Figure 5.8.

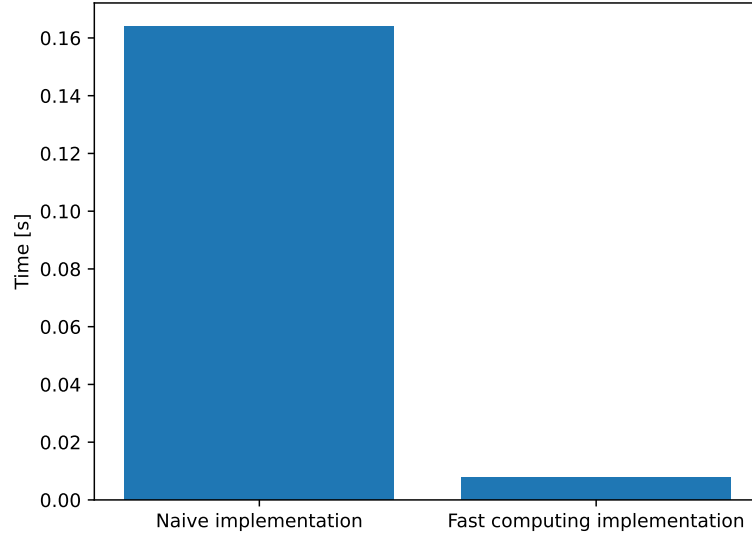


Figure 5.9: Bar plot of the average time between two iterations for the naive implementation and implementation taking advantage of fast computing

Figure 5.9 illustrates the significant reduction in Δt between prediction steps, achieved by optimizing for fast computation. We reduced Δt from 0.16[s] to 0.013[s]. However, this improvement comes at a cost: unlike in the initial experiment where we transferred data after each Kalman iteration. This time we do a hundred prediction and update steps before sending the data.

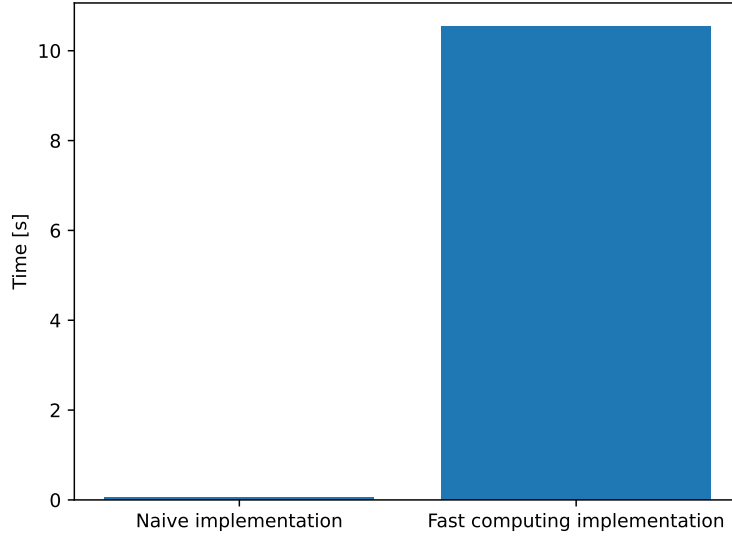


Figure 5.10: Comparison of the time between data transfer

Figure 5.10 highlights the trade-off between these two intervals. Decreasing the Δt between Kalman iteration increases the Δt between data being transferred to other cards. Consequently, we must fine-tune the number of iterations performed before sending the data to others cards to maximize accuracy.

Still, these modifications have significantly improved the performance of our model. In fact we can see on Figure 5.11, that for a model with an AHRS as in Figure 5.3, we outperformed the former implementation.

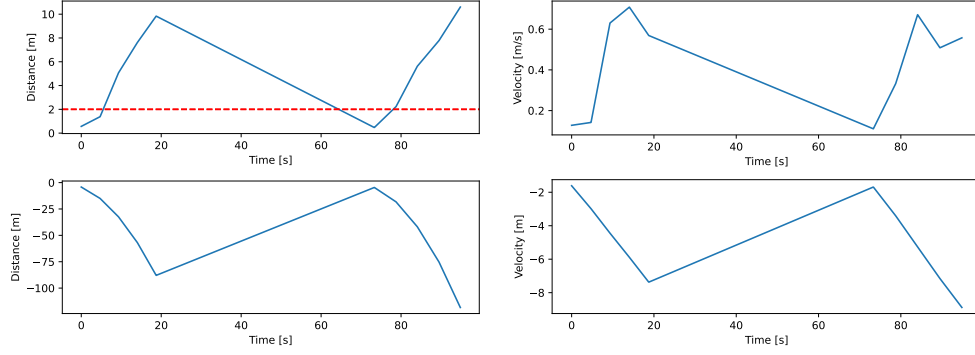


Figure 5.11: Above distance traveled and norm of the velocity along the East and North axis. Under: position and velocity along the up axis. $\mu_a = 1$

It is important to note that while all the experiment from section 5.2 were done on the same data, it is not the case anymore. So these comparison are to take with precaution since we can not control the speed and acceleration of the subject to be exactly the same. Nevertheless, we can confidently say that it improved the performance on such a task.

Figure 5.11 shows that this model seems to perform really well, the distance traveled is not quiet accurate but there is less error that in the other experiments.

The speed for the distance is way better than in the experiment in Figure 5.4, it doesn't get as high. It also seems that there is a sudden burst of acceleration at the end, otherwise the result could have been better velocity wise.

For this experiment we choose a $\mu_a = 1$, as it is this parameter that gave the best result in the preceding experiments.

As we did for the preceding model, we will also check the performance of this implementation while the sensor is lying still.

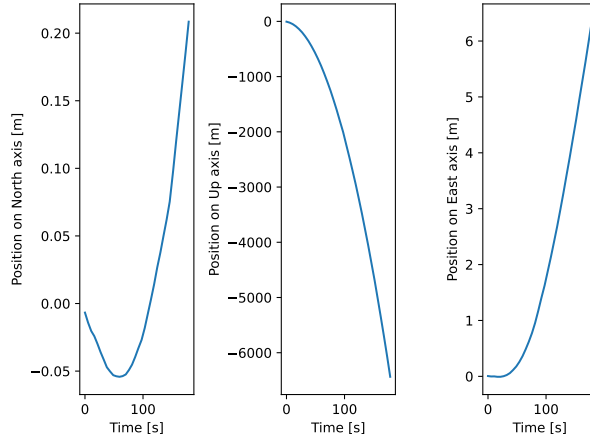


Figure 5.12: Position of the sensor when place still on North, Up and East axis over time

First given the implementation of this model, the time between two data registration is longer than with the previous model. Also since a 100 iteration are done before registering the state, it explains why the first position registered is not 0 on all axis.

Comparing the result of Figure 5.12 to the one on Figure 5.5 we can see a significant improvement on the position prediction for the *North* and *East* axis. While the position on the Up axis got worse.

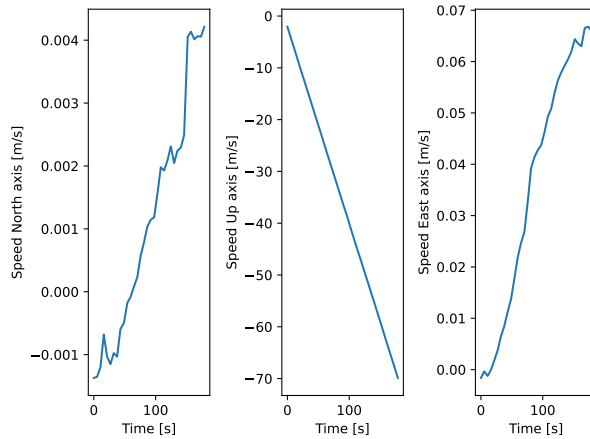


Figure 5.13: Speed of the sensor when placed still on North, Up and East axis over time

The speed of the sensor detailed on Figure 5.13 is relatively stable for the *North*

and *East* axis. It does work far better than with the previous implementation shown on Figure 5.6. It seems that the *Up* direction has a constant acceleration, like in the preceding experiment

5.4 Conclusion on the model

This section emphasized the importance of computation time between steps for result precision. The findings clearly indicate that reducing computation time enhances result accuracy.

However, latency between two steps is still a bottleneck in the implementation and reducing this latency could significantly decrease the error.

Chapter 6

Parameter settings

6.1 Variance for each variable

The data obtained from different sensors necessitates distinct measurement functions tailored to each sensor type. We measure angular velocity and quaternions across all sensors, with an additional measurement of acceleration specifically for the upper chest sensor.

6.1.1 Angular velocity

Angular velocity is computed under the assumption of constancy during the process step. In the measurement step, angular velocity is derived from the readings provided by the Pmod Nav sensor. Given this setup, it is prudent to assign a relatively low variance to the measurement step, as it is directly observed and hence assumed to be precise. Conversely, a larger variance is assigned to the process step since in practice we know that the constancy assumptions does not hold.

6.1.2 Quaternions

The measurement of quaternions differs from that of angular velocity. Quaternions are not directly obtained from the Pmod Nav sensor; instead, they are computed as detailed in Section 4.1.4. The process step for quaternions is dynamic, utilizing angular velocity to predict subsequent orientations. A key challenge in determining appropriate variances arises from the absence of an experimental setup that provides a known “true” orientation while the sensor is in motion. Given this limitation, an equal variance has been adopted for both the process and measurement steps, balancing the uncertainties inherent in both stages.

6.1.3 Acceleration of the Upper Chest

The upper chest sensor is unique in that it measures acceleration, in addition to angular velocity and quaternions. For the acceleration measurements, a similar approach to that used for angular velocity has been employed, assigning an almost negligible variance to the measurement step. This reflects the high confidence in the acceleration data provided by the sensor.

6.1.4 Position and speed variances

Position and speed are not directly measured by the sensors in this study, rendering their associated variances less impactful on the overall model output. Since adjustments to these variances do not influence the results for these variables, they primarily serve to indicate the model's confidence in its predictions. As such, these variances can be set in a way that reflects the model's expected uncertainty in these predicted states.

Chapter 7

What are the next steps ?

7.1 Decreasing the time interval between two steps

7.1.1 Why reducing the time interval

The primary objective of the previous thesis was to improve computational efficiency, focusing on reducing the time between successive steps in the algorithm. The goal was to minimize the time required for calculations, thereby decreasing the interval between updates.

To assess the impact of these improvements, a comparative experiment was conducted, mirroring the conditions described in Figure 5.2. This experiment employed the Hera framework in its state prior to the modifications introduced in the preceding thesis ([8]), specifically without the inclusion of Native Implemented Functions (NIFs).

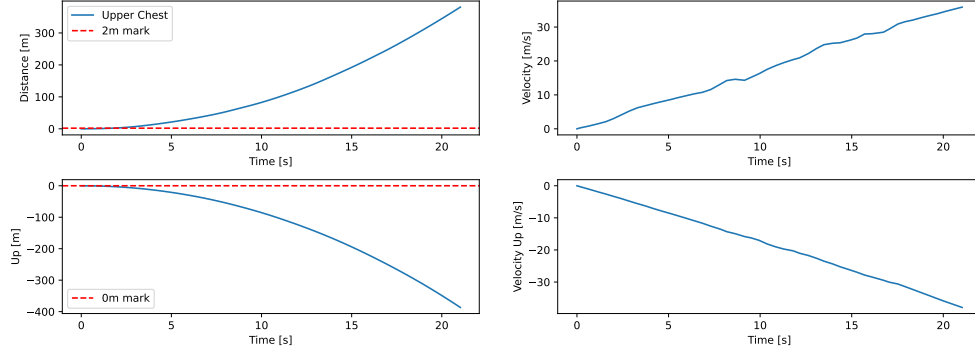


Figure 7.1: 2m walk without the NIFs. Above: distance traveled and velocity along the East and North axis. Under: position and velocity on the up axis

The time taken between update step of the Kalman filter, increased from $160[ms]$ to more than $500[ms]$. This increase in time drastically damaged the model's performance. Consequently, further reducing the time between iterations remains a critical focus area for enhancing model performance.

7.1.2 Using fewer GRiSP cards

Right now, the model is using one GRiSP card by member of the body, one for the upper chest, one for the abdomen, one for the arm and one for the foerarm.

This strategy offers several advantages, such as system redundancy. If one card fails, the remaining cards continue to function, allowing uninterrupted data recording and enabling reconnection of the failed card without halting the entire system.

However, as depicted on Figure 7.2 a majority of time taken between two steps is not spent on computation.

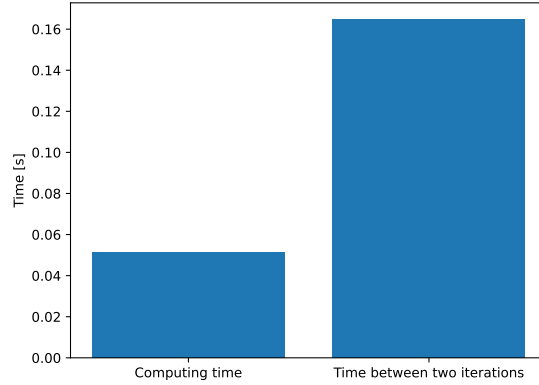


Figure 7.2: Computation time for a step of the Kalman algorithm and time between two steps of the Kalman algorithm

To optimize performance, it is crucial to eliminate non-computational delays, as indicated in Figure 7.2. This reduction would significantly decrease the time between Kalman filter steps and, consequently, improve overall system performance. Currently, the mean computation time for a single Kalman filter step is 0.051[s], while the time between iteration is 0.16[s] .

In section 5.3 we tried using one card that does 100 iteration before registering the state. We saw that this implementation increased the frequency of a Kalman filter step and at the same time the performances of the model. From that experiment we conclude that with the changes depicted here we should improve the performances, since this change would also increase the frequency of a Kalman filter step.

7.1.3 Changing the solver

To reduce the calculation time a NIFs implementation was made as explained in the thesis "low-cost high-speed sensor fusion with grisp and hera".[8].

Given the nature of the problem, the matrices to be solved are predominantly sparse. While the NIFs implementation significantly enhanced calculation speed, changing the solver could yield additional performance improvements.

The current implementation uses Gauss-Jordan elimination to invert the matrix $R + HP_k^-H^T$ in the Kalman gain equation. Although this method is functional, alternative methods, such as QR decomposition, have been proposed in the litera-

ture as more efficient and numerically stable solutions. [9]

While theoretically superior, the practical benefits of switching to a different decomposition method may be less pronounced compared to the potential gains from using only one GRiSP cards, as discussed in the preceding subsection.

7.2 Optimizing the hyperparameters

One of the main point to have more accomplished model is the optimization of parameter. The main difficulty is the lack of equipment to optimize the parameters. This section elaborates on the types of equipment needed for parameters optimization.

The optimization challenge is divided into two parts: position tracking and orientation (attitude) tracking.

7.2.1 Attitude tracking

To optimize the parameters for the attitude tracking component, it is essential to conduct numerous experiments across diverse settings. As discussed in "A new quaternion-based kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm" ([5]), several robust attitude tracking systems exist that could serve as references. Utilizing these systems as benchmarks would enable more accurate parameter tuning.

7.2.2 Position tracking

For position tracking, a variety of camera-based systems are available that can provide accurate positional references. These systems could be leveraged to calibrate and optimize the position tracking parameters within our model.

Chapter 8

Conclusion

In this master’s thesis, we first developed an attitude tracking system, which is a stepping stone for the project. This system transforms the raw data from the Pmod Nav sensors into a global coordinate system common to all sensors. Although the attitude tracking was initially addressed in previous work, further refinements were made in this thesis, including the addition of a parameter to enhance precision.

Key architectural decisions were also made regarding the model structure. One of the most significant choices was the adoption of a hierarchical model. This model’s key advantage lies in its ability to prevent drifting errors in the relative positions of the sensors. Had this choice not been made, the relative error between sensors would have been prone to drift, potentially leading to uncontrolled error propagation, as evidenced by our results.

Extensive experimentation was conducted to evaluate the model’s performance. Initial tests, conducted while the system was stationary, yielded promising results, with random walk phenomena observed but remaining within acceptable limits, except in the vertical (up) direction.

However, the experiment involving a 2-meter walk in a straight line highlighted some significant shortcomings. In the initial model, which lacked an Attitude and Heading Reference System (AHRS), the model failed to detect the subject’s deceleration. A second model did detect the deceleration, but the initial acceleration was significantly overestimated compared to the expected real-world values. A third model attempted to improve accuracy by combining data from multiple sensors, but this approach also failed to adequately detect deceleration.

A final experiment aimed to determine whether increasing the frequency of the Kalman filter steps could enhance performance. The results indicated a clear im-

provement in accuracy with higher frequency updates, underscoring the importance of this parameter in future implementations.

In conclusion, we identified several promising avenues for further improvement. To reduce latency, consolidating all sensors onto a single GRiSP card and exploring alternative solvers were proposed. Additionally, we discussed the potential benefits of employing more robust attitude and position tracking systems to better calibrate hyperparameters. These steps are expected to significantly enhance the model's performance in future iterations.

Chapter 9

Bibliography

- [1] Erlang – NIFs — erlang.org. <https://www.erlang.org/doc/tutorial/nif.html>. [Accessed 02-05-2024].
- [2] Erlang for embedded systems. <https://www.erlang.org/doc/readme.html>, 2024. Accessed: 2024-07-28.
- [3] Diligent. Pmod nav reference manual. <https://reference.digilentinc.com/reference/pmod/pmodnav/reference-manual>, 2024. Accessed: 2024-07-28.
- [4] Engineerine. True north vs. magnetic north, n.d. Accessed: 2024-05-22.
- [5] Kaiqiang Feng, Jie Li, Xiaoming Zhang, Chong Shen, Yu Bi, Tao Zheng, and Jun Liu. A new quaternion-based kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm. *Sensors*, 17(9), 2017.
- [6] Felix Govaers. *Introduction and Implementations of the Kalman Filter*. IntechOpen, Rijeka, May 2019.
- [7] GRiSP Project. Grisp official website. <https://www.grisp.org/>, 2024. Accessed: 2024-07-28.
- [8] Nélis Lucas. low-cost high-speed sensor fusion with grisp and hera, 2022-2023.
- [9] DAVID FUERTES RONCERO. A study of qr decomposition and kalman filter implementations. Master’s thesis, KTH, September 2014.
- [10] Soheil Sarabandi and Federico Thomas. Accurate computation of quaternions from rotation matrices. In Jadran Lenarcic and Vincenzo Parenti-Castelli, editors, *Advances in Robot Kinematics 2018*, pages 39–46, Cham, 2019. Springer International Publishing.

- [11] Joan Solà. Quaternion kinematics for the error-state kalman filter, 2017.
- [12] ChangWhan Sul, Soon Jung, and Kwangyun Wohn. Synthesis of human motion using kalman filter. pages 100–112, 01 1998.
- [13] Vincent V ERPOTEN Sébastien K ALBUSCH. The hera framework for fault-tolerant sensor fusion on an internet of things network with application to inertial navigation and tracking, 2020-2021.
- [14] Rong Zhu and Zhaoying Zhou. A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(2):295–302, 2004.

Appendix A

Jacobian matrix of the prediction function

For the derivative defined below, given the size of a state (46 variables), we will only define non zeros derivative.

The quaternion relative to each sensor has a derivative defined as below:

$$\frac{\partial q_0^{(k+1)}}{\partial X} = \begin{bmatrix} \frac{\partial q_0^{(k+1)}}{\partial q_0^{(k)}} = 1 \\ \frac{\partial q_0^{(k+1)}}{\partial q_1^{(k)}} = \frac{-\omega_x^{(k)} * \Delta t}{2} \\ \frac{\partial q_0^{(k+1)}}{\partial q_2^{(k)}} = \frac{-\omega_y^{(k)} * \Delta t}{2} \\ \frac{\partial q_0^{(k+1)}}{\partial q_3^{(k)}} = \frac{-\omega_z^{(k)} * \Delta t}{2} \\ \frac{\partial q_0^{(k+1)}}{\partial \omega_x^{(k)}} = \frac{-\Delta t * q_1^{(k)}}{2} \\ \frac{\partial q_0^{(k+1)}}{\partial \omega_y^{(k)}} = \frac{-\Delta t * q_2^{(k)}}{2} \\ \frac{\partial q_0^{(k+1)}}{\partial \omega_z^{(k)}} = \frac{-\Delta t * q_3^{(k)}}{2} \end{bmatrix}^T \quad (\text{A.1})$$

$$\frac{\partial q_1^{(k+1)}}{\partial X} = \left[\begin{array}{c} \frac{\partial q_1^{(k+1)}}{\partial q_0^{(k)}} = \frac{\omega_x^{(k)} \Delta t}{2} \\ \frac{\partial q_1^{(k+1)}}{\partial q_1^{(k)}} = 1 \\ \frac{\partial q_1^{(k+1)}}{\partial q_2^{(k)}} = \frac{-\omega_z^{(k)} \Delta t}{2} \\ \frac{\partial q_1^{(k+1)}}{\partial q_3^{(k)}} = \frac{\omega_y^{(k)} \Delta t}{2} \\ \frac{\partial q_1^{(k+1)}}{\partial \omega_x^{(k)}} = \frac{q_0^{(k)} \Delta t}{2} \\ \frac{\partial q_1^{(k+1)}}{\partial \omega_y^{(k)}} = \frac{q_3^{(k)} \Delta t}{2} \\ \frac{\partial q_1^{(k+1)}}{\partial \omega_z^{(k)}} = \frac{-q_2^{(k)} \Delta t}{2} \end{array} \right]^T \quad (\text{A.2})$$

$$\frac{\partial q_2^{(k+1)}}{\partial X} = \left[\begin{array}{c} \frac{\partial q_2^{(k+1)}}{\partial q_0^{(k)}} = \frac{\omega_y^{(k)} \Delta t}{2} \\ \frac{\partial q_2^{(k+1)}}{\partial q_1^{(k)}} = \frac{\omega_z^{(k)} \Delta t}{2} \\ \frac{\partial q_2^{(k+1)}}{\partial q_2^{(k)}} = 1 \\ \frac{\partial q_2^{(k+1)}}{\partial q_3^{(k)}} = \frac{-\omega_x^{(k)} \Delta t}{2} \\ \frac{\partial q_2^{(k+1)}}{\partial \omega_x^{(k)}} = \frac{-q_3^{(k)} \Delta t}{2} \\ \frac{\partial q_2^{(k+1)}}{\partial \omega_y^{(k)}} = \frac{q_0^{(k)} \Delta t}{2} \\ \frac{\partial q_2^{(k+1)}}{\partial \omega_z^{(k)}} = \frac{q_1^{(k)} \Delta t}{2} \end{array} \right] \quad (\text{A.3})$$

$$\frac{\partial q_3^{(k+1)}}{\partial X} = \left[\begin{array}{c} \frac{\partial q_3^{(k+1)}}{\partial q_0^{(k)}} = \frac{\omega_z \Delta t}{2} \\ \frac{\partial q_3^{(k+1)}}{\partial q_1^{(k)}} = \frac{-\omega_y \Delta t}{2} \\ \frac{\partial q_3^{(k+1)}}{\partial q_2^{(k)}} = \frac{\omega_x \Delta t}{2} \\ \frac{\partial q_3^{(k+1)}}{\partial q_3^{(k)}} = 1 \\ \frac{\partial q_3^{(k+1)}}{\partial \omega_x^{(k)}} = \frac{q_2 \Delta t}{2} \\ \frac{\partial q_3^{(k+1)}}{\partial \omega_y^{(k)}} = \frac{-q_1 \Delta t}{2} \\ \frac{\partial q_3^{(k+1)}}{\partial \omega_z^{(k)}} = \frac{q_2 \Delta t}{2} \end{array} \right] \quad (\text{A.4})$$

$$\frac{\partial x_{uc}^{(k+1)}}{\partial X} = \left[\begin{array}{l} \frac{\partial x_{uc}^{(k+1)}}{\partial x_{uc}^{(k)}} = 1 \\ \frac{\partial x_{uc}^{(k+1)}}{\partial x_{uc}^{(k)}} = \Delta t \\ \frac{\partial x_{uc}^{(k+1)}}{\partial q_0^{(uc)}} = \Delta t^2 (2q_0^{(uc)} a_x - q_3^{(uc)} a_y + q_2^{(uc)} a_z) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial q_1^{(uc)}} = \Delta t^2 (2q_1^{(uc)} a_x - q_2^{(uc)} a_y + q_3^{(uc)} a_z) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial q_2^{(uc)}} = \Delta t^2 (q_1^{(uc)} a_y + q_0^{(uc)} a_z) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial a_x} = \frac{\Delta t^2}{2} (2((q_0^{(uc)})^2 + (q_1^{(uc)})^2) - 1) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial a_y} = (q_1^{(uc)} q_2^{(uc)} - q_0^{(uc)} q_3^{(uc)}) \Delta t^2 \\ \frac{\partial x_{uc}^{(k+1)}}{\partial a_z} = (q_0^{(uc)} q_2^{(uc)} + q_1^{(uc)} q_3^{(uc)}) \Delta t^2 \end{array} \right] \quad (\text{A.5})$$

$$\frac{\partial y_{uc}^{(k+1)}}{\partial X} = \left[\begin{array}{l} \frac{\partial x_{uc}^{(k+1)}}{\partial y^{uc}} = 1 \\ \frac{\partial x_{uc}^{(k+1)}}{\partial v_y^{uc}} = \Delta t \\ \frac{\partial x_{uc}^{(k+1)}}{\partial q_0} = \Delta t^2 (q_3^{(uc)} a_x + 2q_0^{(uc)} a_y - q_1^{(uc)} a_z) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial q_1} = \Delta t^2 (q_2^{(uc)} a_x - q_0^{(uc)} a_z) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial q_2} = \Delta t^2 (q_1^{(uc)} a_x + 2q_2^{(uc)} a_y + q_3^{(uc)} a_z) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial q_3} = \Delta t^2 (q_1^{(uc)} a_x + 2q_2^{(uc)} a_y + q_3^{(uc)} a_z) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial a_x} = \Delta t^2 (q_1^{(uc)} q_2^{(uc)} + q_0^{(uc)} q_3^{(uc)}) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial a_y} = \frac{\Delta t^2}{2} (2((q_0^{(uc)})^2 + (q_2^{(uc)})^2) - 1) \\ \frac{\partial x_{uc}^{(k+1)}}{\partial a_z} = \Delta t^2 (q_2^{(uc)} q_3^{(uc)} - q_0^{(uc)} q_1^{(uc)}) \end{array} \right] \quad (\text{A.6})$$

$$\frac{\partial z_{uc}^{(k+1)}}{\partial X} = \left[\begin{array}{l} \frac{\partial z_{uc}^{(k+1)}}{\partial z_{uc}^{(k)}} = 1 \\ \frac{\partial z_{uc}^{(k+1)}}{\partial v_z^{(uc)}} = \Delta t \\ \frac{\partial z_{uc}^{(k+1)}}{\partial q_0^{(uc)}} = \Delta t^2 (-q_2^{(uc)} a_x + q_1^{(uc)} a_y + 2q_0^{(uc)} a_z) \\ \frac{\partial z_{uc}^{(k+1)}}{\partial q_1^{(uc)}} = \Delta t^2 (q_3^{(uc)} a_x + q_0^{(uc)} a_y) \\ \frac{\partial z_{uc}^{(k+1)}}{\partial q_2^{(uc)}} = \Delta t^2 (q_3^{(uc)} a_y - q_0^{(uc)} a_x) \\ \frac{\partial z_{uc}^{(k+1)}}{\partial q_3^{(uc)}} = \Delta t^2 (q_1^{(uc)} a_x + q_2^{(uc)} a_y + 2q_3^{(uc)} a_z) \\ \frac{\partial z_{uc}^{(k+1)}}{\partial a_x} = \Delta t^2 (q_1^{(uc)} q_3^{(uc)} - q_0^{(uc)} q_2^{(uc)}) \\ \frac{\partial z_{uc}^{(k+1)}}{\partial a_y} = \Delta t^2 (q_0^{(uc)} q_1^{(uc)} + q_2^{(uc)} q_3^{(uc)}) \\ \frac{\partial z_{uc}^{(k+1)}}{\partial a_z} = \frac{\Delta t^2}{2} (2((q_0^{(uc)})^2 + (q_3^{(uc)})^2) - 1) \end{array} \right] \quad (\text{A.7})$$

$$\frac{\partial x_a^{(k+1)}}{\partial X} = \frac{\partial x_{uc}^{(k+1)}}{\partial X} + \left[\begin{array}{l} \frac{x_a^{(k+1)}}{\partial q_0^{(uc)}} = -4 * q_0^{(uc)} d_{bodylength} \\ \frac{x_a^{(k+1)}}{\partial q_1^{(uc)}} = -4 * q_1^{(uc)} d_{bodylength} \end{array} \right] \quad (A.8)$$

$$\frac{\partial y_a^{(k+1)}}{\partial X} = \frac{\partial y_{uc}^{(k+1)}}{\partial X} + \left[\begin{array}{l} \frac{\partial y_a^{(k+1)}}{\partial q_0^{(uc)}} = -2q_3 d_{bodylength} \\ \frac{\partial y_a^{(k+1)}}{\partial q_1^{(uc)}} = -2q_2 d_{bodylength} \\ \frac{\partial y_a^{(k+1)}}{\partial q_2^{(uc)}} = -2q_1 d_{bodylength} \\ \frac{\partial y_a^{(k+1)}}{\partial q_3^{(uc)}} = -2q_0 b_{bodylength} \end{array} \right] \quad (A.9)$$

$$\frac{\partial z_a^{(k+1)}}{\partial X} = \frac{\partial z_{uc}^{(k+1)}}{\partial X} + \left[\begin{array}{l} \frac{\partial z_a^{(k+1)}}{\partial q_0^{(uc)}} = 2q_2 d_{bodylength} \\ \frac{\partial z_a^{(k+1)}}{\partial q_1^{(uc)}} = -2q_3 d_{bodylength} \\ \frac{\partial z_a^{(k+1)}}{\partial q_2^{(uc)}} = 2q_0 d_{bodylength} \\ \frac{\partial z_a^{(k+1)}}{\partial q_3^{(uc)}} = -2q_1 b_{bodylength} \end{array} \right] \quad (A.10)$$

$$\frac{\partial x_s^{(k+1)}}{\partial X} = \frac{\partial x_{uc}^{(k+1)}}{\partial X} + \left[\begin{array}{l} \frac{\partial x_s^{(k+1)}}{\partial q_0^{(uc)}} = 2q_3 d_{ucs} \\ \frac{\partial x_s^{(k+1)}}{\partial q_1^{(uc)}} = -2q_1 d_{ucs} \\ \frac{\partial x_s^{(k+1)}}{\partial q_2^{(uc)}} = -2q_1 d_{cus} \\ \frac{\partial x_s^{(k+1)}}{\partial q_3^{(uc)}} = 2q_3 d_{cus} \end{array} \right] \quad (A.11)$$

$$\frac{\partial y_s^{(k+1)}}{\partial X} = \frac{\partial y_{uc}^{(k+1)}}{\partial X} + \left[\begin{array}{l} \frac{\partial y_s^{(k+1)}}{\partial q_0^{(uc)}} = -4q_0 d_{ucs} \\ \frac{\partial y_s^{(k+1)}}{\partial q_2^{(uc)}} = -4q_2 d_{ucs} \end{array} \right] \quad (A.12)$$

$$\frac{\partial z_s^{(k+1)}}{\partial X} = \frac{\partial z_{uc}^{(k+1)}}{\partial X} + \left[\begin{array}{l} \frac{\partial y_s^{(k+1)}}{\partial q_0^{(uc)}} = -2q_1 d_{ucs} \\ \frac{\partial y_s^{(k+1)}}{\partial q_1^{(uc)}} = -2q_0 d_{ucs} \\ \frac{\partial y_s^{(k+1)}}{\partial q_2^{(uc)}} = -2q_3 d_{ucs} \\ \frac{\partial y_s^{(k+1)}}{\partial q_3^{(uc)}} = -2q_2 d_{ucs} \end{array} \right] \quad (A.13)$$

$$\frac{\partial x_{arm}^{(k+1)}}{\partial X} = \frac{\partial x_s^{(k+1)}}{\partial X} + \left[\begin{array}{l} \frac{x_{arm}^{(k+1)}}{\partial q_0^{(arm)}} = -4q_0^{(arm)} d_{sa} \\ \frac{x_{arm}^{(k+1)}}{\partial q_1^{(arm)}} = -4q_1^{(arm)} d_{sa} \end{array} \right] \quad (A.14)$$

$$\frac{\partial y_{arm}^{(k+1)}}{\partial X} = \frac{\partial y_s^{(k+1)}}{\partial X} + \begin{bmatrix} \frac{\partial y_{arm}^{(k+1)}}{\partial q_0^{(arm)}} = -2q_3 d_{sa} \\ \frac{\partial y_{arm}^{(k+1)}}{\partial q_1^{(arm)}} = -2q_2 d_{sa} \\ \frac{\partial y_{arm}^{(k+1)}}{\partial q_2^{(arm)}} = -2q_1 d_{sa} \\ \frac{\partial y_{arm}^{(k+1)}}{\partial q_3^{(arm)}} = -2q_0 b_{sa} \end{bmatrix} \quad (A.15)$$

$$\frac{\partial z_{arm}^{(k+1)}}{\partial X} = \frac{\partial z_s^{(k+1)}}{\partial X} + \begin{bmatrix} \frac{\partial z_{arm}^{(k+1)}}{\partial q_0^{(arm)}} = 2q_2 d_{sa} \\ \frac{\partial z_{arm}^{(k+1)}}{\partial q_1^{(arm)}} = -2q_3 d_{sa} \\ \frac{\partial z_{arm}^{(k+1)}}{\partial q_2^{(arm)}} = 2q_0 d_{sa} \\ \frac{\partial z_{arm}^{(k+1)}}{\partial q_3^{(arm)}} = -2q_1 b_{sa} \end{bmatrix} \quad (A.16)$$

$$\frac{\partial x_{elbow}^{(k+1)}}{\partial X} = \frac{\partial x_{arm}^{(k+1)}}{\partial X} + \begin{bmatrix} \frac{x_{elbow}^{(k+1)}}{\partial q_0^{(arm)}} = -4 * q_0^{(arm)} d_{ae} \\ \frac{x_{elbow}^{(k+1)}}{\partial q_1^{(s)}} = -4 * q_1^{(arm)} d_{ae} \end{bmatrix} \quad (A.17)$$

$$\frac{\partial y_{elbow}^{(k+1)}}{\partial X} = \frac{\partial y_{arm}^{(k+1)}}{\partial X} + \begin{bmatrix} \frac{\partial y_{elbow}^{(k+1)}}{\partial q_0^{(arm)}} = -2q_3 d_{ae} \\ \frac{\partial y_{elbow}^{(k+1)}}{\partial q_1^{(arm)}} = -2q_2 d_{ae} \\ \frac{\partial y_{elbow}^{(k+1)}}{\partial q_2^{(arm)}} = -2q_1 d_{ae} \\ \frac{\partial y_{elbow}^{(k+1)}}{\partial q_3^{(arm)}} = -2q_0 b_{ae} \end{bmatrix} \quad (A.18)$$

$$\frac{\partial z_{elbow}^{(k+1)}}{\partial X} = \frac{\partial z_s^{(k+1)}}{\partial X} + \begin{bmatrix} \frac{\partial z_{elbow}^{(k+1)}}{\partial q_0^{(arm)}} = 2q_2 d_{ae} \\ \frac{\partial z_{elbow}^{(k+1)}}{\partial q_1^{(arm)}} = -2q_3 d_{ae} \\ \frac{\partial z_{elbow}^{(k+1)}}{\partial q_2^{(arm)}} = 2q_0 d_{ae} \\ \frac{\partial z_{elbow}^{(k+1)}}{\partial q_3^{(arm)}} = -2q_1 b_{ae} \end{bmatrix} \quad (A.19)$$

$$\frac{\partial x_{forearm}^{(k+1)}}{\partial X} = \frac{\partial x_{arm}^{(k+1)}}{\partial X} + \begin{bmatrix} \frac{x_{forearm}^{(k+1)}}{\partial q_0^{(forearm)}} = -4 * q_0^{(forearm)} d_{ef} \\ \frac{x_{forearm}^{(k+1)}}{\partial q_1^{(s)}} = -4 * q_1^{(forearm)} d_{ef} \end{bmatrix} \quad (A.20)$$

$$\frac{\partial y_{forearm}^{(k+1)}}{\partial X} = \frac{\partial y_{arm}^{(k+1)}}{\partial X} + \begin{bmatrix} \frac{\partial y_{forearm}^{(k+1)}}{\partial q_0^{(forearm)}} = -2q_3 d_{ef} \\ \frac{\partial y_{forearm}^{(k+1)}}{\partial q_1^{(forearm)}} = -2q_2 d_{ef} \\ \frac{\partial y_{forearm}^{(k+1)}}{\partial q_2^{(forearm)}} = -2q_1 d_{ef} \\ \frac{\partial y_{forearm}^{(k+1)}}{\partial q_3^{(forearm)}} = -2q_0 b_{ef} \end{bmatrix} \quad (\text{A.21})$$

$$\frac{\partial z_{forearm}^{(k+1)}}{\partial X} = \frac{\partial z_s^{(k+1)}}{\partial X} + \begin{bmatrix} \frac{\partial z_{forearm}^{(k+1)}}{\partial q_0^{(forearm)}} = 2q_2 d_{ef} \\ \frac{\partial z_{forearm}^{(k+1)}}{\partial q_1^{(forearm)}} = -2q_3 d_{ef} \\ \frac{\partial z_{forearm}^{(k+1)}}{\partial q_2^{(forearm)}} = 2q_0 d_{ef} \\ \frac{\partial z_{forearm}^{(k+1)}}{\partial q_3^{(forearm)}} = -2q_1 b_{ef} \end{bmatrix} \quad (\text{A.22})$$

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl