Architecture of Ultra Low Power Node for Body Area Network

Alexis Aulery

THÈSE / UNIVERSITÉ DE BRETAGNE SUD

sous le sceau de l'Université Bretagne Loire

Pour obtenir le titre de
DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE SUD
Mention : STIC
École Doctorale SICMA

présentée par

Alexis AULERY

Préparée à l'Unité Mixte de Recherche n 6285 Lab-STICC

Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance

UFR Sciences et Sciences de l'Ingénieur

Architecture of Ultra Low Power Node for Body Area Network

Thèse soutenue le 01 Décembre 2016,
devant la commission d'examen composée de :

Pr.Laurent Clavier
Professeur, Télécom Lille - IEMN / Président

Pr. Laurent Clavier
Professeur, Télécom Lille - IEMN / Rapporteur

Dr. Alain Pegatoquet
Maitre de Conference HDR, Université de Nice - LEAT / Rapporteur

Dr. Yannick Le Moullec
Senior Research Scientist HDR, Tallinn University of Technology / Examinateur

Pr. Bernard Uguen
Professeur, Université de Rennes 1 / Examinateur

In. Eric Mercier
Ingenieur-Chercheur, CEA LETI / Examinateur

DR. Jean-Philippe DIGUET
Directeur de thèse / Directeur de recherche CNRS

Pr. Olivier SENTIEYS
Co-directeur de thèse / Directeur de recherche INRIA

Résumé

Le réseau de capteurs porté est une technologie d'avenir prometteuse à multiple domaines d'application allant du médical à l'interface homme machine. Le projet BoWI a pour ambition d'évaluer la possibilité d'élaborer un réseau de capteur utilisable au quotidien dans un large spectre d'applications et ergonomiquement acceptable pour le grand public. Cela induit la nécessité de concevoir un nœud de réseau ultra basse consommation pour à la fois convenir à une utilisation prolongée et sans encombrement pour le porteur.
La solution retenue est de concevoir un nœud capable de travailler avec une énergie comparable à ce que l'état de l'art de la récolte d'énergie est capable de fournir. Une solution ASIC est privilégiée afin de tenir les contraintes d'intégration et de basse consommation.
La conception de l'architecture dédiée a nécessité une étude préalable à plusieurs niveaux qui comprend un état de l'art de la récolte d'énergie afin de fixer un objectif de budget énergie/puissance de notre système. Une étude des usages du système a été nécessaire notamment pour la reconnaissance de postures afin de déterminer les cas applications types. Cette étude à conduit au développement d'algorithmes permettant de répondre aux applications choisies tout en s'assurant de la viabilité de leurs implémentations.
Le budget énergie fixé est un objectif de $100\mu W$. Les applications choisies sont la reconnaissance de posture, la reconnaissance de geste et la capture de mouvement. Les solutions algorithmiques choisies sont une fusion de données de capteurs inertiels par Filtre de Kalman étendu (EKF) et l'ajout d'une classification par analyse en composante principale. La solution retenue pour obtenir des résultats d'implémentation est la synthèse de haut niveau qui permet un développement rapide.
Les résultats de l'implémentation matérielle sont dominés principalement par l'EKF. À la suite de l'étude, il apparait qu'il est possible avec une technologie $28nm$ d'atteindre les objectifs de budget énergie pour la partie algorithme. Une évaluation de la gestion haut niveau de tous les composants du nœud est également effectuée afin de donner une estimation plus précise des performances du système dans un cas d'application réel. Une contribution supplémentaire est obtenue avec l'ajout de la détection d'activité qui permet de prédire la charge de calcul nécessaire et d'adapter dynamiquement l'utilisation des ressources de traitement et des capteurs afin d'optimiser l'énergie en fonction de l'activité.

Abstract

Wireless Body Sensor Network (WBSN) is a promising technology that can be used in a lot of application domains from health care to Human Machine Interface (HMI). The BoWI project ambition is to evaluate and design a WBSN that can be used in various applications with daily usage and accessible to the public. This necessitates to design a ultra-low power node that reach a day of use without discomfort for the user.
The elected solution is to design a node that operates with the power budget similar to what can be provided by the state of the art of the energy harvesting. An Application Specific Integrated Circuit (ASIC) solution is privileged in order to meet the integration and low power constraints.
Designing the dedicated architecture required a preliminary study at several level which are: a state of the art of the energy harvesting in order to determine the objective of energy/power budget of our system, A study of the usage of the system to determine and select typical application cases. A study of the algorithms to address the selected applications while considering the implementation viability of the solutions.
The power budget objective is set to $100\mu W$. The application selected are the posture recognition, the gesture recognition and the motion capture. The algorithmic solution proposed are a data-fusion based on an Extended Kalman FIlter (EKF) with the addition of a classification using Principal Component Analysis (PCA). The implementation tool used to design the architecture is an High Level Synthesis (HLS) solution.
Implementation results mainly focus on the EKF since this is by far the most power consuming digital part of the system. Using a $28nm$ technology the power budget objective can be reached for the algorithmic part. A study of the top level management of all components of the node is done in order to estimate performances of the system in real application case. This is possible using an activity detection which dynamically estimate the computing load required and then save a maximum of energy while the node is still.

n d'ordre : 000000000419

Université de Bretagne Sud

Centre de Recherche Christiaan Huygens - Rue de Saint-Maudé - 56100 Lorient, FRANCE
Tél : + 33(0)2 97 87 45 62 Fax : + 33(0)2 97 87 45 27

Contents

1 Introduction

1.1 Wireless Sensor Network and Body Area Network

Wireless Sensor Network (WSN) is a system of devices distributed in a large or confined space that communicate through radio channels [1]. Devices are called nodes and constitute together a network. A network is characterized by its topology, which defines which node can communicate with each node. Each device is equipped with sensors, which provide samples that are locally processed according to the application. In general case there is an external base station that gathers data for a global processing. Most of the time, we can also find a master node or gateway node which, as called, connects the network with the base station. Figure 1.1 shows several WSN topology examples.

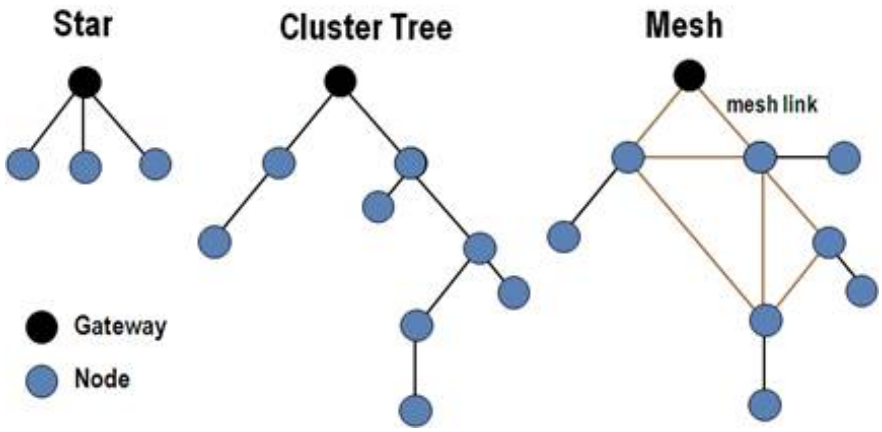


Figure 1.1: Example of WSN topologies.

In case the system is used on a person the system is called a Wireless Body Sensor Network (WBSN) or Body Area Network (BAN) [2]. Applications intended are more user-based and imply different constraints than for usual WSN: the proximity of body from radio antennas, the proximity of nodes from each other, the mobility of nodes on the body. Figure 1.2 shows an example of a BAN topologies.

Note that sometimes external beacons are used for specific applications. In this case, it is called an environment equipped system.

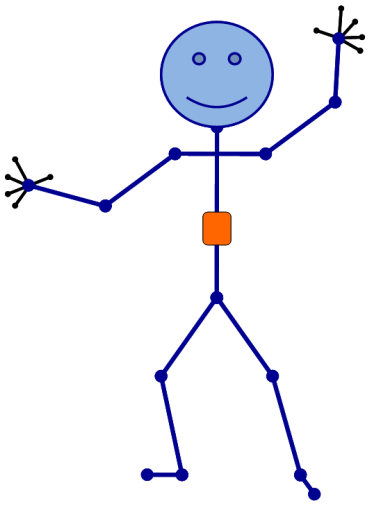


Figure 1.2: Example of a BAN.

1.2 BAN Applications

1.2.1 Interest in BAN

The applications of BAN are numerous. Each application domain leads to different constraints for the system in terms of performance, reliability, life-time, integration, etc.

- Regarding health monitoring, we observe the increase of the self-quantified phenomenon [3]. More and more people use technology devices to monitor their own health index as pulse rate, calories, sleeping time. etc. This increase is assisted by the democratization of smart-phones which provides a convenient device with a relatively high computing capacity. Moreover, the addition of a BAN would be a method to reach higher level of monitoring. Although, for ergonomic and private acceptance purpose, the use of a BAN can be conceivable only if the BAN does not add constraints for user, which means high integration of BAN and energy autonomy.

- In the medical field, BAN is a really useful tool for patient monitoring [4]. Generally, this is a good compromise that allows to adapt the medical equipment to patient concerns. This can reduce the medical equipment discomfort for many patients meanwhile health workers can be focused on care demanding cases. Here in the medical field, BAN has to be reliable, compact and with a good life-time. We can take example of fall detect systems for elder patient, wearable ECG, etc.

- In the movie and broadcast industry field, BAN is mostly used for motion capture [5]. A BAN system is a good alternative to optical systems based on camera that require an expensive and not always needed equipped environment. The motion capture is now a common practice used to produce animated movies, games, virtual systems, etc.

Here the constraints are not in terms of life-time or integration, the concern is about performance and accuracy.

With these application fields, we identify a global necessity to recognize posture or gesture of the user. So we defined three application cases to implement by increasing order of difficulty i.e. performance requirements:

- Posture recognition
- Gesture recognition
- Motion Capture

1.2.2 Considered usages of the BoWI project

This thesis takes place in the context of the BoWI project. The BoWi project stands for Body World Interactions and initially stems from a proposal of the CominLabs think tank focused on the society challenge called Digital Environment for the Citizen. It is also related to the social challenge ICT for Personalized Medicine and to the research track Energy Efficiency in ICT.

The BoWI system intends to be used for a large domain of applications. Figure 1.3 gives an overview of the application space considered: Human Machine Interface (HMI), home automation, health monitoring, sport, video games, etc.

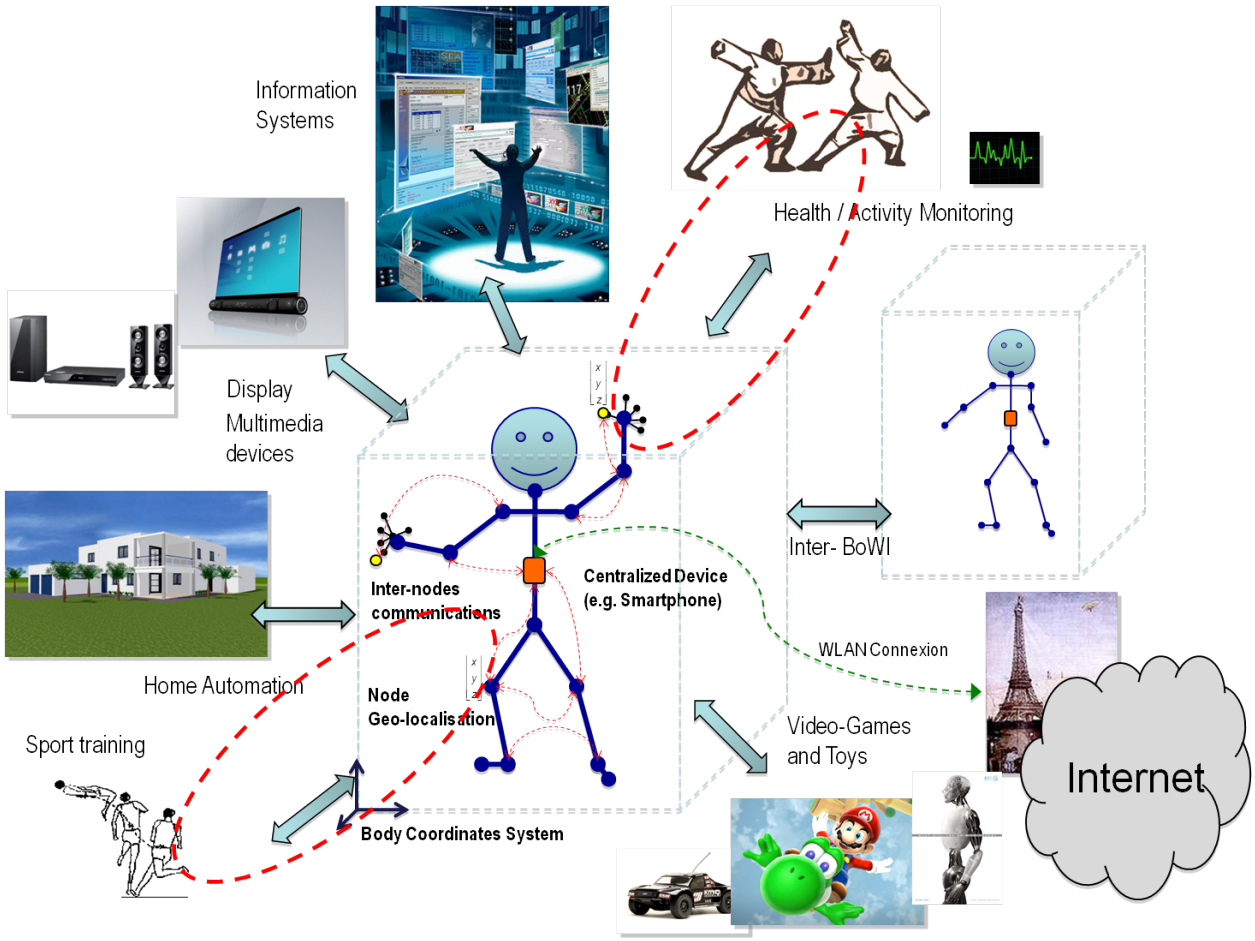


Figure 1.3: BoWI applications.

The BoWI project was also an opportunity to think about new usages. On this context an idea of a posture-based version of the SIMON game has emerged under the guidance of a designer, Clement Gault. This version is based on the tracking of sequences of postures. Several users, each having a BAN, are connected to a device that leads the game. Each player at his/her turn must imitate the sequence of postures of previous player and add personal posture. Figure 1.4 gives an illustration of the progress of the game. This application implies

a posture detection with a dynamic record of new posture.

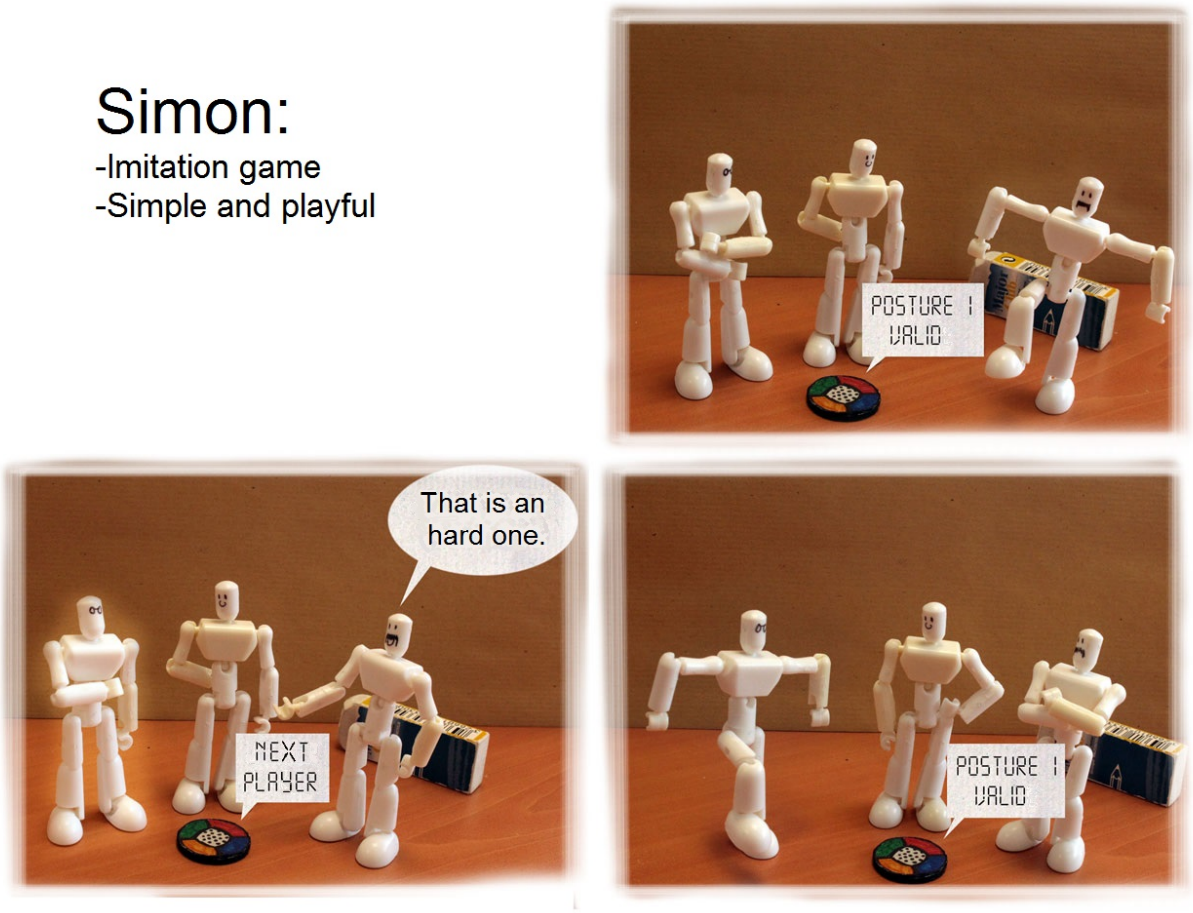


Figure 1.4: BoWI usage: Simon, imitation game.

We can also consider two important applications for health concern. First, Musculoskeletal Disorder (MSD, TMS in french) is by far the main cause of occupational disease (MP in french) in France, see Figure 1.5. The ability of a BAN to detect, quantify and prevent the duration of bad postures would be a big help on the social and economic domains.

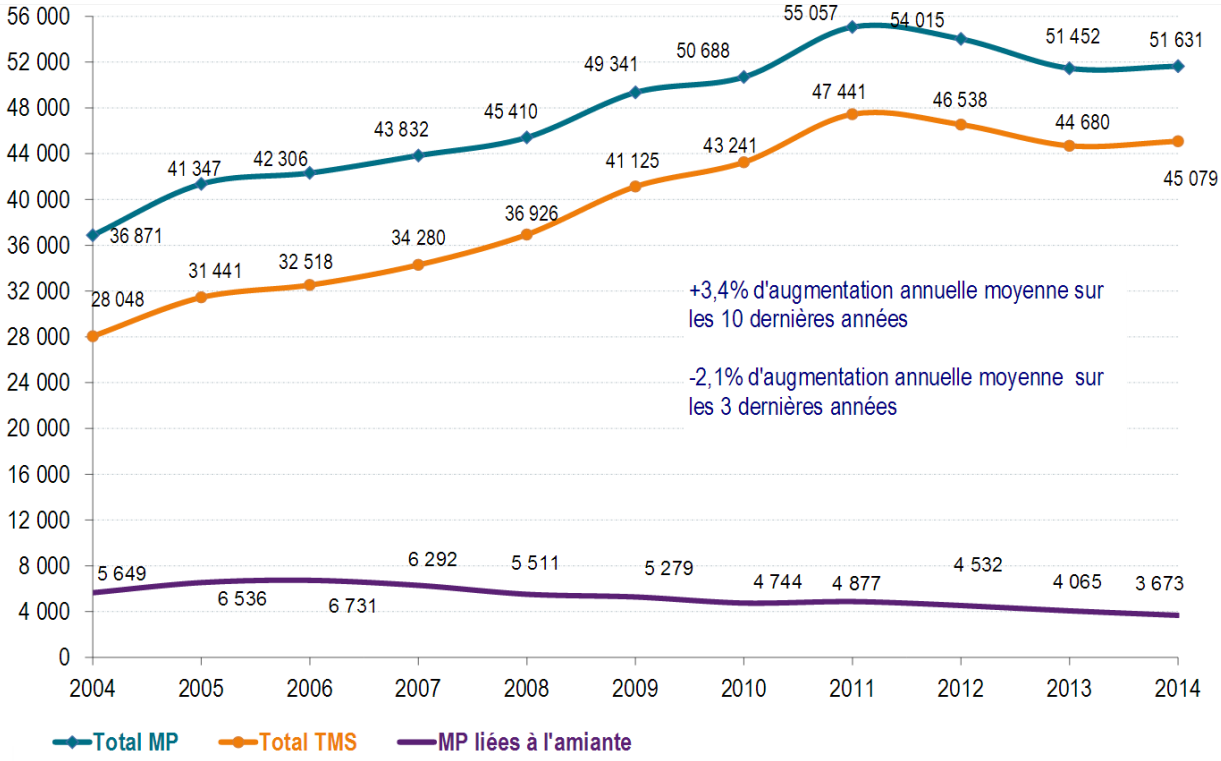


Figure 1.5: Occupational disease cases in France. Source: CNAMTS

Such a posture detection application of the system can be extended to the private domain with a daily monitoring of the user. This brings us the second application of health concern, the rehabilitation. In this case, the system can inform in real-time if the user stands in a bad posture, see Figure 1.6. The system can also be used for the monitoring of distant rehabilitation exercises. Gestures of the exercises are recorded in presence of the practitioner, thus the patient can exercise at home. With the help of the system, the practitioner can monitor the progress of the exercise. Home rehabilitation is an important issue of our society. The transportation to the practitioner generates additional cost in money and time, where it is not always required, especially when it is about physical rehabilitation out of serious health issues.

Figure 1.6: BoWI usage: Functional rehabilitation, Daily monitoring.

1.3 Energy Constraint of the BoWI BAN Nodess

The main constraint of our BAN system is the energy. In our study we consider that the system is energy autonomous. Ideally this means the use of ultra small battery or no battery at all. Is such cases, the system should scavenge its own energy using harvesting method from the environment. We show here the possibility and limitation of the energy harvesting system and comparison with energy reservoir.

There are two strategies: 1) produce the needed energy with energy harvesting or 2) supply the system with batteries to achieve long-life operating of the devices. In this thesis, we mainly focus on supplying nodes with energy harvesting.

Energy harvesting includes methods that generate or collect energy from the environment of the circuit. There are many sources to generate energy from the environment: solar, vibrations, acoustic noise, temperature gradient, radio, etc. Many studies on energy harvesting exist. An interesting, but not exhaustive, summary is presented in [6]. This study gives a comparison of different power sources and energy capacities for Wireless Sensor Network applications. Considering a system operating with a given lifetime of 1 or 10 years, we compare the mean power provided with a normalized volume of $1cm^3$ from different harvesting sources and energy reservoirs. Table 1.1 from [6] summarizes these results. If we consider realistic solutions that can be used in case of BAN, we have vibrations, temperature and shoe inserts. The power delivered is around few hundreds of μW.

From Table 1.1 we need to add the recently developed RF-harvesting that uses ambient radio as Wi-Fi, a good example can be found in [12]. The power generated depends on the number of ambient Wi-Fi communication and the size of the antenna. In [12], an antenna of dimension $9cm \times 9cm \times 1cm$ provides $18\mu W$ for typical ambient RF and $1.37\text{m}W$ with a dedicated radio energy source. For system with dimension of $1cm \times 1cm \times 1cm$, considering the energy scales to dimensions, the harvester should provide $222\text{n}W$ for typical ambient RF and $17\mu W$ with a dedicated radio energy source.

In this thesis, we do not consider WSN but BAN applications. In this case, the usage of the system is daily-based, thus lifetime target is about a day. Dimensions of the system have also to be kept low to respect integration perspective. Considering these constraints, we set as a constraint that our system must not exceed the mean power budget of $100\mu W$.

In another point of view, if we consider a system working for a day (e.g. a daily use of 16 hours of activity) with this power budget, this gives a daily energy budget of $5.76J$. This is equivalent to the energy of $500\mu Ah$ of a $3.3V$ battery. If we consider an energy density of $300Wh/L$ [13] this makes a battery of $5.3mm^3$. This shows that a BAN system, which can operate at the power budget of an energy harvesting, could be supplied by an equivalent battery as small as the system itself.

1.4 Issues and solutions

One of the issues of BANs is that currently it is not possible to design a system that meets all constraints in term of performance, life-time and integration.

Table 1.1: Comparison of energy scavenging and energy storage methods from [6]. Note that leakage effects are taken into consideration for batteries

	Power Density (μW) 1 Year lifetime	**Power Density (μW) 10 Years lifetime**	**Source of information**
Solar (Outdoors)	15,000 - direct sun 150 - cloudy day	15,000 - direct sun 150 - cloudy day	Commonly Available
Solar (Indoors)	6 - office desk	6 - office desk	Author's Experiment [6]
Vibrations	200	200	Roundy et al 2002 [7]
Acoustic Noise	0.003 @ 75 Db 0.96 @ 100 Db	0.003 @ 75 Db 0.96 @ 100 Db	Theory
Daily Temp. Variation	10	10	Theory
Temperature Gradient	15 @ 10 °C gradient	15 @ 10 °C gradient	Stordeur & Stark 1997 [8]
Shoe Inserts	330	330	Starner 1996 [9] Shenck & Paradiso 2001 [10]
Batteries (non-recharg. Lithium)	45	3.5	Commonly Available
Batteries (rechargeable Lithium)	7	0	Commonly Available
Hydrocarbon fuel (micro heat engine)	333	33	Mehra et. al. 2000 [11]
Fuel Cells (methanol)	280	28	Commonly Available
Nuclear Isotopes (uranium)	$6 \cdot 10^6$	$6 \cdot 10^5$	Commonly Available

- High performance is characterized by both a lot of computation and sensor data. Current inertial sensor data are provided by the MEMs from which the gyroscope consumes a lot of energy.
- Ideally the life-time of the system should reach a daily use of 16h. However, the constant use of gyroscope required for high performance would necessitate by itself the use of a battery of about $0.8cm^3$ with lithium technology.
- The integration allows for example hiding nodes in clothes. This is important for the user since it prevents from any discomfort and provides an inconspicuous system for every day usage. The use of battery of about $1cm^3$ is slightly too big to be properly hidden.

The design of a generic system that could meet the three aforementioned constraints is considered as the *Graal* of BAN systems, and could pave the way to a large and heterogeneous set of applications. The implications are larger than just BAN system since such a system could be combined with the Internet Of Things (IoT) and other cross domain applications [14]. The three constraints are connected by the battery and energy concerns. We can consider two approaches, increasing the battery/energy integration so it can be used

with clothes. But this leads to health concerns and acceptance issue if we consider that users literally carry significant batteries on their skins. The other approach is to reduce the overall energy consumption of the system. This approach has the advantage to keep low energy level. Moreover, if the energy considered is low enough the system can be autonomous with an energy harvesting solution. If we summarize, the current challenges in terms of power requirements are:

- The MEMs sensor gyroscope. Its energy consumption out-classes our energy budget of $100\mu W$.
- Radio communication. Standard wifi/bluetooth radio transceivers and protocols are not adapted to on body applications.

The gap between the current radio performance and the expectations for our application is slowly reducing with time. But, the MEMs gyroscope limitation tends to stick over time, so the only solution is to workaround. We can take advantage of:

- The relative low cost of compute operations.
- All nodes are not always moving and not at the same time.

The idea is that locally processing data does not increase significantly the energy consumption of the system. Moreover, from the application point of view, all data are not required, only new data are useful.

These considerations on the energy constraints challenges and the possible way to reduce energy leads to the following statements:

- Node activity has to be driven by the motion. On a daily use basis, the majority of the time nodes are motionless.
- The use of each sensor has to be based on application and motion. We do not use gyroscope if this is not necessary to provide good results.
- The radio communication has to be reduced to the minimum. Do not communicate when there are no new data. Moreover we can perform data fusion before sending if the amount of data is reduced, i.e., local computing means data compression.
- The hardware implementation has to be optimized. Even if computation consumes few energy, it is important to ensure that the hardware itself does not exceed the power budget. So the architecture has to be dedicated to implement target algorithms.
- The algorithms have to be customized to fulfill multiple application modes. If possible, we use methods that allow reusing hardware.

In this study, we focus on the architecture and algorithm parts of the node. We do not consider the communication strategy but only the number of data to be sent for each case. The philosophy of our approach is to design a hardware with energy proportional

consumption. This allows different operating mode corresponding to each application case. The absolute goal should be to reach a daily lifetime (16 hours) for the three application cases. We already know that the motion capture application cases goal can not be reached, instead we will estimate the life-time of our system.

1.5 Contributions and Document Organization

The manuscript is composed of four chapters and a conclusion organized as follows. Chapter 2 presents, in the first section, the results of a preliminary study on the energy consumption of current technologies. We give the state of the art of the energy consumption of usual inertial MEMs sensors, several radio transceivers and several computing operators. For radio transceiver and computing operators we give results of usual solution and results of high performance. The energy is compared to the fixed power budget of $100\mu W$. In the second section we present the working environment, which is a simulator developed in Matlab. This simulator is used to test and compare results of algorithms.

Chapter 3 presents a study of the algorithms used for the applications considered. The main algorithm is an Extended Kalman Filter (EKF), associated to classification and localization algorithms. After a state of the art of the usual algorithms of the domain, we present our methods to answer each application case. Then, we present the selected algorithms and the results for different cases.

Chapter 4 presents the implementation method and results of the EKF. Since the EKF is, by far, the main power consumer of the computing unit, we focus on this part. The architecture is designed using High Level Synthesis (HLS) tools. In this chapter we present the design flow strategy used to implement the EKF algorithm with consideration from the application level down to technology level. Results are given at gate level, which gives a decent approximation of the area and energy cost of the EKF.

Chapter 5 aims to complete the study by adding a system-level management consideration. Previous chapters mainly focus on the results of the EKF, here we consider the whole node energy budget. In the first section we define several operating modes of the node based on sensor sampling rates and EKF computing rate. Then in the second section we design an activity detection module that will drive the operating mode of the node according to the motion. The goal is to show the necessity to manage the global energy of the node and the efficiency of our simple solution to encourage the development of this process.

The final chapter 6 concludes with a synthesis of the contributions of each chapter. We also put into perspective the results considering the BoWI project and possible future works. We can note the importance of the use of motion activity which exceeds the context of the BoWI project since any system with an Inertial Measurement Unit (IMU) is concerned.

2

Preliminary Study and Realization of a Test and Simulation Environment

2.1 Energy profile of technology

Considering our power budget, we can make a preliminary study of the system. The power budget of the system is composed of three contributions: computation, communication and sensors. In this section we summarize the estimated power consumption of current state of the art.

2.1.1 Sensors

Considering applications are related to body postures and motion, we use inertial sensors integrated in an Inertial Motion Unit (IMU). Micro-Electro-Mechanical System (MEMS) sensors have to be used since it is the only way to be integrated into a circuit. The three inertial sensors commonly used are accelerometer, magnetometer and gyroscope. Each of these sensors provides important information but current technological limitations do not allow the continuous use of them with our power budget. Table 2.1 shows the actual state of the art power consumption of these sensors. We give the typical power consumption of the sensor and the equivalent fraction relative to the power budget of $100\mu W$.

Table 2.1: Power consumption of MEMs sensors.

	Power	**Budget part(%)**	**References**
Accelerometer	$15\mu W$	15%	ST MIS2DH
Magnetometer	$80\mu W$	80%	ST LSM303AH
Gyroscope 2 axis	$9.5mW$	9500%	ST L2G2IS
Gyroscope 3 axis	$15mW$	15000%	ST 3G4250D

From a quick look we can already note that:

- Accelerometer is the least power expensive sensor. It can be used continuously since the power budget fraction is reasonable. This particularity allows us to use it as a motion

detection sensor in addition to its normal purpose. With appropriate processing, this sensor can be used as a wake-up for the rest of the system.

- Magnetometer power budget fraction is high. By default this sensor has to be switched off, and powered only when activity is detected.
- Gyroscope exceeds power budget by far. The issue is that this sensor is absolutely necessary in high performance applications such as motion capture. The use will be restricted to strictly needed cases.

2.1.2 Radio transceiver

To be accurate, the power budget of the communication part of the system should take into account a lot of contributions. We can mention the operating conditions such as the environment, the reflection, devices locations,etc. We also have the communication strategy and the topology of the network. The communication protocol, the Medium Access Control (MAC) layer, the physical layer (PHY) and the antennas implementation impact the energy profile of the communication.

However, the global communication strategy is not the purpose of this section. Here, we focus on the radio transceivers performance. The relevant information we are looking for is the number of data that can be sent in a period of time with our power budget considering different transceivers solutions.

Table 2.2 gives an overview of the energy budget of different radio transceivers. The first transceiver considered is ZigBee, which is a usual low power radio. Zyggie [15] is a BAN prototype designed for the BoWI project, whose radio is based on 802.15.4 with an ATmega256RFR2 Microcontroller. The last transceiver is an Impulse Radio Ultra Wide Band (IR-UWB) radio, which is an example of ultra energy efficient radio. In this table we give the energy cost for $1bit$ transmission. We also give an estimation of the number of bits that can be sent using each transceiver with the considered budget of $100\mu W$. This equivalence of bits is given considering we send data at the frame rate of 50Hz, which is a realistic but still low value for motion capture system.

Table 2.2: Power consumption of radio communication transceiver.

	Energy per bit	**Maximum frame size Relatively to budget**	**References**
Zygbee (TX)	50nJ	40$bits$ @ 50Hz	[16]
Zyggie (TX)	40nJ	50$bits$ @ 50Hz	measures [15]
IR-UWB (TX)	13pJ	154k$bits$ @ 50Hz	[17]

Note that the bandwidth has to be shared among all nodes of the BAN. Note also that the cost value is given in the most favorable conditions. First it considers a high data rate of few Gb/s, which is far from a typical BAN requirements. Secondly, it considers only the physical layer and not the protocol that significantly impact the power budget in practice. But it still gives a good overview of possible future projections.

2.1.3 Computation

Computation term includes all the intelligence embedded in the system. We can separate the computation into several parts:

- System-level control that manages sensors, algorithm, radio, etc.
- Computations for activity detection.
- Computations for algorithms related to application.

The most important part of the computation is due to the algorithm part. Table 2.3 collects some energy cost of computation from different implementation solutions. In the first row we present a typical solution using a microcontroller Cortex M3 [18]. The second and third rows also give the energy cost, estimated with Design Compiler and PrimeTime, for a 65nm and 28nm Application Specific Integrated Circuit (ASIC) implementation of the same operation. The energy cost is given for a Multiply and Accumulate (MAC) operation using a fixed-point data type of $32bits$. We also give the equivalent number of operations that can be performed with our budget of $100\mu W$. These values are given considering we perform the same computation at the rate of 50Hz.

Table 2.3: Power consumption of computation.

	Energy per operation	**Maximum operations Relatively to budget**
32-bit MAC with Cortex M3	$0.1 - 1\text{n}J$	$2 - 20k_op$ @ 50Hz
65nm 32-bit MAC	$34\text{p}J$	$58k_op$ @ 50Hz
28nm 32-bit MAC	$5.9\text{p}J$	$340k_op$ @ 50Hz

We can see that the choice of the implementation solution impact the number of operations allowed. Current solutions employed such as Cortex M3 are far from the efficiency that can be achieved using advanced technologies. These values also show that a relative high number of computations can be performed with the available power budget. This means we have room for local computation and so data compression. In the next section we study the comparison between radio and computation.

2.1.4 Comparison between Radio and Computation

In this section we give an overview of the tradeoff between radio and computation. We saw in previous section that the energy cost of an operation is way lower than the energy to transmit a bit. The idea is to add computation to reduce the amount of data to transmit. The first parameter to estimate is the limit from which it is equivalent to send all data or to compute to reduce this amount. We consider for this estimation that the data to be sent are the MEMs sensor values. These data are coded in 9 values of 15 *bits* resulting in a frame of 135 *bits* to send all data. In this study we consider that our algorithm is able to reduce

data to be sent by 50% with one iteration equivalent to 1000 MAC operations. These values are not necessary exact, it is a rough estimate to get a first overview of the tradeoff between radio and computation.

Figure 2.1 presents the possible configurations as function of the cost of one bit to be transmitted and the cost of one iteration of data fusion. The blue dashed line represents the limit where it is equivalent to sent all data by radio or to compute a data fusion and send the resulting data. Under this line, the system has no energy advantage to make any computation and it is therefore better to send all data directly through radio. Above this line, the data fusion achieved using local computation provides an energy gain. We reported values of technology on this figure for radio and computation considering 1000 MAC operations for the iterations. Radio values are those presented in Table 2.2. Computing values are those presented in Table 2.3 for a Cortex M3 processor and dedicated processing in 65nm and 28nm technologies. We added a red dashed line that represents the budget of $100\mu W$ for a frame rate of 50Hz. This line considers that the budget is only dedicated to radio and computation. However, in real scenario, sensors power consumption needs to be considered. Therefore, we also added purple and black dashed lines that represent 50% and 10% of the $100\mu W$ budget at 50Hz respectively.

We can see that, in this configuration Zigbee and Zyggie radio platform consume too much energy to match our objective. We need a more efficient radio transceiver or a better data fusion compression rate. We also see that for the IR-UWB radio [17], the data fusion would not be efficient at all in these conditions. But the system could easily meet our power budget if the fusion is made on a central node like a smartphone that has not the same energy constraint. For the computation, all solutions seem to be able to meet the budget but we will see later that it is not the case.

In chapter 3, which provides a study of the algorithms, we will re-conduct this study with more realistic values for our system.

In the next section we present the simulator used to develop custom algorithms considering specific applications constraints.

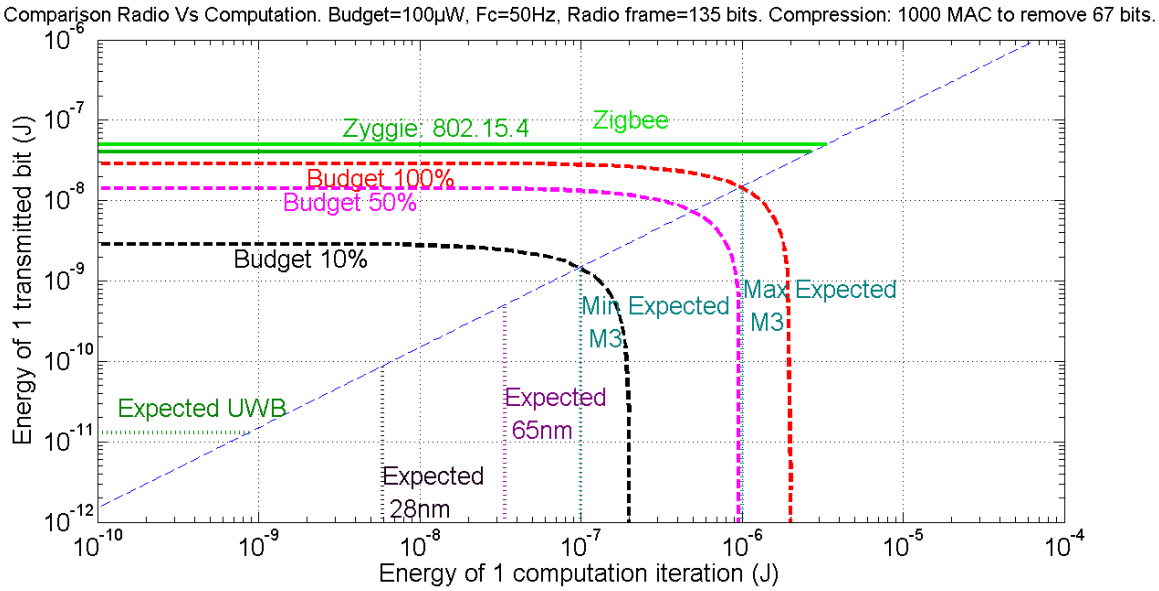


Figure 2.1: Preliminary comparison between Radio and Computation.

2.2 Simulator

In the previous section we saw that algorithms have to be customized. Moreover, network constraints depend on the application considered. The algorithmic exploration has to be performed using scenario-based simulation. Algorithm performance can be compared as a function of parameters such as number of nodes, location on body, noise model, topology, scenario. An environment that allows several scenario-based simulations is therefore required.

2.2.1 State of the art simulators

WBSN simulators already exist. We can cite IMUSim [19] and Bodysim [20], which is an add-on of IMUSim. IMUsim uses Biovision Hierarchy (BVH) files to produce IMU data. BVH is a common format of files that is used to store motion capture [21]. A BVH file is composed of two parts. The first part is the description of the avatar by hierarchy of linked segments that represent limbs. The second part contains angles of joints that link segments for each frame. Each frame is a posture and the succession of frames put together builds the motion. Bodysim uses IMUSim results and add a tool that estimates the radio link strength between nodes of the network. This estimation takes account of distances between nodes and Line-Of-Sight (LOS) or Non-Line-Of-Sight (NLOS) conditions with a solid avatar.

These simulators do not provide all the data and functionality we need. Indeed, in our case a full control on scenarios parameters is requiered. Accessing location and orientation of limb of the avatar allows to directly modify the biomechanical data and then generate sensor data with more consistency. We decided to develop our own simulator environment using Matlab. The use of Matlab offers the two very important following features : the

possibility to repeat and replay simulations to compare solutions with different parameters and to estimate the statistical impact of scenario parameters on results.

2.2.2 Simulation environment

2.2.2.1 Data synthesis scenario based

The first step of the simulation environment is to produce sensor data as if they were sensed from an IMUsim. The purpose of the study is also to evaluate the possibility to use radio link information in algorithms. In our simulation framework, radio link information is also produced.

The process steps to produce sensors data are depicted in Figure 2.2. The simulator uses BVH files that contain motions of an avatar. This capture is usually made by a motion capture system such as Moven [22] or Vicon [23]. With such a capture the position and orientation of each joint of the avatar can be computed for each frame. Depending on the file, the motion is different allowing a scenario-based approach. Sensors are synthesized according to parameter choices such as location, number, and noise, which permits flexible and repeatable simulations. Steps of the sensors data generation are:

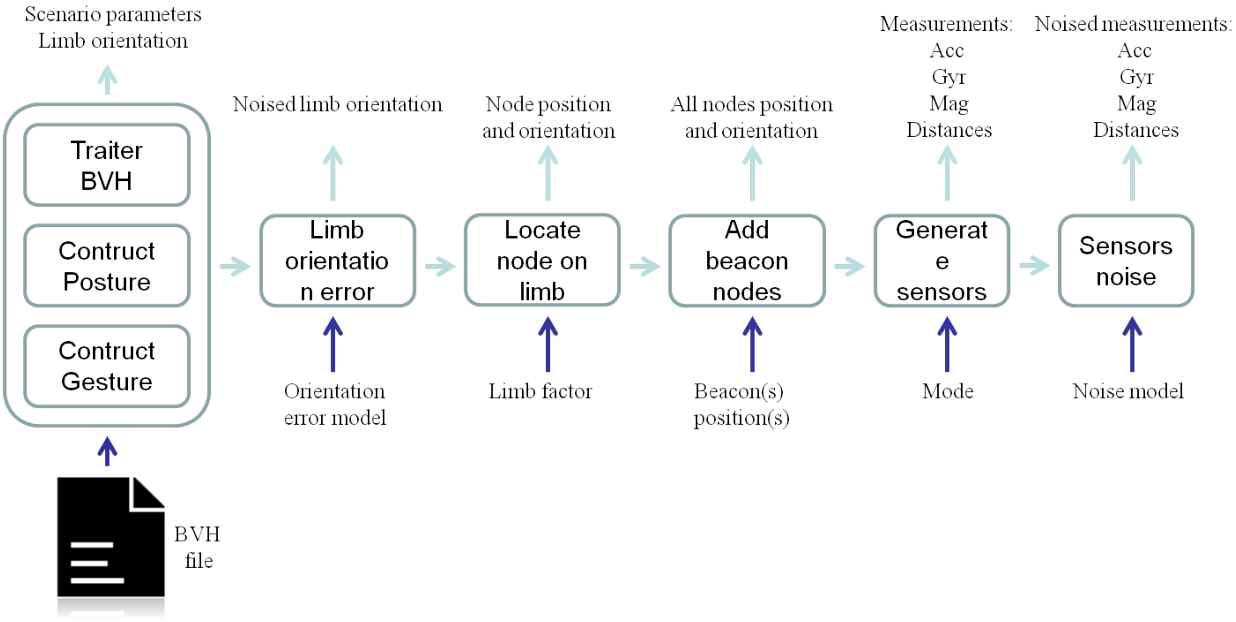


Figure 2.2: Data generation process.

- Extract raw data from BVH file, joint orientations and scenario parameters.
- Add a limb orientation error that reflects the user error from original scenario.
- Locate node on limb.

- Add position of beacons if any.
- Generate perfect IMU data and distances from orientation and position of nodes.
- Add noise to IMU sensor data.

2.2.2.2 Node position and orientation

BVH files provide direct orientations and location of joints of an avatar. Then, we also need to consider the location of the node on the limb and that the user does not always exactly reproduce the same motion. This is the reason why we added two steps to the process. The first add an error on orientation of limb. This error can be customized from the generic Additive White Gaussian Noise (AWGN) model to any specific error model for each limb. The location issue is addressed using a factor, which value is between 0 and 1 that represents the fraction of the limb length.

Another issue needed to be considered is the potential use of beacons. Beacons are static nodes that are located in the environment. Beacons do not provide IMU data but radio link information. Some scenarios consider the use of beacons since it is a useful reference of ground to locate the user in his environment.

2.2.2.3 Sensors data

IMU data synthesis

Inertial data are the easiest data to produce. In a first step, theoretical perfect measurements are produced. The second step applies a measurement noise to each sensor. Here only errors from the acquisition system are considered since errors due to user are previously handled. With node position and orientation at each frame of the motion, it is then easy to compute inertial data from IMU. Accelerometer measurements are computed as the addition of gravity field and linear acceleration of the node. Magnetometer measurements are computed using earth magnetic field value. Gyroscope measurements are computed using quaternion representation of orientation to ease the translation from earth frame to body frame. The earth gravity and magnetic field can be tuned to adapt values to the earth location of the system.

Each sensor can have its own error model. Considering constructor data-sheet of sensors and after measurements, we can determine the inertial sensors error and noise model. Due to error inherent to manufacture process accelerometer and magnetometer MEMS sensors must be calibrated. These sensor raw data suffer from offset and scale error. These errors can easily be corrected by a calibration process but need to be done regularly. In this study we do not consider calibration errors. This issue is well addressed in literature and a lot of solutions are already proposed [24] [25]. Moreover, we can also consider that the minor remaining calibration error is included in acquisition noise.

Figure 2.3 gives an example of the measurement distribution of an acquisition of IMU (accelerometer, magnetometer and gyroscope) data for a motionless node. Each picture corresponds to an axis of a sensor. A Gaussian interpolation that matches the distribution

of measurements is added for each picture. We can conclude that a reasonable error model of MEMS sensors is a AWGN. A possible complement could be the addition of a calibration

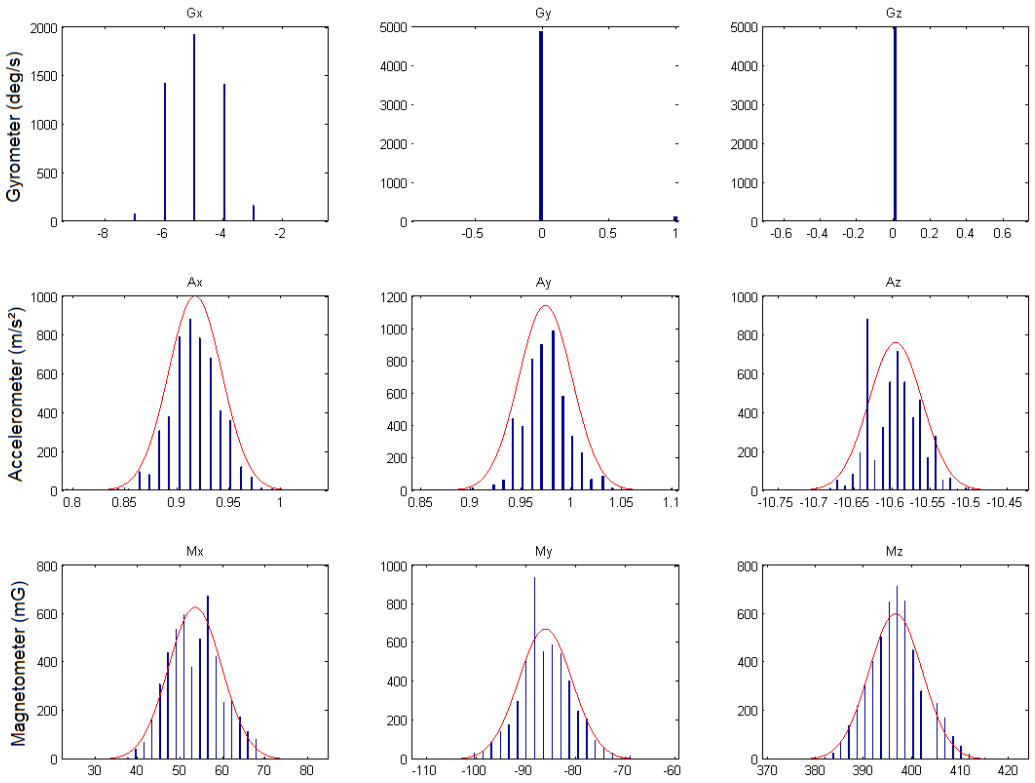


Figure 2.3: IMU aquisition of 5000 samples of a static node. First row represents the three gyroscope axes. Second row represents the three accelerometer axes. Third row represents the three magnetometer axes. Abscissa is the value of the sensor, ordinate is the number of occurrence of this value.

error model. A static magnetic field error could also be added to consider case when and where environment contains strong perturbation.

Distances data synthesis

The most tricky part of the simulator is the radio data generation. However, at the same time, the radio part could be the most important added value of the simulator. Distances between each nodes are known. With an avatar modeling, LOS and NLOS cases can be determined. Radio measurements can be used for several applications. The most common application is the measurement of distances or to determine the location.

- The power of received signal can be a way to estimate distance between two nodes considering attenuation caused by the transmission path [26]. The usual measurement is called Received Signal Strength Indication (RSSI).

- The Time-of-Arrival (ToA) or Time-of-Flight (ToF) can be used to compute distance since the velocity of light and duration of travel can be computed [27].

- The Time-Difference-Of-Arrival (TDOA) measurement coupled with a multilateration algorithm can be used to compute the relative location of nodes and/or beacons [28], [29].

Note that in less complex way the RSSI can also give an estimation of LOS or NLOS cases which can be interpreted as an information on relative location of nodes.

A common expression that links Received Signal Strength Indication (RSSI) and distances in case of LOS condition is given by Equation (2.1):

$$P_{dB} = P_{0dB} + n10\log\left(d_0/d\right) \tag{2.1}$$

with n the path loss exponent and P_{0dB} the reference power at reference distance d_0. However, Equation (2.1) is not really usable as such in BAN case. Shadowing effect of body and environment multipathing are effects among others that can largely alter the radio measurements. At this point we can consider several approaches.

- Consider all point-to-point node communication as generic radio link. According to measurement methods, an error model is applied to distances with a distinction between LOS and NLOS.

- Consider the radio measurement as scenario-based. For each scenario, a radio error model is evaluated by real measurements or simulation using tool such as Computer Simulation Technology [30].

The first solution is the easiest way to implement distance measurement. However, based on real measurements using the BoWI platform, we can state that results largely differ from reality. The second solution has the advantage to be more accurate but needs additional work and can be very time consuming.

An alternative solution would be to use results from some previously published measurement campaign. [31] and [32] have performed RSSI measurements for few node link configurations considering walking, running in several environment conditions. The idea is to collect different typical node link models that depend on location on body, environment and typical motion of user. Then, according to the scenario, to simulate the corresponding pre-estimated model could be selected to generate radio data.

2.2.3 Integration of Algorithm in simulator

The developed simulator is similar to a toolbox. Several functions are used to generate sensors data and scenario parameters. These data are collected in structures, which are used by the algorithms to compute scenario results. Additional scripts have been written to help simulate typical application cases such as posture recognition or gesture recognition. This simulator has a statistical approach, which means some scripts produce data in files and

other scripts process results for interpretation. The user chooses which parameters to fix and which to monitor. In case of motion capture, methods also exist to visualize the results of algorithm

2.2.4 Algorithm complexity monitoring

An interesting feature added in our simulator is the possibility to estimate the computing load of an algorithm. This computing load is estimated at runtime, which gives accurate results on the real complexity involved. The idea is to include counters in critical point of algorithm as loops. In this case the user has to estimate only core loop operations and to store it into a Comma-Separated Values (CSV) file. The simulator automatically sums up all operations for each nodes. With the add of an energetic model for each operation, it is then possible to estimate computing energy consumed at each frame. The advantage of this method is that granularity of this estimation can be controlled by the user. Figure 2.4 is a capture of the interface that summarizes the computing load of a given simulation, it is composed of 6 panels. During a run, the number of calls of the core loop that the user wants to monitor is incremented and stored in a file that represents the current scenario. These values are in the *Scenario Profile Panel* of the interface. Each core loop is associated with a file that contains its computation load in terms arithmetic operations (e.g. addition, multiplication). This is represented in the *Algorithm Profile Panel*. So, we can summarize the total number of operations of each node at each frame and for the whole run as described in the *Operation Cost Panel*. Then, with an energy model of each of these operations, as in the *Energy Model Panel*, the energy consumption at each frame can be estimated. We can see in the upper right corner the energy consumption of the node number 13, which is different depending on the mode of computation at each frame. A summary of the energy and then the global power consumption for each node is given in the *Node Energy Budget Panel*.

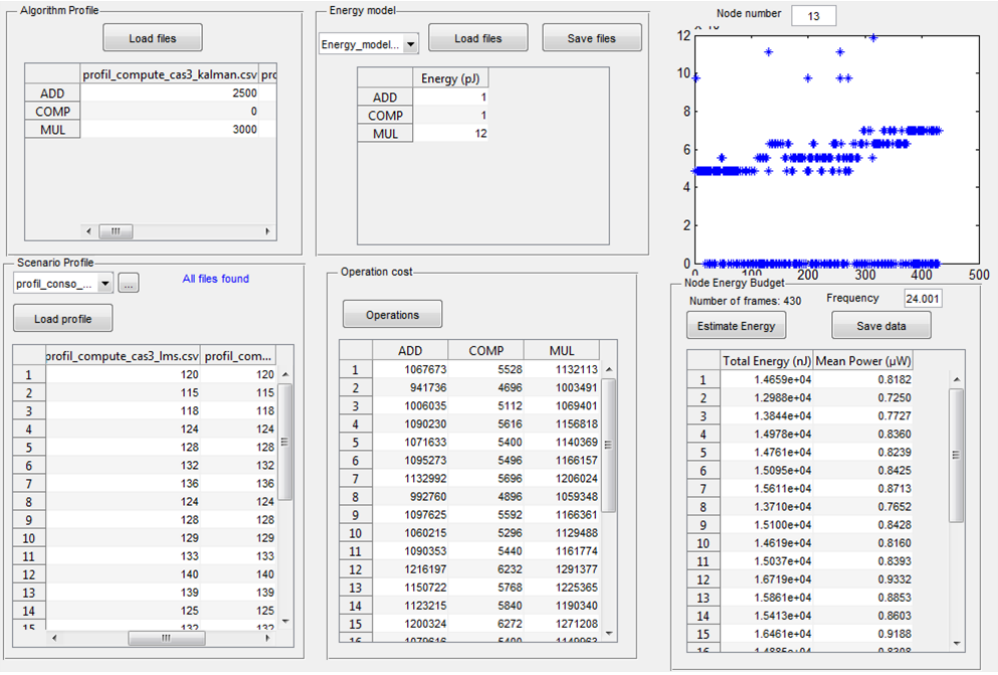


Figure 2.4: Computing load complexity monitoring interface.

2.3 Conclusion

In this Section we presented the different component of the energy budget of our system: sensor, radio and computation. We showed that the power consumption of state of the art MEMs sensor is to high for a constant use of them with our power budget objective of $100\mu W$. A smart usage of sensors corresponding to real-time condition and to application needs is necessary to achieve our goal. The study of energy consumption of different radio solutions showed that usual solution using Zigbee transceiver and even custom made solution such as Zyggie, can not be used for our power budget objective. Other solution like IR-UWB have to be used. The study of energy consumption of different computation solutions showed that it is possible to use a classic solution like Cortex M3 microcontroller since we do not exceed a relative low number of operations around several thousand. The use of ASIC with 65nm or 28nm technology would allow 10 to 100 times more operations. Also, the balance between the computation and radio consumptions must be considered to ensure the viability of the system.

The design of the algorithms and the testing have to be performed considering specific applications and scenarios. The developed simulator uses motion capture files (e.g. BVH) which allows scenario-based simulation and provides a full control on the sensor data produced and the scenario parameters.

The next Section explores the algorithms to be implemented for the three application cases considered. The exploration is supported by simulation results and constantly takes account of the necessity to minimize the power consumption of the system in all aspects (e.g.

sensor usage, radio usage, computation).

3

Posture/Gesture Recognition and Motion Capture Algorithms for Low Power Implementation

In this section we explore and propose some algorithms that compel application with requirements and constraints limitations.

The three application functionalities to implement are:

- posture recognition (static),
- gesture recognition (sequence of postures),
- motion capture (full record).

An important point is that algorithms have to be implemented in hardware for efficiency reasons previously discussed. This leads to some preferences or simplification in the algorithm choices.

The point is that posture and gesture recognition algorithms do not necessitate estimating the real stance of the user. The idea is to estimate the most likely posture/gesture among a selection of possible gestures that belong to an application specific set.

However, the motion capture necessitates the use of biomechanical models which can be used to visualize the resulting capture. The motion capture has a second usage in our case, which is the possibility for the user to capture the posture and gesture for a custom use of the system.

3.1 State of the Art Algorithms

Application targets necessitate the use of several algorithms.

The first category of algorithms is the Attitude and Heading Reference System (AHRS). These algorithms estimate the orientation of the sensors (and so the device) regarding the local cardinal direction of the ground. This local frame acts like a common frame reference for all nodes of the system.

The second category of algorithms is the localization. The use of distance measurements extracted from radio communication allows estimating location of nodes. This location can be used mainly for motion capture purpose.

The third category of algorithms is the classification for recognition purpose.

The kind of sensors available on a BAN is limited. We can use an Inertial Measurement Unit (IMU), which is composed of a 3-axis accelerometer, a 3-axis magnetometer and a 3-axis gyroscope. We also use data from radio to provide distance measurements. The application case of posture/gesture recognition can be presented by the Figure 3.1. The IMU provides inertial data that is processed to estimate the orientation. The radio provides distances measurement, which combined with orientation, are used to recognize the posture/gesture through a classification method.

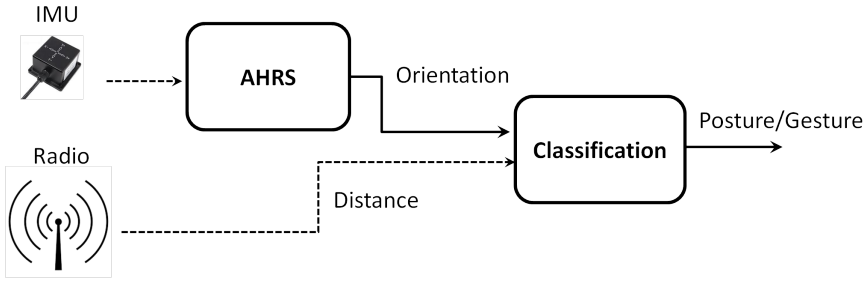


Figure 3.1: Synoptic view of a Posture/Gesture recognition application.

The motion capture is a little different, see Figure 3.2. It uses the location of nodes, the biomechanical model and the orientation to produce an accurate record of motion.

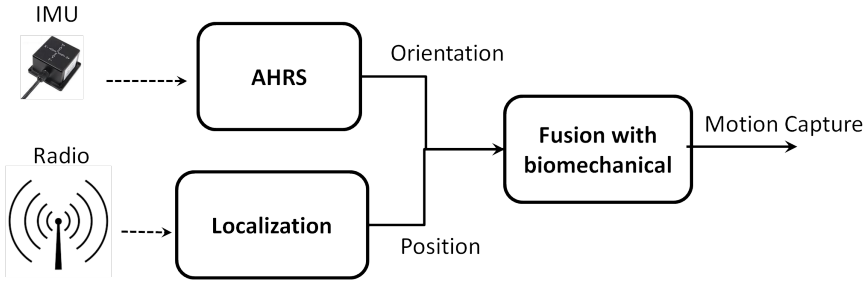


Figure 3.2: Synoptic view of a Motion Capture application.

3.1.1 Attitude and Heading Reference System

In dynamic conditions, which is during a motion, real time orientation is needed. A tracking of the orientation has to be done using appropriate algorithms that take advantage of gyroscope measurement. The point is that, in the dynamic case, the accelerometer does not only measure gravity but also linear acceleration. the magnetometer is not affected but is not always enough to compensate for this error. Moreover, magnetometer can be disturbed by local magnetic field caused by metallic object in the environment. For these reasons, we have to use a sophisticated algorithm to estimate the orientation during a motion.

A lot of algorithm variations have been developed in the literature to perform inertial data fusion. The most common algorithm used for inertial data fusion is the Extended Kalman Filter (EKF) [33] [34] [35]. We can also mention the complementary filter [36], and a gradient descent algorithm [37]. However, they are all based on the same principles. Gyroscope helps to track orientation during motion while accelerometer and magnetometer correct orientation during static phase. A good overview and comparison of these methods can be found in [38].

Our EKF implementation is based on [39] that is gyro-free. It means that the filter can work without data from gyroscope, which is a key factor of our solution. Considering the power consumption of a gyroscope, which is two orders of magnitude larger than the accelerometer, it is important to make a sparse use of it. Therefore we implement a version of the filter that can avoid the use of the gyroscope. We will describe our EKF algorithm in details in section 3.3.2.

3.1.2 Localization

Localization is a global concept that includes:

- Navigation by satellite known as Global Positioning System (GPS) or Galileo.
- Navigation by inertial sensors.
- Global System for Mobile communications (GSM) localization for cell phone.
- Indoor positioning system using radio of acoustic waves.

In our case the localization method is limited by the available sensors.
Each node potentially has the available measurements:

- Its own inertial measurements with IMU.
- Radio power measurements by radio link between other nodes.
- Distance measurements by radio (e.g. UWB) or other method (acoustic) with other nodes.

Our localization method can target two functionalities

- Globally locate the WBSN, and so user, in the environment.
- Locate each individual node relatively to other nodes.

The first functionality can be used for indoor or outdoor localizations. We can also think to several WBSN systems that estimate location from each other. The second functionality is commonly used in large-scale Wireless Sensor Networks (WSN) to create a location map of the network. In our case of body-scale application, the node location could be used to improve or replace motion capture systems.

3.1.2.1 Step detection algorithm

The location of the user in the environment can be a big deal depending on the system used. If the system is a standalone device placed on the body of the user, location can not be directly estimated with only intra-network radio communications and inertial measurements, since the system has no external reference to estimate the ground frame. However, some algorithms have been developed to estimate the displacement of the user from his initial location. These algorithms are usually called step detection.

Several elements can be used to estimate the steps of the user which leads to two main methods:

- The frequency analysis of inertial measurements informs on the user motion. With a relatively complex method, the system estimates if the user is walking and the step size. This solution has proven its value in case of indoor localization with the use of a single smartphone. The accuracy is enough to track a user path in a building [40] [41].
- The estimation of the user posture allows the system to estimate which foot is touching the ground. Then, considering the foot on the ground is static, the remaining body moves according to this static point [42]. This method necessitates the use of a biomechanical model and the estimation of inferior limb orientations (or other limbs for unusual cases). It can be improved with the detection of discontinuities in values of inertial sensors, which occur with the contact with external environment [43].

The first solution, with frequency analysis, is generally used for indoor localization as a complement of outdoor detection with GPS, which is the reason why the use of a smartphone is privileged. This solution requires relative complex algorithm, suitable for a smartphone, but not for a low-power hardware implementation. The second solution is used by inertial-based motion capture systems. The ground detection is required to provide accurate motion capture, which is not our case. Our motion capture intend to be used as a record for the gesture recognition, which do not require the absolute location of the user.

3.1.2.2 Network localization algorithm

Depending on the scale, the same algorithm can be applied to locate only the user or each node separately. In [44] the authors gives a method for network localization using distance estimation by radio. This method works as follow:

- Each node measures radio power received from other nodes.
- Each node computes its estimated location considering locations of other nodes and radio measurements.
- Each node updates its location to all other nodes.
- This algorithm is iterated until the network positioning stabilizes.

Radio links used are usually of poor quality. However, this algorithm is efficient since the high number of nodes gives enough redundancy to obtain good results. The estimation of location is done using a global cost function to minimize. This cost function put in parallel the radio power and theoretical distances from estimated positions. This quantity must be minimized when the locations matches the real configuration. In fact, this global cost function is split for each node to its own local cost function. The main difficulty of this algorithm is the way to minimize cost function.

Let consider a generic system of n on-body nodes and m beacons dispatched in the environment. Let x_i denote the estimated coordinates of node i.

$$x_i \quad = \quad \{x_i, y_i, z_i\}. \tag{3.1}$$

Let X denote the coordinate matrix of all nodes of the system

$$X = \left| \begin{array}{l} \left.\begin{array}{l} x_1 \\ x_2 \\ . \\ x_n \end{array}\right\} n \\ \left.\begin{array}{l} x_{n+1} \\ . \\ x_m \end{array}\right\} m \end{array} \right. \tag{3.2}$$

δ_{ij} is defined as the measurement of the distance between node i and all nodes j seen by sensor i. ω_{ij} is the weight accorded to the corresponding measurement which reflects the certitude of the measure. Finaly,$d_{ij}(X)$ is the Euclidean distance between nodes i and j computed from estimated coordinates.

$$d_{ij}(X) = \sqrt{\|x_i - x_j\|}. \tag{3.3}$$

Prior coordinate of some non-beacon node noted $\hat{x}_i$ is supposed to be known with a certitude noted r_i.

[45] formulates the global cost function, called STRESS function, as follow:

$$S \quad = \quad 2 \sum_{1 \leq i \leq n} \sum_{1 \leq j \leq n+m} \sum_{1 \leq t \leq K} \omega_{ij}^{(t)} (\delta_{ij}^{(t)} - d_{ij}(X))^2 + \sum_{1 \leq i \leq n} r_i ||x_i - \hat{x}_i||^2 \tag{3.4}$$

Equation 3.4 is generic since it considers beacon, multiple measurements and possible prior knowledge on node coordinates.

This function can be decomposed in a local cost function S_i as

$$S = \sum_{i=1}^{n} S_i + c \tag{3.5}$$

with

$$S_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \overline{\omega}_{ij}(\overline{\delta}_{ij} - d_{ij}(X))^2 + \sum_{j=n+1}^{n+m} 2\overline{\omega}_{ij}(\overline{\delta}_{ij} - d_{ij}(X))^2 + r_i||x_i - \hat{x}_i||^2 \tag{3.6}$$

where c is a constant indépendant of the nodes locations X. Here $\overline{\omega}_{ij}$ and $\overline{\delta}_{ij}$ are weights and distances that summarize the K measurements in a single weighted mean value.

This formulation shows that each local cost function S_i depends only on measurements of node i. Moreover, if there is no beacon ($m = 0$) and no prior estimation of location of all node ($\forall i, r_i = 0$), then $\frac{\partial S}{\partial x_i} = 2\frac{\partial S_i}{\partial x_i}$. This reveals that each S_i can be minimized locally on node i with no negative repercussion on the global cost function S.

[44] proposes the use of a gradient method to minimize the cost function which is an efficient method. However, there are two drawbacks to the proposed method:

- The euclidean distance $d_{ij}(X)$ needs the use of a square root computation.
- The gradient method uses several divisions.

Moreover there is a main difference between the case study of [44] and our case: the on-body node's location is dynamic. During a motion, we do not necessary have the time and the energy to make several measurements.

Then, we propose an algorithm based on the same principle with a slightly different cost function and another minimizing method. Moreover we take advantage of inertial measurements to predict node location. This will be described in section 3.3.3.

3.1.3 Classification

Classification is a method that allows to organize a set of vectors into classes. If a new vector is given, the classification method should allow to determine if it belongs or not to one of these classes. A lot of methods exist and each problematic can be addressed differently. We can note three main categories of problems:

- The simplest case is with invariant data. In this case, the Bayesian classifiers [46] are a good solution. Bayesian classifiers use a probabilistic approach for potentially several parameters. These methods are efficient if the data of the same class are strongly correlated. In some cases, we can even consider simplified version. We can use the

center of each class to determine the closest class of a new data. A norm adapted to the problem must be chosen to compute distances. This kind of trivial method is called Nearest Centroid Classifier (NCC).

- If data have more variability or the class center does not make sense for the distribution, the belonging to a class is not determined by a distance but by the relative position from space delimiter. This is the case of Support Vector Machine (SVM) [47].

- Finally, if the data set is too complex to be handled by classical solutions, the usual universal answer to this kind of problem is the machine learning. The representative candidate of machine learning are the neural network [48] and more recently the deep learning [49] [50]. Machine learning is a method, which offers good results, but the solution is complex and does not fit with low power, low cost embedded system especially If we consider a dedicated hardware implementation.

In several cases, there is a lot of invariant but data cannot be used as such for classification. An intermediate data set has to be computed to extract relevant information from original data. This is the purpose of methods like the Principal Component Analysis (PCA) [51], which are also interesting since the classification execution (not the learning) is very simple..

3.2 Application Solutions

3.2.1 Handling Posture and Gesture with Principal Component Analysis

Before we present the solutions for posture and gesture recognitions, we show in this section examples of postures, gestures and application of PCA.

Some measurements of sensors data have been made with the Zyggie prototype. The nodes have been installed on a user as showed in Figure 3.3.

For this example only postures of arms are considered. Figure 3.4 shows the three possible postures for each arm, then by combination we can produce nine global posture. The prototype recorded IMU and RSSI data. Because radio power data are sensitive to the environment, we recorded the set of postures in three different places. A small office that we qualify of a narrow space, a meeting room that we qualify of a closed space and outdoor that we call free space. During the recording and for each posture, the user stay as motionless as possible.

Then, we apply the PCA to the accelerometer values. The PCA is a linear method to extract relevant information of a set of data. This is done by the search of eigenvalues and eigenvectors of a matrix. This matrix is composed of rows that are the classes we want to differentiate (postures in our case) and columns are the parameters to use (sensor data in our case). In our example, we give a matrix to the PCA like if the nine postures of the three

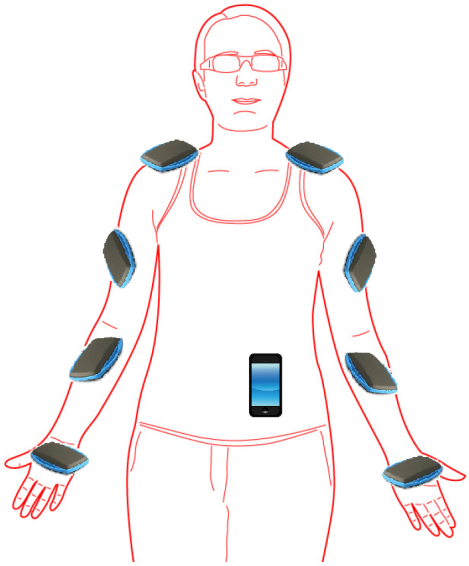


Figure 3.3: Positioning of the nine nodes on the body.

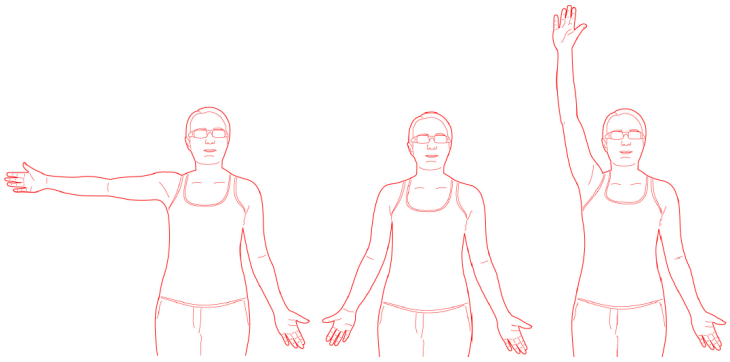


Figure 3.4: The three possible positions for each arm.

environments were all 27 different postures. The idea is to observe in the eigenspace the relative position of each posture.

Figure 3.5 shows the projection using accelerometer data on the eigenspace. We can see on the left that the same posture in the three space are gathered. We can clearly recognize a 3×3 square were each group of three points is the same posture. We can see that only two dimensions are enough to classify the set of nine postures. It is normal since accelerometer data are independent for each arm, then the first two dimensions for the two arms are orthogonal. With such projections, classification can be done with simple method as NCC.

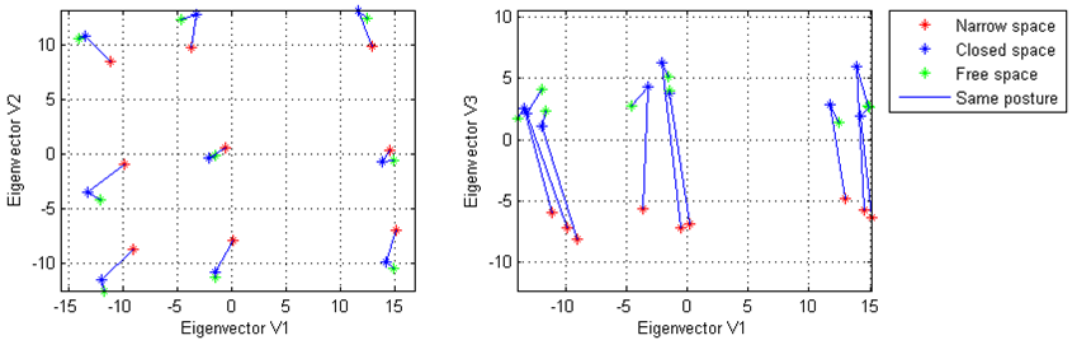


Figure 3.5: Projection of accelerometer data for each of the nine postures and the three kinds of environment. On the left, projection along first and second eigenvalues. On the right, projection along first and third eigenvalues.

We try now the same observation but using RSSI data. Figure 3.6 shows the projection using RSSI data on the eigenspace. Here, we cannot observe the square pattern. It is normal since the power measurements of the two arms are correlated, there is no orthogonality. However, posture are not randomly located, we can see there is a tendency of the posture of the same environment (same color on the figure) to be on the same side. Here, a trivial classification as NCC is not possible, an efficient classification would require a method that delimits space like SVM.

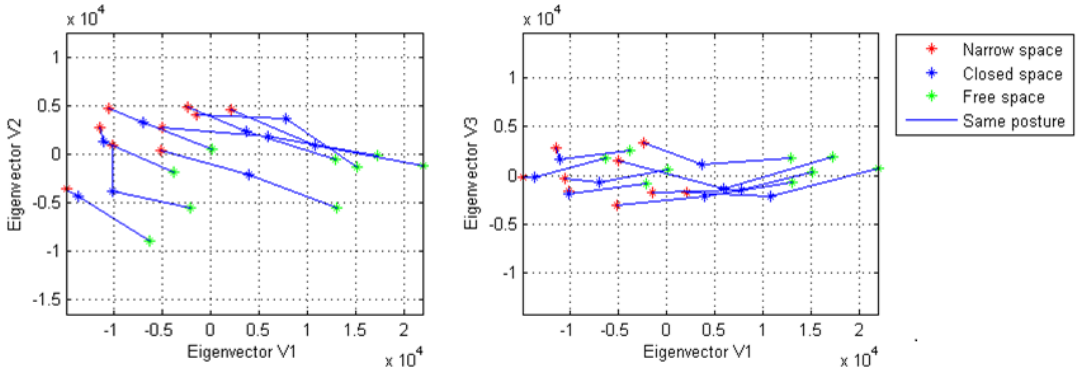


Figure 3.6: Projection of rssi data for each of the nine postures and the three kind of environment. On the left, projection along first and second eigenvalues. On the right, projection along first and thrid eigenvalues.

This example on a set of postures shows that the PCA can be a really efficient method to automatically extract relevant dimensions of the classification. The idea is now to determine which data should be used to produce the projections. This will be the objective of the next section.

In this last example we apply the PCA to a gesture. Using a motion capture file we generate distances data between each pair of nodes. Then, each frame of the motion is considered as different posture. We apply the PCA to this set of postures that composes the gesture using distances computed. Figure 3.7 shows the projection in the eigenspace of each of the 86 postures of the motion in blue. We can see at the bottom of the figure some corresponding postures. We also added green stars that correspond to projections of a noised set of postures. We consider a possible error on each distance measurements caused by a noise with a standard deviation of $10cm$. We see that all projections of possible noised postures are restricted in the eigenspace around the original posture.

This shows another aspect of our approach. Even with noised data, the redundancy due to the number of nodes and data considered robustify the method. So we do not need very accurate data since we have a lot of them that compensate for each others. In a way, it is like each posture has a sensor signature and we have to determine the most likely.

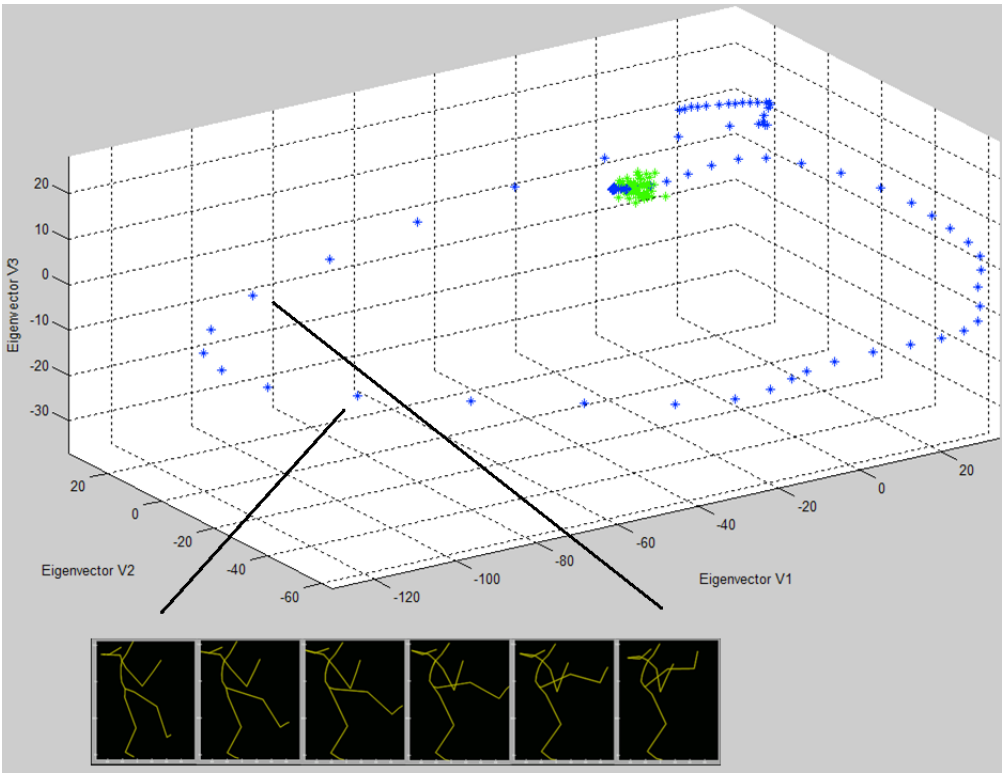


Figure 3.7: Projection of distances data for the Capoeira motion. Projection is along first, second and third eigenvalues. Each blue star is one of the posture that compose the gesture. Green stars are projections of a set of noised data around an initial posture.

3.2.2 Posture Recognition

In this case we have to recognize a posture among an existing set. The solution has to be of low complexity to be implemented and the data used are IMU and potential radio measurements or distances. The posture recognition works as follows:

- A set of postures called library is chosen and recorded.
- Each posture is characterized by a set of values.
- These values can be inertial or radio measurements or other data-fusion results.
- A classification method is chosen to determine the user posture among the library.

If we review the available data we can see that the Accelerometer data is necessary but insufficient. Accelerometer measures gravity acceleration in static case that is used to know the bottom direction of the sensor. However measurements are invariant of the North direction of the node and an orientation direction is missing. Magnetometer data is necessary but insufficient as for the accelerometer. Principle is the same, the vector measured is earth magnetic field, but cannot determine alone the exact direction of the sensor. Gyroscope only measures motion and thus it cannot be used in static case. Radio measurements can be used as posture signature. Indeed, depending on limb configurations, it changes node distances and antenna coupling.

A good candidate to posture description is the orientation of limbs. But, a set of postures must not depend on the North direction. Then, values used must be independent of the north direction of the user. Then, in the case of the use of orientation, it must be normalized by a central node orientation. It remains that candidates to posture descriptions are accelerometer measurements, normalized orientation, distances measurements and radio measurements.

The posture recognition of the system is in four steps:

- Detect when the user is static.
- Take inertial and/or radio measurement.
- Compute orientation estimation and data-fusion.
- Apply classification to determine the current most likely posture among library.

A priori, the classification step has to be performed with all data gathered at the same location. The posture decision is carried out by the master node.

3.2.3 Gesture Recognition

If we consider a gesture as a succession of postures, then a gesture recognition can be performed using the same principle as in previous section. Figure 3.8 shows some intermediate postures that compose the Capoeira kick move.

Figure 3.8: A gesture is composed of successive postures correlated in time.

We adjust the method for posture recognition considering a gesture as a succession of postures. We apply the posture recognition to the set of postures that composes the gesture. The method becomes:

- Record a gesture.
- Split the gesture into a set of postures that compose the library.
- Constantly apply the classification to determine the current most likely posture among the library.
- Add an algorithm to recognize the sequence of postures and confirm the gesture detection.

The case of gesture leads to other constraints. The main difference is due to the motion of user. In this case, we need a tracking of the orientation of node. The orientation algorithm is more elaborated and the frame rate correlated to the motion.

The gesture recognition is performed with the follows steps:

- Detect when the user is moving.
- Sense inertial measurements.
- Compute orientation estimation and eventually data-fusion.
- Apply classification to determine the current most likely posture among library.
- Use the sequence algorithm to determine the gesture.

The steps of this process are illustrated by the Figure 3.9.

3.2.4 Motion capture

The motion capture scenario is really different from previous application cases. The posture and gesture recognition uses IMU and radio data to guess the more likely current posture among a defined and limited set of postures. Even if the method is valid it does not allow to know the exact real-time posture of the user. Here, the motion capture has to provide the orientation of each limb of users and their locations.

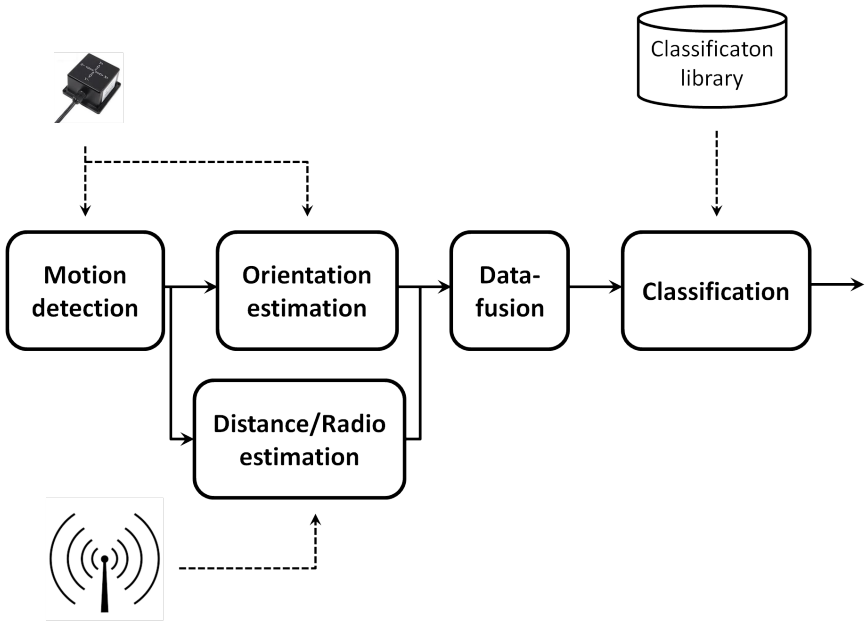


Figure 3.9: Synoptic view of the gesture recognition algorithm.

The standard way to represent the final results of a motion capture is using a biomechanical model. Considering all limbs are connected to joints, by starting from the location of a central joint, which usually is the hip, we can reconstruct the posture. With a fixed biomechanical model, all we need to reconstruct a posture is the orientation of all limbs and the location of the central joint. The succession of postures at a high rate (60-120-250 Hz) composes the motion.

Existing systems use different methods to reach these results. We can consider two categories of methods:

- Methods that use orientation of nodes on the user. This system generally use inertial units. By using a biomechanical model, the system is really close to what a motion capture is. However, the system requires to add the location of the user in a way that allows to follow the user displacement. This can be done by the addition of beacons to locate user in space or by speculation on the user displacement steps. Moreover, particular joints like shoulders or collarbone, can necessitate specific processing to produce high quality motion capture. The case of Moven [22] [52] is an example of such a system.

- The other category of methods uses location in space of markers located on the user. Usually these systems are based on optical solution like infra-red detector and/or camera. These camera-based systems are very accurate, and with enough markers, such a system can also estimate orientation of limbs. These systems also provide lot more information than inertial solutions with the absolute location in space of each limbs and joints. Then, with even more complex and accurate algorithms, the system can provide high quality motion capture. The typical example of this system is vicon [23]

In our case, constraints in energy and free space applications do not allow to produce high quality motion capture. The two main limitations are the local energy, which does not allow to continuously use sensors or either send all data and the final fusion algorithm, which is too complex for our target of simple solution. We propose then to consider the quality needed for the target and to simplify the algorithm so that we can minimize complexity. We use a biomechanical model and orientation of nodes to construct the user posture. Then, the location of the user is computed by a simple step detection algorithm, or another solution would be to put 2 or 3 nodes in the environment.

3.3 Proposed Algorithm Solution

Before explaining the method to compute orientation, we give some conventions on the considered earth frame and the body frame. We note $(\vec{O}, \vec{I}, \vec{J}, \vec{K})$ the earth frame, which is attached to the ground. In our case we consider that the convention of this frame is North, West, Zenith. We note $(\vec{o}, \vec{i}, \vec{j}, \vec{k})$ the frame attached to a node, and call it the body frame. Figure 3.10 shows in red the earth frame and in blue the body frame of an object.

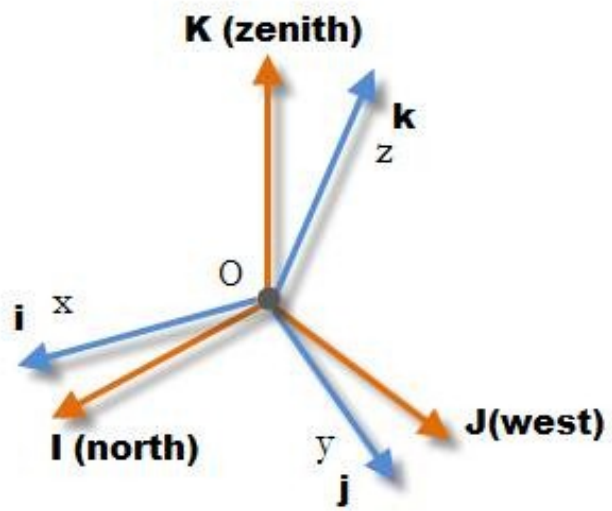


Figure 3.10: Convention of earth frame and body frame of sensors. In red the earth frame with convention North, West, Zenith. In blue the body frame

The earth frame is considered as the absolute frame common to all nodes. To represent the orientation of a node we use the rotation matrix that convert from the body frame to earth frame. This matrix is composed of 3 unitary vectors $(\vec{u}_x, \vec{u}_y, \vec{u}_z)$ that are the unit vectors of the earth frame projected in the body frame.

3.3.1 Static Orientation

In the static case, accelerometer and magnetometer provide a minimum set-up to compute orientation of nodes. There are several methods to compute this orientation. Here, since the posture is static, a computation can be performed with a simple algorithm. Moreover, we consider that for each node the data update is done only if the node is really static, this ensures that data transmitted are reduced to only relevant information and it also allows computing orientation being sure of the input stability.

The principle of the computation is to use environmental information. An accelerometer measures gravity so the down direction and the magnetometer measures earth magnetic field that holds north direction. These two vectors can be used to compute unit vectors of earth frame.

$$\vec{u_z} = -\vec{acc}/||\vec{acc}|| \quad (3.7)$$
$$\vec{U_x} = -\vec{mag} - \vec{u_z}(\vec{u_z} \cdot \vec{mag}) \quad (3.8)$$
$$\vec{u_x} = \vec{U_x}/||\vec{U_x}|| \quad (3.9)$$
$$\vec{u_y} = \vec{u_z} \times \vec{u_x} \quad (3.10)$$

Equations (3.7) to (3.10) show a simple way to compute unit vectors of the earth frame in the static case. $\vec{acc}$ and $\vec{mag}$ are vectors measured by accelerometer and magnetometer and $\vec{u_x}, \vec{u_y}, \vec{u_z}$ are unit vectors of the earth frame. This works as follow: we know that gravity is along the vertical of the earth frame, so the projection of this vector is proportional to the corresponding unit vector, so we normalize this vector to compute $\vec{u_z}$. Then, we know that the magnetic field is on the plan (xOz). So we remove the vertical projection of the magnetic field to only keeps the projection along the north axis and normalize the result. This gives unit vector $\vec{u_x}$. The last unit vector is deducted from the 2 others with a cross product considering a positive oriented frame. These vectors form the rotation matrix and so the orientation of node.

3.3.2 Extended Kalman Filter

The proposed EKF is based on the state-of-the art of Attitude and Heading Reference Systems (AHRS) such as [39]. The EKF uses an Inertial Measurement Unit (IMU) to compute the orientation of a sensor node as a quaternion. Quaternion is a mathematical object that can be used as a representation of orientation [53]. Currently, there are 3 ways [54] to represent an orientation: 1) using the 3 rotation angles, which are called Euler angles; 2) using the rotation matrix; 3) using the quaternion, which is a vector of four values. Quaternion is the less intuitive representation [55] but is more robust in terms of stability since it prevents from the gimbal lock and is less sensitive to rounding off errors.

In EKF equations, we call state vector, noted $X_k = (q_0, q_1, q_2, q_3,)$, the vector that contains the variables that we want to estimate at step k, which here are quaternion values. We call observation vector, noted z, the data that are used as feedback, which here are the

normalized accelerometer and the normalized magnetometer. We will see after why and what we mean by normalized. The EKF is composed of two steps which are the prediction and the update. The prediction is the ability of the system to estimate the state vector at next time step only from its current value. This is represented by the mean of the function f that computes $\hat{X}_{k+1}$ from X_k. The update is the use of feedback data to estimate if observations are in accordance with the predicted state vector $\hat{X}_{k+1}$. This is represented by the mean of the function h that computes the estimated observation vector $\hat{z}$ from the predicted state vector$\hat{X}_{k+1}$. Then with some matrix equations, shown hereafter, the state vector X_{k+1} is updated at step $k+1$. An important point of the EKF is the use of covariance matrix P_k. This matrix is not easy to understand. We just say that, like the state vector X_k, there are equations to predict the next time step value $\hat{P}_{k+1}$ and other equations to compute P_{k+1}. These equations imply the use of Jacobians of f and h, which are $F = \frac{\partial f}{\partial X}$ and $H = \frac{\partial h}{\partial X}$. For a better understanding we can separate equations in several categories. Some equations are specific to EKF computation whatever is the application considered. These equations are presented below, Equations (3.11)-(3.12) and (3.13)-(3.18) correspond to the prediction part and the update part respectively. These equations use f, h, their Jacobians F and H and the identity matrix I_P. Q and R are matrices that are respectively the model noise covariance and the observation covariance. These matrices are important and will be discussed later in this section.

The other equations considered are those specific to the our model of the system. These equations are also presented below, Equations (3.19)-(3.21) are related to the prediction model and Equations (3.22)-(3.23) are related to the update model. M is computed using gyroscope data noted (g_x, g_y, g_z). Rot is computed using the predicted state vector values $\hat{X}_{k+1} = (q_0, q_1, q_2, q_3)$. $\vec{G}$ and $\vec{H}$ are respectively the normalized gravity vector and the normalized earth magnetic field vector. These vectors are pseudo-constants that depend on the location on earth, we will see after why these vectors are normalized. The Jacobian $H = \frac{\partial h}{\partial X}$ can easily be computed by derivative of Rot if we approximate considering $\frac{\partial q_i}{\partial q_j} = 0$ for $i \neq j$.

$$\hat{X}_k = f(X_{k-1}) \tag{3.11}$$
$$\hat{P}_{k-1} = F \cdot P_{k-1} \cdot F^T + Q \tag{3.12}$$

$$\hat{z} = h(\hat{X}_k) \tag{3.13}$$
$$H = \frac{\partial h}{\partial X} \tag{3.14}$$
$$S = H \cdot \hat{P}_k \cdot H^T + R \tag{3.15}$$
$$K = \hat{P}_k \cdot H^T \cdot S^{-1} \tag{3.16}$$
$$X_k = \hat{X}_k + K \cdot (z - \hat{z}) \tag{3.17}$$
$$P_k = (I_P - K \cdot H)\hat{P}_k \tag{3.18}$$

$$f(X_{k-1}) = X_{k-1} + dt \cdot F \cdot X_{k-1} \tag{3.19}$$
$$F = M/2 \tag{3.20}$$

$$M = \begin{pmatrix} 0 & -g_x & -g_y & -g_z \\ g_x & 0 & -g_z & g_y \\ g_y & g_z & 0 & -g_x \\ g_z & -g_y & g_x & 0 \end{pmatrix} \tag{3.21}$$

$$h(\hat{X}_k) = \begin{bmatrix} Rot \cdot \vec{G} \\ Rot \cdot \vec{H} \end{bmatrix} \tag{3.22}$$

$$Rot = \begin{pmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & 2(q_0^2 + q_3^2) - 1 \end{pmatrix} \tag{3.23}$$

The method used to compute Equation (3.16), which could imply a matrix inversion, is the Gauss-Jordan elimination. Strictly speaking it is an equation solving but we call it a matrix inversion to keep generality.

Quaternion is a unitary vector, that is why we need to add a normalization after each modification of these values, which are the prediction and the update. The vector normalization method used is a derived from Goldschmidt iterative reciprocal square root algorithm [56]. This method is really effective since the vector is already close to be unitary, only few iterations are required to achieve an accuracy of 10^{-4} on quaternion, which is far enough for our application.. Figure 3.11 gives an overview of the EKF step and sensor usage. The prediction step uses gyroscope data to compute the prediction of the covariance P and state vector X, which is normalized before the next step in our application case. The update uses accelerometer and magnetometer as observations data to update P and X, which is again normalized before the next iteration.

As mentioned before the accelerometer and magnetometer data are not used as such but normalized. This normalization of observation vector values is done for stability reason. The EKF mixes heterogeneous observation vectors each of these vectors having a specific ranges of values. Considering there is a matrix inversion during a filter iteration and the implementation uses fixed point, it is important for stability content to keep values on similar range.

The observation vector z is recomputed as follow: the accelerometer and magnetometer values are respectively divided by expected earth gravity and magnetic field norm which is shown in Equation (3.24). In fact there is no division but a product by the inverse which is

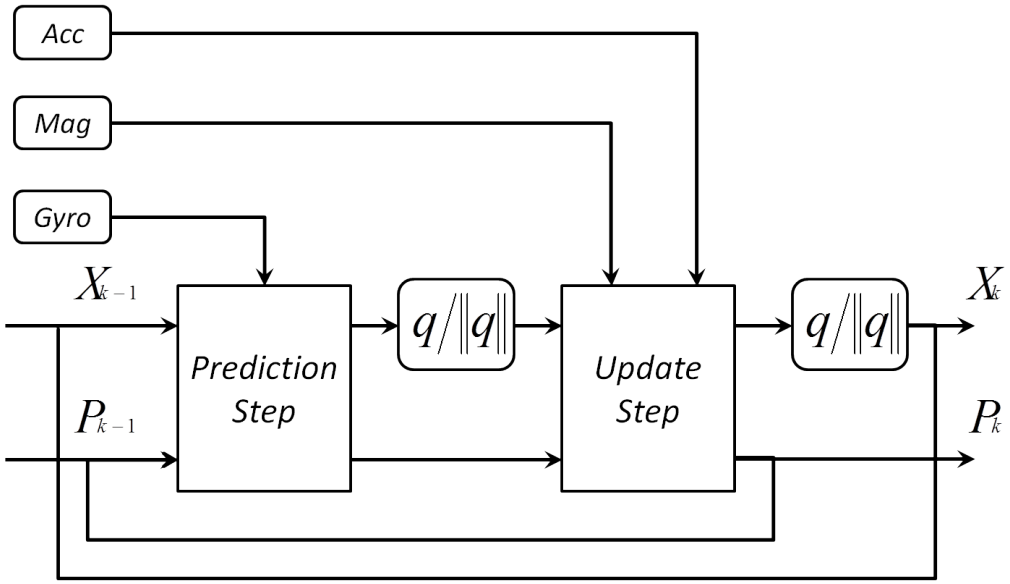


Figure 3.11: Schematic of Extended Kalman FIlter

pre-computed.

$$z = \begin{vmatrix} a_x \cdot (1/acc_{norm}) \\ a_y \cdot (1/acc_{norm}) \\ a_z \cdot (1/acc_{norm}) \\ m_x \cdot (1/mag_{norm}) \\ m_y \cdot (1/mag_{norm}) \\ m_z \cdot (1/mag_{norm}) \end{vmatrix} \tag{3.24}$$

For the same reason, this normalization is applied to $\vec{G}$ and $\vec{H}$. If we note δ the local inclination angle of magnetic field, the normalized values of $\vec{G}$ and $\vec{H}$ are given by Equations (3.25)-(3.26)

$$\vec{G} = \begin{vmatrix} 0 \\ 0 \\ -1 \end{vmatrix} \tag{3.25}$$

$$\vec{H} = \begin{vmatrix} -cos(\delta) \\ 0 \\ sin(\delta) \end{vmatrix} \tag{3.26}$$

It is important to note that good results of the EKF are liable to the matrices Q and R. The tuning of these matrices is to user responsibility and depends on the model and problem considered. The physical meaning of matrix Q is the model noise covariance. R is the observation noise covariance matrix. In case each dimension is independent Q and R are diagonal with the corresponding variance of each dimension.

There are two ways to determine Q and R values: theoretically or empirically. The empiric method can only be used if we assume that the matrix is constant. This is the case

of Q, the dynamic model f is reliable in all human motion condition, so we can consider that Q can be given the Equation 3.27. The matrix is diagonal with the same value q_n for all terms, which is the covariance noise of the quaternion.

$$Q = \begin{pmatrix} q_n & 0 & 0 & 0 \\ 0 & q_n & 0 & 0 \\ 0 & 0 & q_n & 0 \\ 0 & 0 & 0 & q_n \end{pmatrix} \tag{3.27}$$

The case of R matrix is different. The observation model assume that vector z is always equal to earth gravity vector and earth magnetic vector in any condition, which is not the case. In case of static measurement, the R value should contain the covariance of noise as in Equation 3.28 with σ_a^2 and σ_m^2 respectively accelerometer and magnetometer noise variance.

$$R = \begin{pmatrix} \sigma_a^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_a^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_a^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_m^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_m^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_m^2 \end{pmatrix} \tag{3.28}$$

As we will see in the result part, this solution of choosing a fixed value for R is not usable. EKF equations assume that the error of observation is Gaussian with zero mean. But, if the node is close to a strong magnetic interference the measurements are biased. Likewise, motion of node adds linear acceleration to earth gravity vector. This error does no match zero mean error models.

Then, depending on real-time condition, the reliability of observation values z can change. The solution proposed, inspired from [57], is to compute R dynamically at each iteration of the filter to temporary increase the σ_a^2 and σ_m^2 values to the actual error mean size.

The value of R must reflect the error between theoretical value of observation vector and actual observation. The difference between the two vectors z and $\hat{z}$ is called the innovation and noted $y = z - \hat{z}$. We note y_a the innovation related to the accelerometer and y_m the innovation related to the magnetometer. We can consider the use of these values for R, which gives the Equation 3.29. But, the drawback in this configuration is that R can increase even if there is no linear acceleration but only orientation error.

$$R = \begin{pmatrix} \|y_a\|^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \|y_a\|^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \|y_a\|^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \|y_m\|^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \|y_m\|^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \|y_m\|^2 \end{pmatrix} \tag{3.29}$$

The only thing we can assume is that the norm of accelerometer and magnetometer have to be constant and equal to 1 when normalized. So the other solution proposed is to add to

the sensor variance σ_a^2 a bias variance equal to the difference between observed and expected norm. R is computed as Equation 3.30 with biases on variance given by Equation 3.31 and 3.32.

$$R = \begin{pmatrix} \sigma_{z_a}^2 + \sigma_a^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{z_a}^2 + \sigma_a^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{z_a}^2 + \sigma_a^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{z_m}^2 + \sigma_m^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{z_m}^2 + \sigma_m^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{z_m}^2 + \sigma_m^2 \end{pmatrix} \tag{3.30}$$

$$\sigma_{z_a}^2 = |\|z_a\|^2 - 1| \tag{3.31}$$

$$\sigma_{z_m}^2 = |\|z_m\|^2 - 1| \tag{3.32}$$

3.3.3 Iterative Position Estimation

Iterative Position Estimation (IPE) is an algorithm that estimates position of nodes using distances between them. This method is not greedy in terms of computation and provides results as accurate as possible with the distances measurements.

We consider in our case no beacon, no prior location information and one measurement. The global cost function (STRESS function) simplified from S is S' and the corresponding local cost function is S'_i.

$$S' = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \omega_{ij} (\delta_{ij} - d_{ij})^2 \tag{3.33}$$

$$S'_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \omega_{ij} (\delta_{ij} - d_{ij})^2 \tag{3.34}$$

δ_{ij} is defined as the measurement of the distance between node i and all nodes j seen by sensor i. ω_{ij} is the weight accorded to the corresponding measurement which reflects the certitude of the measure. Finally,$d_{ij}(X)$ is the Euclidean distance between nodes i and j computed from estimated coordinate. This function uses the Euclidean distance d_{ij}, which implies the use of square root for its calculation. Here we propose a slightly different equation. The error $(\delta_{ij} - d_{ij})^2$ is replaced by $|\delta_{ij}^2 - d_{ij}^2|$. Then we do not need to use a square root anymore. This modification changes the profile of the cost function and adds a bias to the minimum. But in case there is a large number of node and all distances are in a limited range, the bias is small. The global and local cost function are C and C_i.

$$C = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \omega_{ij} |\delta_{ij}^2 - d_{ij}^2| \tag{3.35}$$

$$C_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \omega_{ij} |\delta_{ij}^2 - d_{ij}^2| \tag{3.36}$$

Another feature that differs from original algorithm is the choice of ω_{ij}. In case nodes are static, distance measurements are likely to be of constant reliability which mean ω_{ij} can be estimated once and fixed. In case of a moving node the shadowing effect and changing environment on the body modify the reliability of measurements. Weights ω_{ij} on distance error must be chosen wisely and dynamically according to link quality.

The last point to address is the method to minimize the local cost function C_i. The gradient method is too computation greedy since it uses several divisions. The solution proposed is to evaluate the local cost function for different locations of space. For each C_i, the only unknown variable is x_i, the current location of the node i. If we consider the initial value of x_i close to the minimum, C_i is locally monotone. Then, around the initial value of x_i, the global minimum of the cost function can be searched.

This evaluation of local cost function is done by binary search on variable x_i. At each iteration, the cost function is evaluated for 8 possible new locations of x_i, the minimum is adopted as the new initial position as we can see in Figure 3.12. The 8 possible locations are the vertex of a square. At each step the size of the square is divided by two. The binary search iterates until the size of the square is equal to the accuracy of the system. The initial size of the first square is important since it must be high enough to prevent to be stuck in a local minimum.

This iteration process is done for all C_i at the same time. As for the original algorithm, the final results of x_i are updated for all C_i. When the global cost function C reaches the minimum we assume the system has converged. In case of BAN this process can be done once when the user is not moving. But when the user moves, new iteration are constantly needed. To minimize the need in iterations and to increase the convergence rate, we combine inertial sensors to predict new location of nodes. Figure 3.13 shows the principle of combining inertial prediction and localization algorithm. At the frame k the IPE estimates the location of the node on the left side of the Figure. Blue dot is previous estimated location which is the starting point of the IPE. The new location estimated at frame k is the the green dot on the left. Purple circle represents the maximum space reachable by IPE at a given frame, which depends on the size of the first square. At frame $k+1$ the node moved to a new location due to the motion of the user. If we consider to start the IPE from the last location from frame k with the same step size, we see that the new real location, the green dot at the right of the Figure, can not be reached. The solution proposed is to make a prediction of the new location with the inertial measurement and to use this result to be the initial location

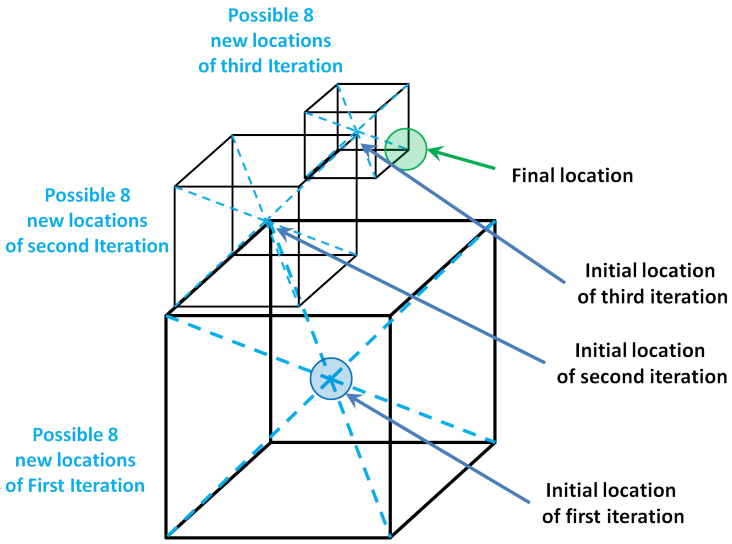


Figure 3.12: Dichotomy research method, example with three iterations. Each square represents an iteration which vertex are the locations where the cost function C_i is evaluated. For each iteration, the minimum among the evaluated locations is selected as the new initial location of the next iteration.

of the IPE. Moreover the first step size of the IPE is tuned with inertial measurements, large motion implies large range of exploration while no motion implies small range of exploration.

3.3.4 Classification

Classification is performed in two steps. The first step is a dimension reduction, similar to a data fusion. The second step is the use of a classification method on these new data.

The dimension reduction is performed by a Principal Component Analysis (PCA) [51]. The choice of PCA is due to its simplicity in terms of computation. Moreover the linearity of this method allows for the computation to be distributed over all nodes of the WBSN. An important parameter of PCA is the number of eigenvalues considered. Each eigenvalue contains a part of whole information. Then by accepting an information degradation, the number of dimensions kept can be reduced. As power optimization is a key factor, the number of eigenvalues must be minimized to reduce local memory size and the amount of data to transmit with wireless communications.

The second step is the classification by a Nearest Centroid Classifier (NCC). For each reference posture, the projection resulted from PCA is the center of this class. Then, for any posture tested, the distances between the projection and class centers are computed and the nearest one is chosen.

The other advantage of this solution is the flexibility of the implementation for different use cases. Locally on the node, the computation is just a matrix multiplication regardless of the postures considered. Only coefficients have to be updated to change the posture set.

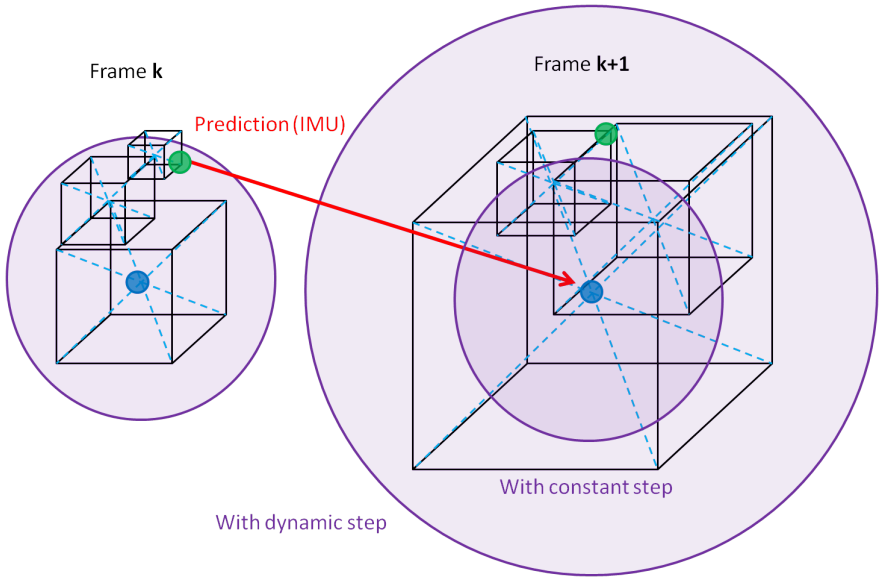


Figure 3.13: Dichotomy research associated with inertial prediction.

3.4 Results

3.4.1 Extended Kalman Filter

This section provide an example to show the importance of the R value. Figure 3.15 shows the three axis of accelerometer measurements of the right ankle for a Capoeira gesture (Figure 3.14). We can see that:

- At the beginning the node is static.
- Around frame number 27, the first noticeable linear acceleration peak appears.
- Around frame number 50, the first huge linear acceleration peak begins.

Figure 3.14: Frames extracted from the capoeira move.

We can see here that linear acceleration can be very large compared to gravity. The Kalman filter has to deal with this acceleration by decreasing the certitude of the observation.

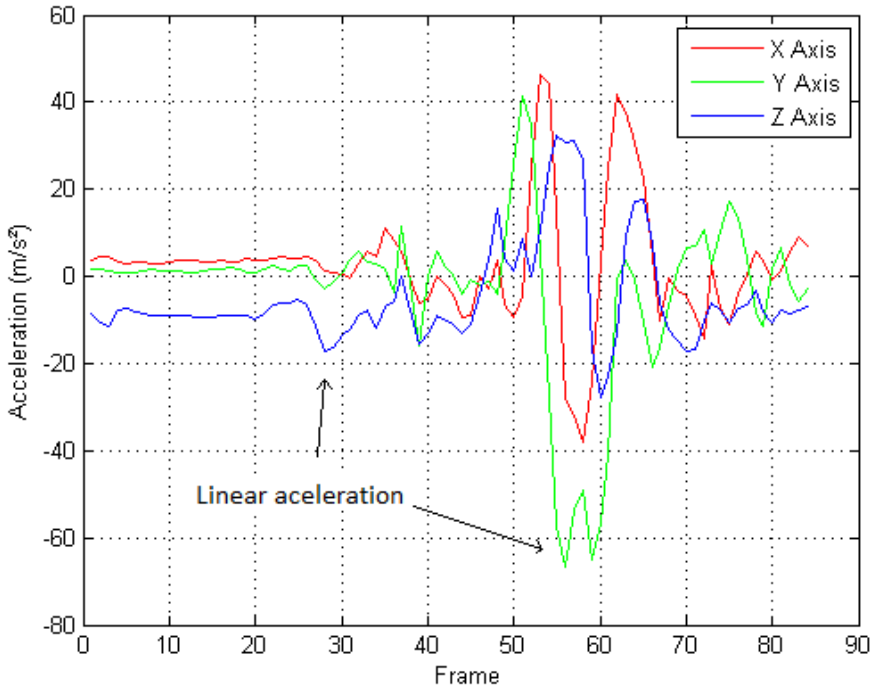


Figure 3.15: Accelerometer measurement for the node located on the foot during a Capoeira motion.

Figure 3.16 gives quaternion EKF results with R fixed at the sensor noise level. The red line is the quaternion estimation and the blue is the original quaternion. We can see that after frame number 27, which the first linear acceleration peak, the tracking begin to fail. The filter cannot deal with linear acceleration with a fixed value of R.

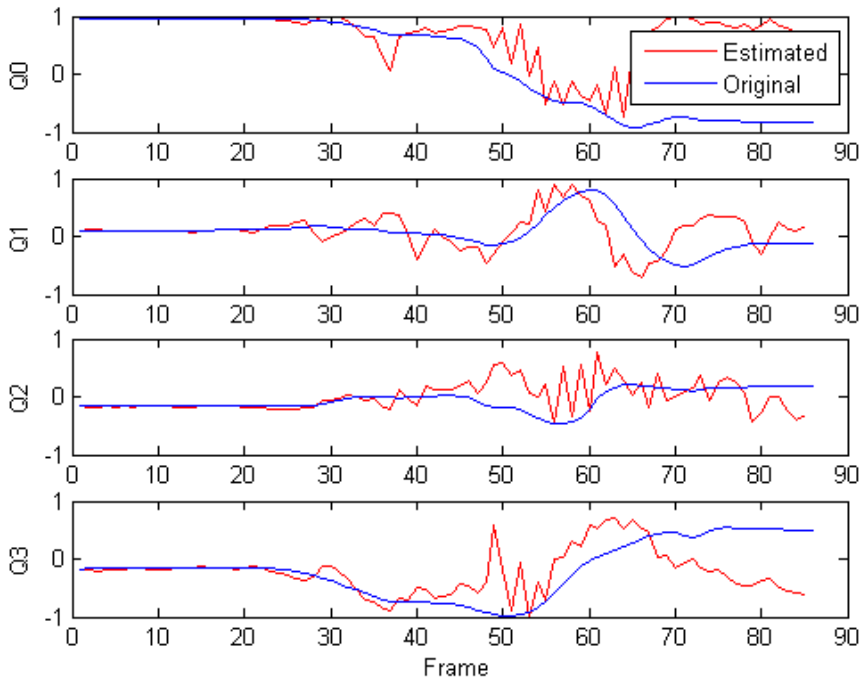


Figure 3.16: Quaternion estimation by EKF without dynamic R for each frame of the motion in red compared to the original quaternion in blue.

As presented in previous section, R can be dynamically computed by several means. Figure 3.17 shows the result of the same example with R computed with innovation using Equation (3.29). Results are better but the tracking fails again with a large linear acceleration around frame 50. The innovation is not suited for the computation of R. The reason is that any error in orientation implies an increase of innovation. A bigger innovation means a bigger R for both observation accelerometer and magnetometer. So, with this solution, if the orientation error is only due to linear acceleration it impacts the certitude of magnetometer observation anyway.

The other solution using norm subtraction, see Equation (3.30), solves the issue of cross-observation disturbance. The strong point of a Extended Kalman filter is its ability to take the best from each observation sources independently. Figure 3.18 shows the results using this method. We can see that tracking is not perfect, but is largely improved compared to the previous methods. Around frame 40, a small error appears which was not present on previous method. But from a global point of view the accuracy is far better, the tracking does not fail even with a large linear acceleration. This is due to the fact that for a short

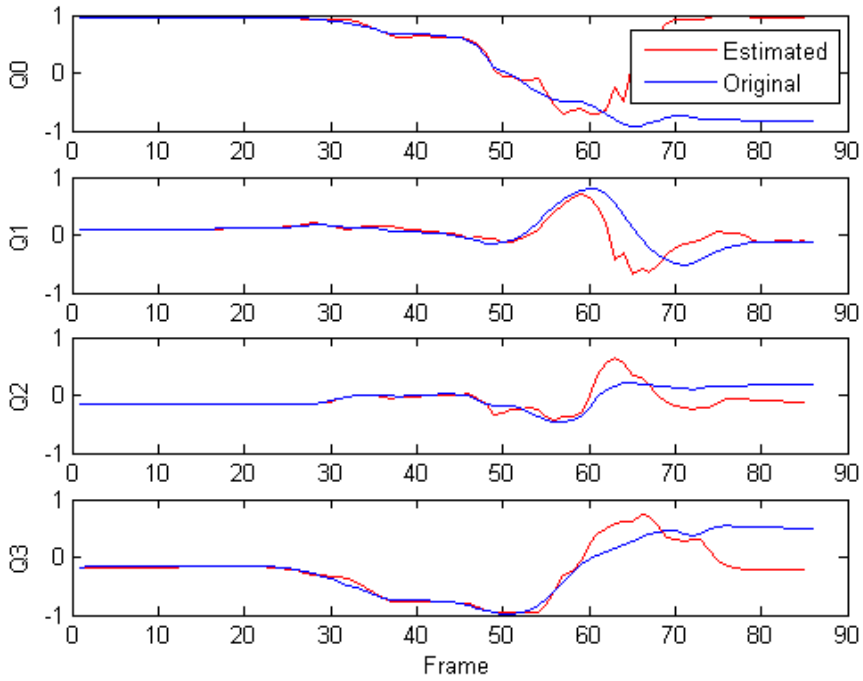


Figure 3.17: Quaternion estimation by EKF using dynamic R with innovation for each frame of the motion in red compared to the original quaternion in blue.

duration, even if accelerometer measurements are incorrect the magnetometer still continues to observe relevant data.

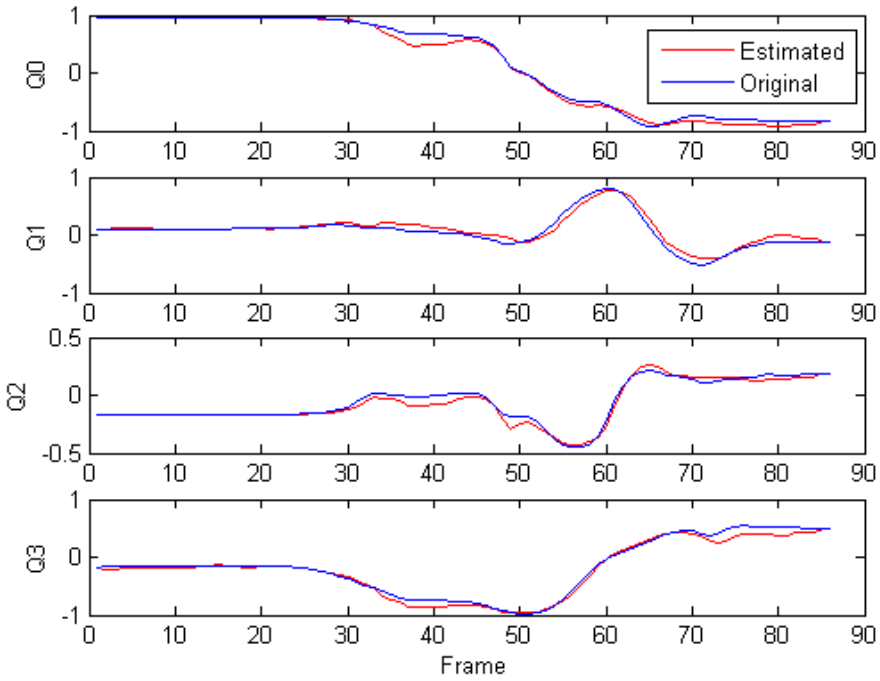


Figure 3.18: Quaternion estimation by EKF using dynamic R with norm subtraction for each frame of the motion in red compared to the original quaternion in blue.

Obviously, there is a limit on the duration and error that can be compensated by magnetometer. But, remind that the example presented is a high speed movement which is one the most extreme case we can encounter in real life experience.

3.4.2 Posture Recognition

In this section we present results on posture recognition. For each case experimental conditions are first described before giving some results on recognition accuracy for different conditions and algorithms. Results are obtained with the simulator presented in section 2.2.2 and using data from a motion capture file recorded for this application case.

The 12 postures depicted in Figure 3.19 were chosen as a testbed for this case study. Nodes of the WBSN are located on the shoulders, elbows, wrists, chest, and hip. There are two kinds of errors that affect the inputs. The first one is the error on the posture reproduction and the second one is the sampling noise of the inertial sensors (IMU). An Additive White Gaussian Noise (AWGN) is used as a model for the IMU noise as we have seen in section 2.2.2.3. The standard deviation of sensor noises is set to 2% of gravity norm for the accelerometer and to 2% of earth magnetic field norm for magnetometer. For distances an AWGN is also added with a standard deviation of 1 centimeter. This noise model on

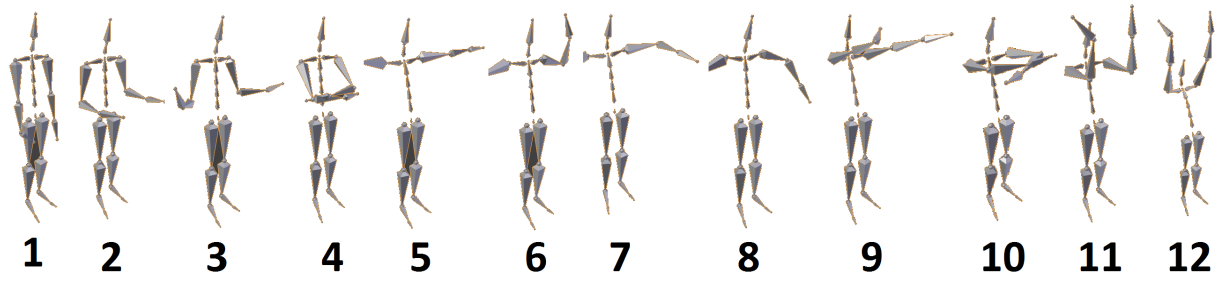


Figure 3.19: Posture of arms chosen for recognition.

distances corresponds to the case where distances can be evaluated with centimeter precision. All following results are obtained by the average of 1000. The orientation errors on limbs are with a standard deviation of 0° to 10°. Considering that the differences between angles of the 12 different reference postures are about 90°, then all postures can be classified with a very low asymptotic error.

PCA is tested for different numbers of eigenvalues. A higher number of eigenvalues means more precision but also requires more data to be transmitted. First simulation results showed that only few eigenvalues are enough to give good recognition rate for each posture. Following simulations are performed with only 2, 3, and 4 eigenvalues.

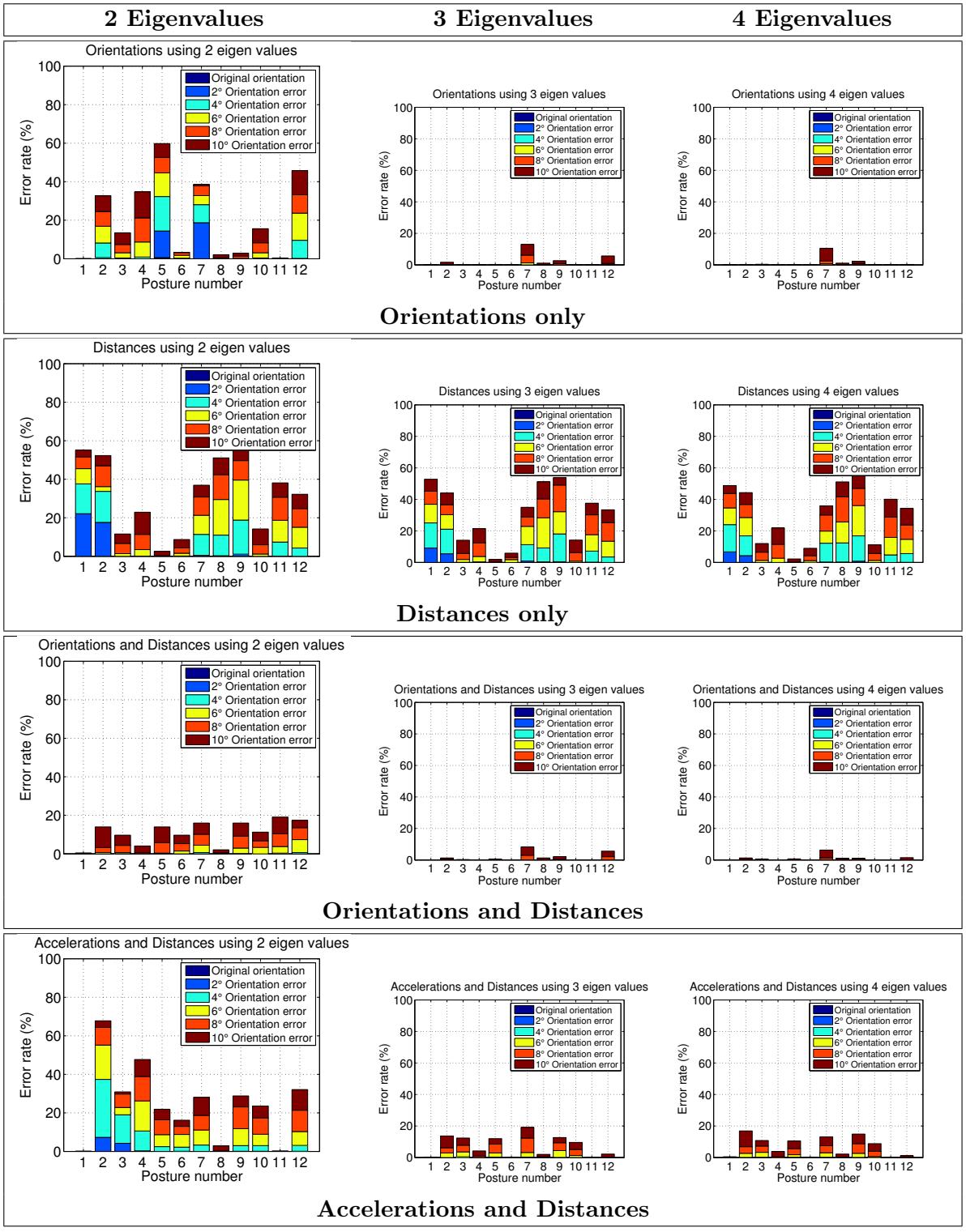
Figure 3.20 gathers simulation results with 2, 3, and 4 eigenvalues (each column of the Figure) for four combinations of inputs: *orientations only*; *raw distances only*; *orientations and raw distances*; *raw accelerometer data and distances* (each row of the Figure). Simulations are performed to measure the recognition tolerance for orientation errors that represent the inexact reproduction of posture by the user. For each configuration and each posture we added orientation error on all body angle of 2°, 4°, 6°, 8° and 10°. These errors correspond to the different bar colors in Figure 3.20. The value of each bar is the rate of failed posture recognition for the configuration considered.

The first row of Fig. 3.20 shows that inputs with *orientation only* provide good results with at least 3 eigenvalues. The maximum error is about 12% for posture 7 and other postures are about 1%. But with only 2 eigenvalues, the decrease in information is too large to provide good results. Even orientation errors of 2° or 4° lead to errors of up to 30%. On the other hand, the difference between 3 and 4 eigenvalues is not significant, and adding more eigenvalues does not decrease error rate significantly.

The second row of Fig. 3.20 shows that *distances only* are not tolerant at all to orientation errors. There is no error with original postures, which means that 1 centimeter precision on distances is enough to allow the discrimination between postures with no orientation errors. But, no tolerance to orientation error means that the user has to be extremely accurate on his posture. In most of the cases, this is a strong constraint for the user and for the application. Moreover, increasing the number of eigenvalues does not improve significantly the results. Therefore, *distances only* are not a good candidate for a posture classification with PCA and NCC.

The third row of Fig. 3.20 presents interesting results. Even if *distances only* does not

Figure 3.20: PCA error rate for different combinations of inputs and number of eigenvalues.



help that much, the combination of *distances and orientations* provides better results than *orientations only*. Results with 3 and 4 eigenvalues are similar and slightly better for the third combination. Once again, posture 7 gives the maximum error with about 9%. The interesting point here is the case with only two eigenvalues on the left that shows a real improvement. The maximum error is about 20% with no significant error before 6° of orientation error. Depending on the accuracy required by the application, two eigenvalues can be enough.

Magnetometer data could be very frequently disrupted by other objects (e.g. phones) or by parts of metal. In this case, only *accelerometer data and distances* can be used and results without magnetometer are given in the fourth row of Fig. 3.20. From the figure, it can be observed that 3 or 4 eigenvalues give good results with no significant error before 6° of orientation error. This shows that even without magnetometer, a classification is possible with an acceptable error rate. Moreover, *acceleration and distances* are independent of north direction, which means that no pre-processing on inputs is necessary to remove this dependency. Raw data extracted from sensors can be used directly in this case.

3.4.3 Gesture recognition

All postures forming a gesture can be considered as a posture library. Applying a PCA on this library would lead to a continuous curve in the PCA space. Then a curve matching algorithm could be used to recognize a gesture. Here we propose and use a less sophisticated method based on the same principle. We consider two motions, a quick motion and a slow motion. The quick motion is the capoeira movement recorded with the Xsens/MVN system [52] and is composed of 84 frames at 30 frames per second (fps), some frames of this gesture are depicted in Figure 3.14. The slow motion is not recorded but generated. This movement corresponds to the transition from static postures 3 to 11 and from 11 to 6 from Figure 3.19, at 30 fps with a duration of 3 seconds in total (from 6 to 11 and 11 to 6). For the classification, the reference postures of the gesture are taken every 10 frames, which mean 3Hz. The nodes of the WBSN considered for these simulations are those located on ankles, knees, elbows, and wrists.

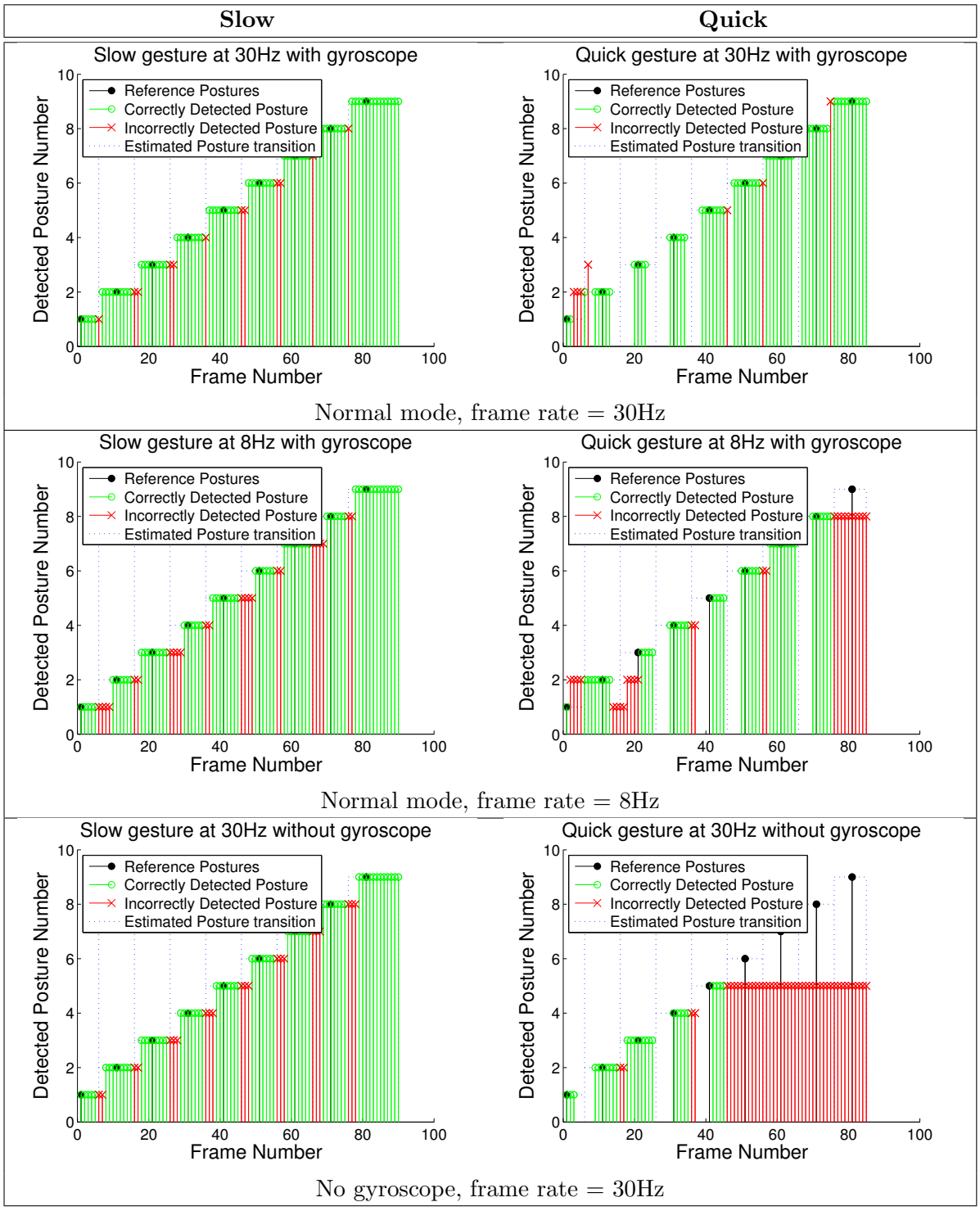
Inputs of PCA classification are orientations of nodes represented by quaternion calculated by EKF. For each gesture, three configurations are tested. The first configuration is the mode with gyroscope of the EKF at 30Hz, the second one is the mode with gyroscope of the EKF at 8Hz, and the last one is the EKF without gyroscope data at 30Hz. The objective is to test different sensor data sets and acquisition rates that directly impact power consumption. Figure 3.21 presents the results of the gesture recognition for the two gestures and for the three configurations. The abscissa represents each frame of the motion. The ordinate represents the classification result. For each frame of the gesture, the detected posture, if any, is given in green with a circle if the result is right and in red with a cross is the result is wrong. We added, in blue dashed lines, bounds that delimit time domain of each reference posture. This bounds are at the middle of each consecutive references postures, which are represented by black dots.

In the first configuration, at 30Hz, both slow and quick gestures are well decomposed and recognized. We can clearly see a regular monotonic progression of detected posture number that corresponds to the continuity of the gesture. In the second configuration, at 8Hz, the slow gesture is still well decomposed. Even if some postures are not within the right bounds, we can observe that the detection profile is still regular and monotonic. On the other hand the quick gesture is not so well decomposed, especially in the beginning and at the end. This is partially due to the fact that the first and the last postures of this gesture are physically similar, indeed this movement is a martial kick so the user is in guard at the beginning and at the end of the motion.

For the two considered gestures, the reduction of sampling rate (and so the computation requirements) does not have the same impact. This shows that, depending on the type of application (slow or fast movements), a coarse or fine detection can be used. Therefore, different algorithm configurations can be considered to address each case and thus allowing for a significant reduction in the energy consumption.

In the last configuration, without gyroscope data, the two gestures show significantly different results. The slow gesture is still well decomposed but the algorithm is unable to decompose and to track the quick gesture. This shows the importance of gyroscope

Figure 3.21: Results on gesture recognition of slow and quick movements.



measurement in case of quick motion tracking. This is a really interesting achievement since the power consumption of the gyroscope is three order of magnitude higher than the accelerometer. Then, these results give room to save energy by keeping high computation rate but by cutting off the gyroscope when considering applications dealing with slow gestures.

In these results, bounds that delimit the time domain validity were chosen arbitrary in the middle of references postures. But the real posture transitions do not necessary happen at the chosen middle time. So in the case where a fine detection is required, future work may include the definition of a precise time domain or domain transition.

Moreover, there are many ways to compute a metric for Quality of Service (QoS). For example, we can take into account the duration of each posture in addition to the sequence of postures. The metric has to be defined accordingly to the application chosen. The advantage here is that this metric is only computed by the central node (e.g. a smartphone), which can be configured. This kind of configurations can be explored with our simulator which was designed for that purpose.

3.4.4 Update of Comparison Between Radio and Computation

Based on our results we can now refine the estimation of the Radio/Computaion tradeoff introduced in Section 2.1.4 with Figure 2.1. For this new estimation we need to know the number of bits to send, the amount of bits reduction by fusion and the computation load of the EKF. Table 3.1 provides the number of operations that compose the EKF. We see that we have only 6 divisions for more that 3500 multiplications and 3000 additions. For our estimation we approximate the computation load to 3500 MAC.

Table 3.1: Number of operations of one iteration of Extended Kalman Filter.

ADD	MULT	DIV
3088	3584	6

We now consider the number of bits to send. Figure 3.22 shows a typical frame considered. We have the data, and few bits for the identification, the configuration mode and an overhead. If we send all data, the frame is $135bits$ for sensors and $19bits$ additional, which makes $154bits$. If we send only classification data, the frame is $3 \times 12bits$ for sensors and $19bits$ which makes $55bits$. This allows a reduction is $99bits$, which correspond to 64%.

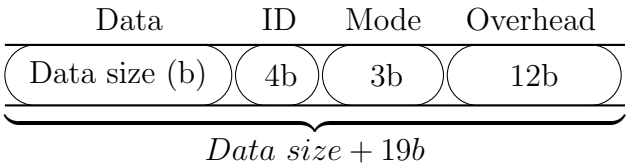


Figure 3.22: Radio frame considered composed of data, identification of nodes, the node configuration and an overhead.Values are given in bits.

We can see in Figure 3.23 a better approximation of the tradeoff between radio and computation. The radio cases have not changed compared to the preliminary study, Zigbee and Zyggie are about to match the budget objective if we do not consider sensors. The ultimate projection IR-UWB radio still works for our budget. On the other hand, computation values have been multiplied by 3.5. We can see that the Cortex M3 seems now borderline as a solution for the system. The $28nm$ consumes about 20nJ for 1 iteration and the $65nm$ consumes about 100nJ. These values are considered as the lowest bounds that could be achieved by these technologies using a full custom architecture. If these values can be reached the power budget would be easily met by our system. Note that these estimations consider a data of $32bits$ fixed-point. Figure 3.23 also shows that the absolute limit to be lower for 1 iteration is 1.29μJ. Otherwise the data fusion would not be able to provide an energy advantage compared to radio only and meet the budget at the same time. In the same way, the absolute limit to be lower for $1bit$ transmitted is 35.9nJ, if we consider a data fusion of 20nJ for 1 iteration, which is our lower reachable bound. The limit to be able to dispense with data fusion is 12.4nJ, which is a third of the Zyggie value.

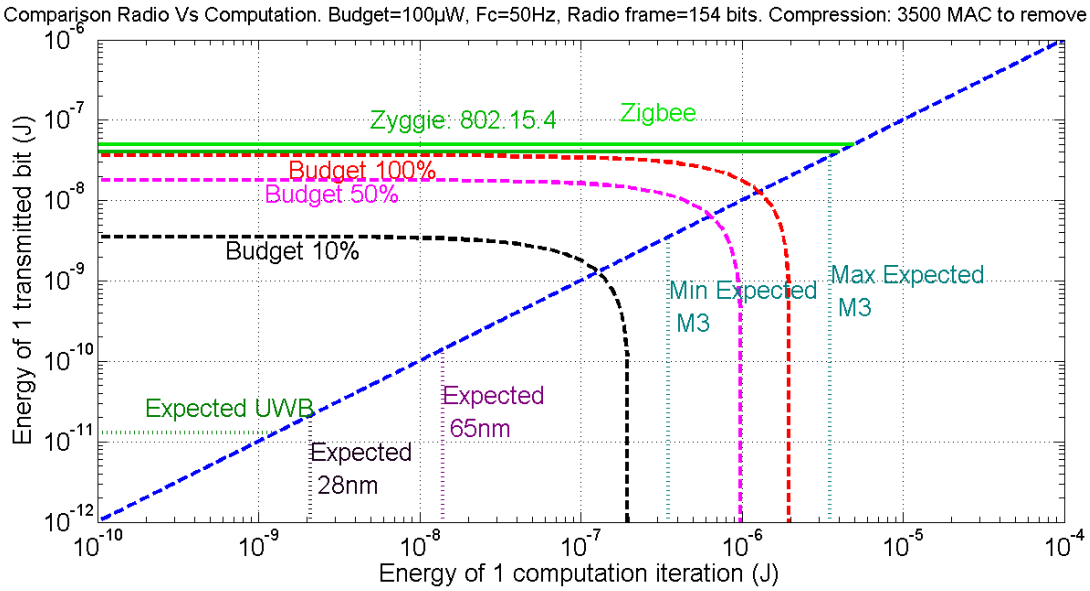


Figure 3.23: Estimation of the comparison between Radio and Computation for our system.

3.5 Conclusion

In this chapter we studied solutions for posture/gesture recognition and motion capture. We saw that in our case of no equipped environment, the number of solutions is restricted. Systems that can be used outdoor like Moven [22] [52] consume a lot of energy to be able to provide the high performance required by their applications. Moreover the real-time fusion of data requires the use of a computer. We also saw a localization method using distances and proposed a different method more adapted to be implemented. But, in our case we do not consider the use of external beacons, which would allows us to make an exact motion capture. Moreover the distance measurement has a relative high cost in energy with current technology. The use of UWB radio that would allow this measurement is not possible by now for an energy autonomous system. We proposed here a solution with an accuracy in accordance with our goal. We use the EKF to estimate the orientation of nodes and then a classification method to recognize postures or gestures.

Results for posture recognition showed that it is possible to use PCA to reduce the number of dimensions that represent a posture, in our case down to 3 eigenvalues. We tested several combinations of data for classification. Three of the tested solutions performed well and have their own advantages . The first with only orientations requires accelerometer and magnetometer data. The second combines orientation with distances and provides slightly better results at the cost of radio measurements. The third one combines raw accelerometer and distances for simple implementation and still provides good results.

The gesture recognition use case showed that even a quick gestures can be recognized. Moreover if we tune the frame rate according to the motion, we can still recognize motion and save power. This opportunity will be addressed in Chapter 5 with the introduction of

dynamic frame rate.

In the next chapter, the architecture that implements the EKF is designed and characterized in terms of area cost and energy consumption.

4

Design and Implementation of the Architecture of an Extended Kalman Filter

4.1 Introduction

In the state of the art of solutions for posture or gesture recognition using BAN, the computing part is usually managed by a microcontroller.Table 4.1 provides the typical energy cost for a state of the art solution composed of a Cortex M3 microcontroller and a Zigbee transceiver. The energy cost of computing is given for Multiply and Accumulate (MAC) operation. The energy cost of the transceiver is given for 1*bit* transmitted. Systems that use this solution as [22] and [58] are not meant to be used in everyday life, so they exempt from heavy power and footprint constraint.

Table 4.1: Computing VS Radio energy cost for state of the art solutions.

Type	**Energy cost**
Zigbee (TX/bit) [16]	50 nJ
Cortex M3 32b MAC [18]	$0.1 - 1$ nJ

The use of a microcontroller limits the energy efficiency. However, [59] gives power results of different implementation of a same kalman-like algorithm. Results show that we can reach a reduction of one or two magnitude of energy with an Application Specific Integrated Circuit (ASIC) compared to a microcontroller. To the best of our knowledge it is the only evaluation of a Kalman-like filter implementation using an ASIC. Table 4.2 provides the energy cost for a MAC in ASIC technology and the energy cost for 1*bit* transmitted with a Impulse Radio Ultra Wide Band (IR-UWB) transceiver. This shows that the advanced solution consumes far less power than the state of the art of solution. This leads to the design of an ASIC solution for our algorithm implementation.

In this Chapter, we first evaluate the relevancy of our solution with our own functional simulator, which is used in the first design steps to evaluate the accuracy of algorithms and the computing complexity with real data captured with state of the art sensors. At circuit level, we estimate the power consumption of our solutions by means of gate-level synthesis and power estimation tool with 65nm BULK CMOS and 28nm VTB-FD-SOI technologies.

Table 4.2: Computing VS Radio energy cost for advanced solutions. Results on 65nm and 28nm were obtained after gate-level power estimation using PrimeTime from Synopsys.

Type	**Energy cost**			**Comment**
IR-UWB (/bit) [17]	13pJ			@ 2.2Gb/s
ASIC	**32-bit**	**16-bit**	**8-bit**	
65nm MAC	34pJ	3.0pJ	0.30pJ	Synopsys/PrimeTime
28nm MAC	5.9pJ	571fJ	72fJ	-

In the following, Section 4.2 explains step by step the global design strategy. Section 4.3 presents the context at the application level with algorithm choices. Section 4.4 gives our strategy of approximate computing with bitwidth selection. Section 4.5 explains the HLS tool exploration steps and results. Finally, Section 4.6 gives technological results of the selected configuration.

4.2 Design Flow Strategy

4.2.1 Exploration strategy

Before choosing an architecture we must study what should be implemented. A preliminary overview of algorithms is done to evaluate which part of computing we must focus on. On this step we select the algorithm by an evaluation of the operation load.

After we decided which part of computing to implement, the algorithm is described in C++ code. The global design-flow used is depicted in Figure. 4.1. A HLS tool is used to transform C++ code into a Register Transfer Level (RTL) description. Finally, the RTL code is transformed into Gate level with the use of Design Compiler and the chosen technological library.

This method involves a lot of steps with at each level several possible choices providing different performance. The combination of all possibilities could produce a huge amount of solutions, which are not possible to entirely explore. In fact, at each step, architecture and results become more accurate but also take longer time to be produced. The strategy is to use intermediate results to select at each step only few configurations as on the right hand side.

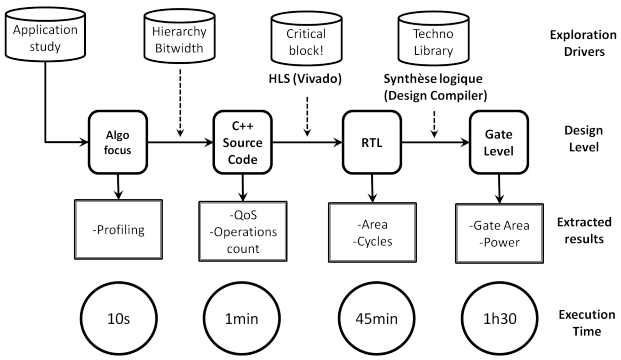


Figure 4.1: Architecture development steps

4.2.2 Exploration Steps

The design space exploration is decomposed into four steps, which are presented in the following sections. This step steps are: the application study, the approximate computing, the architecture exploration using HLS and the technological differentiation.

4.2.2.1 Application study

This step has two objectives: to propose a global architecture that targets the two applications posture and gesture recognitions. The second objective is to show why we focus on the implementation of the EKF.

4.2.2.2 Approximate computing

At the algorithm level, the specification is a C++ code. It permits to quickly simulate and compare a lot of different algorithm choices.

Criteria used to compare configurations are on one side the quality of service provided and on the other side the estimated cost for this quality of service.

In this step we explore data bitwidth, which is a strong scaling factor regarding area and power consumption. It is critical to optimize this parameter while considering the impact on accuracy. We can consider the accuracy for the EKF or for the application level. A reduction of the EKF accuracy do not necessary mean a decrease in application accuracy. This means we can adapt bitwidth to the application level accuracy to reduce it as much as possible.

Our strategy to choose the bitwidth of data is the following. First we define which parts of the computation need specific bitwidth. This will produce a set of bitwidth parameters that will determine an accuracy configuration. Then, with a preliminary study on operator power consumption, all configurations tested are compared to each other. Finally, only a set of solutions with the best Accuracy/Energy tradeoff are kept. Here we choose several application accuracy constraints, then we select a set of four configurations that will represent the best candidates for energy saving for 4 levels of accuracy called Very High (VH), High (H), Medium (M) and Low (L).

4.2.2.3 Architecture Exploration using HLS

At this point the designer must decide how the architecture is organized in terms of functional blocks. With current HLS tools, the choice of designers is simplified but still has a strong impact on the results. We can decide the restructuring of the algorithm code, the instantiating or reuse of functional blocks and how the blocks are connected. The HLS tool offers several options to easily test different loop implementations like pipelining and unrolling. This step is crucial, it is based on the use of HLS tools that speedup exploration which is a support to the designer skills. A good knowledge of the algorithm is necessary to be able to wisely decide the block decomposition, while keeping in mind local HLS customization possibilities to optimize critical part.

Hierarchical configurations Two hierarchical implementations are considered. The first configuration is a straight-forward data-flow, if a block must be used several times for 1 iteration of the EKF, a different instance of the block is used each time. The second configuration implement one instance of each functional block but manages the data-path to reuse several times the blocks needed to provide the same results.

Bloc optimization Some critical blocks are identified for many reasons. It can be related to accuracy concern for algorithm stability. This stability is carefully considered with all solutions provided by the approximate computing step. This can also be due to multiple instances of a given block, which means a factorization effect of any block-level optimization. This case drives the choice of HLS options, especially on loops, to find the best tradeoffs.

4.2.2.4 Technological differentiation

The HLS tool generates a RTL code. This can be transformed into a Gate level description considering a technological library. A simulation at this level gives an accurate power consumption of the design but requires a huge amount of memory and time. It is necessary to reduce the number of configurations selected in previous steps. Here, we select a single implementation of the design architecture and explore different technological options and the impact of accuracy on energy consumption and area.

4.3 Application Level Exploration

The BAN is designed to implement a posture/gesture recognition system with ultra-low power nodes. The proposed architecture is based on the algorithms studied in the previous chapter. Figure 4.2 gives an overview of the node computation steps. The first block *Motion Detection* uses accelerometer sensor for motion detection. If the application considered is posture detection, the motion detection will drive the *static orientation* block only when it detects no motion. If the application is the gesture recognition the motion detection block drives the *AHRS* block to dynamically compute orientation with a maximum rate of $50Hz$.

A local *classification projection* that depends on the application is then applied before data are sent to the central node.

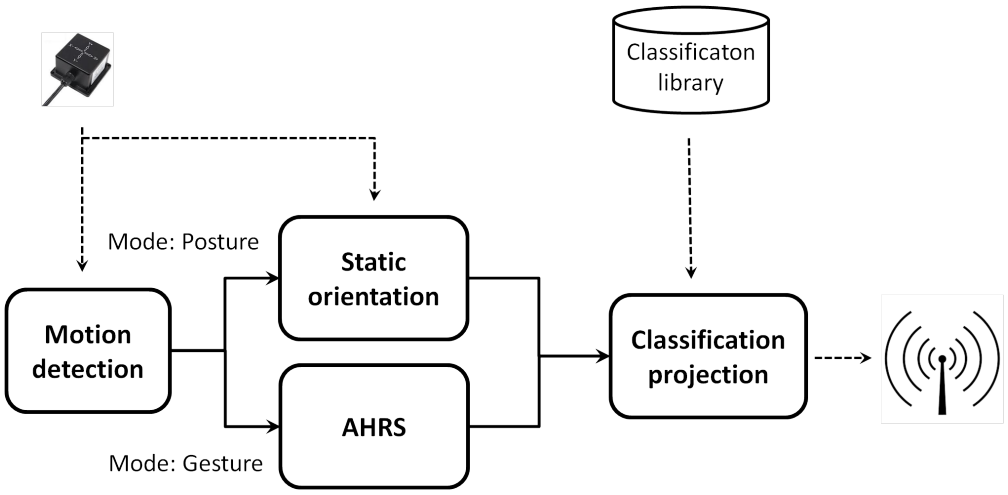


Figure 4.2: Computation block steps for posture and gesture recognition mode

The motion detection has to be really light since it is used to wake-up other parts of the system. It can be performed with a simple digital second-order high-pass on acceleration in addition with a thresholds system. The orientation system can be far more complex depending on accuracy needs. In the static mode, few vector computation can be used. In the dynamic mode, the orientation estimation requires a real-time tracking performed by AHRS. In our case, we implement an EKF algorithm which is the most complex computing block. The classification projection step is a pre-computing of local data for the global classification. The computing complexity is equivalent to a matrix product.

This overview shows that the design effort has to focus on the implementation of EKF since it is by far the most computational intense part.

4.4 Approximate computing

In this step we explore approximate computing opportunities to define the bitwidth of the design. At this level, the algorithm is described in C++, so that it can be quickly simulated. Moreover, these simulations can take into account fixed-point arithmetic number representation and real application case data. In a first step we specify parameters that define the design bitwidth. In a second step we have to choose which values of bitwidth are selected from an accuracy and energy tradeoff study.

4.4.1 Parameter definition

In the implementation, every operator and data-path have a bitwidth. Ideally, in a custom implementation, each bitwidth is chosen as small as possible to save power while it complies

with accuracy requirement. If we take a look at the Equations (3.11)- (3.18), we can see that the matrices F, H, P, K and S are multiplied to each other. The matrix product is used several times for different matrices combination. Considering the individual bitwidth for each matrix variable and custom matrix product for each equation would take a big amount of time and complexify the implementation. Moreover, the EKF is an iterative filter where the output becomes the input of the next step. Lastly, we intend to implement a version with only one matrix product block for all equations. This leads to consider that the whole design has the same bitwidth. In fact, we consider that most of the architecture has the same bitwidth and only specific variables or operators have their own bitwidth. All different bitwidth values are the parameters that will define the global accuracy of an architecture.

The first parameter we consider is the default bitwidth for the whole design. Another specific bitwidth is also used for quaternion data, which are X and $\hat{X}$ in the EKF Equations (3.17) and (3.19). The bitwidth of these variables is the second parameter. Finally we add two parameters, which come from potentially critical points of the design: the quaternion normalization and the matrix inversion. The quaternion normalization, which is implemented using an algorithm derived from Goldschmidt iterative reciprocal square root algorithm [56], is important for the stability of the EKF and thus potentially requires to extend locally the bitwidth of data. The matrix inversion required in Equation (3.16), which is implemented using a Gauss-Jordan elimination, is also important for the stability of the EKF and thus potentially requires to extend locally the bitwidth of data.

This makes a total of 4 bitwidth parameters to explore and define the design bitwidth, each configuration implying a different fixed-point accuracy.

4.4.2 Configuration selection

In this study, the selection will be done considering the same use case for all configurations. This is the recognition of the "Capoeira kick", which is a complex and fast motion, so an extreme case for the algorithm. The error of the EKF quaternion is the combination of error due to implementation approximation and the error due to the algorithm model itself. The error due to the implementation have to be smaller than the error due to the model.

The EKF provides quaternion values that are used to compute the classification. The idea is to determine how the accuracy at application level is affected by the error on the EKF quaternions. In a first step, we study the effect of the quaternion error on the classification error for the Capoeira motion. Figure 4.3 gives the error rate of classification (the color map) of each frame (in abscissa) as a function of the error on the original quaternion (in ordinate) considering a quantification error model AWGN with standard deviation in decibel. This Figure has been made with a simulation of 1000 runs for the statistics. We can see that below $-40dB$ on quaternion error there is no major classification error, but we must reach $-80dB$ to fully remove the error caused by quantification. Above $-40dB$, classification errors appear mainly for frames 1 to 24, the other frames are not really affected even until $-10dB$. This is due to the number of nodes whose redundancy permits acceptable results even with a high noise level.

Now, the accuracy of the algorithm is evaluated for an execution using floating point

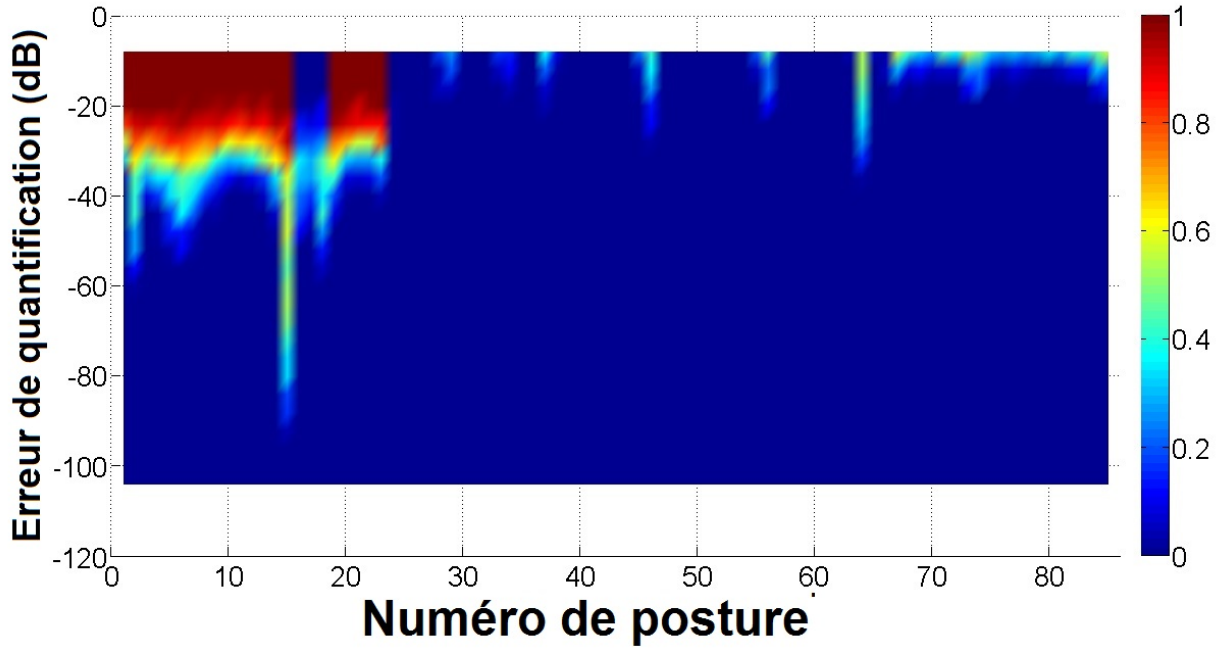


Figure 4.3: Application level: classification error rate for each frame in function of a quantification error on quaternion.

data types to measure error only due to the EKF model. This accuracy of the EKF depends on the motion activity of the node considered. We tested both dynamic and static cases. The dynamic case is the Capoeira kick. For the static case we considered a motionless node. Results are given in Table 4.3. The error is computed using the Root Mean Square (RMS) on angles since RMS includes average and standard deviation and angles are more relevant. We also give the quaternion standard deviation corresponding. The angles representation used to compute orientation error are the Euler angles Yaw, Pitch and Roll, the intrinsic rotations. The RMS error on Euler angles for static case is around 1° accuracy, which corresponds to a quaternion standard deviation of $-37dB$. The error is a magnitude higher for the Capoeira motion with around 8° and $-25dB$.

Table 4.3: Euler angle and quaternion accuracy of Extended Kalman Filter using floating-point data type.

RMS Float Error °	Roll	Pitch	Yaw	Quat std (dB)
Capoeira kick	7.3°	5.8°	9.3°	$5.8 \cdot 10^{-2}(-25dB)$
Motionless	0.86°	0.92°	1.5°	$1.4 \cdot 10^{-2}(-37dB)$

We can compare these results to a statistical analysis of the quaternion error effect on the Euler angles. Figure 4.4 shows the standard deviation (dashed lines) and mean (full lines) of error on Euler angles as a function of standard deviation on quaternion for an AWGN error model. We can observe that standard deviation on quaternion and Euler angle are

linearly correlated, which means we can use both error representations. We can see that a statistical error of 1° is equivalent to an error of $-40dB$ on quaternion, which is close to the $-37dB$ of the static case. For the $-25dB$ corresponding to the Capoeira motion we fine a statistical error of about 6°, which is close to the simulated value. We can consider that the error computed during the simulation of Capoeira is relevant of the statistical error.

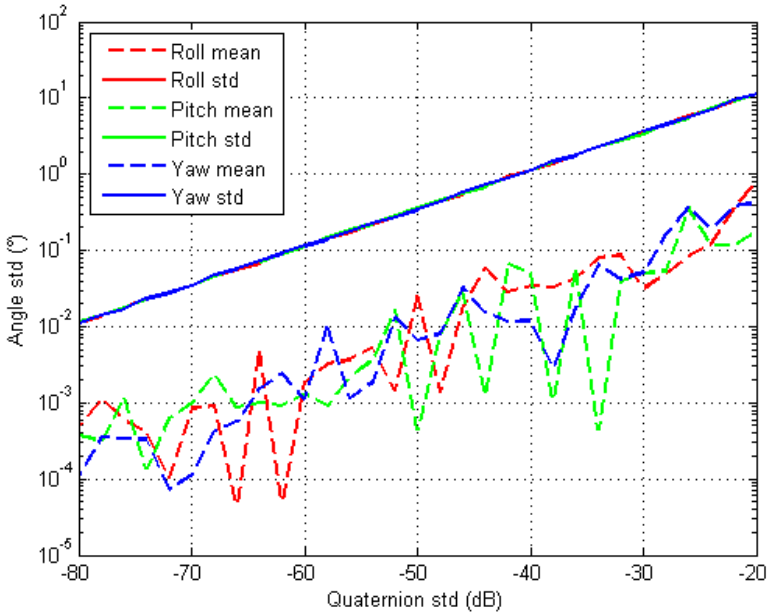


Figure 4.4: Standard deviation of each Euler angle as a function of standard deviation of quaternion.

The last step is to evaluate the impact of the different bitwidth configurations on algorithm accuracy. The error between the floating point execution and each bitwidth configuration using fixed-point is evaluated. The metric used is the standard deviation of quaternion. Using pre-estimated operation power consumption we can estimate the energy of one iteration of each configuration. The pre-estimation values have been made using Design Compiler and PrimeTime simulation on individual operators of different bitwidth. This energy is obviously not representative of the real total energy but this is a good indicator to compare configurations. For each of the four bitwidth parameters defined in Section 4.4.1, the number of operations is counted considering 1 iteration of the EKF. Table 4.4 gives the results of how many additions, multiplications and divisions are required for each parameter. We can see that most of operations, around 90%, are performed with the default bitwidth, which will be dominant in the overall energy consumption.

All configurations tested are plotted in Figure 4.5, where each cross represents a configuration as a function of its estimated energy cost in Joule for 1 iteration of EKF (in x-axis) and its accuracy on quaternion standard deviation (in ordinate) computed by simulation. In combination with Figure 4.4 we have a direct matching between a configuration and its angle error and energy. If we consider a fixed accuracy constraint we can find on this figure one

Table 4.4: Number of operations for one iteration of Extended Kalman Filter for each bitwidth parameter

Parameter	ADD	MULT	DIV
Default bitwidth	3039	3141	0
Quaternion bitwidth	17	12	0
Extented inversion bitwidth	0	343	6
Extended quaternion normalizetion bitwidth	32	88	0

configuration which has the minimum energy consumption for this accuracy. Similarly for a given energy consumption we can find a configuration which provides the best accuracy. These configurations are the optimal configurations considering the two criteria of accuracy and energy consumption. This collection of configurations is the Pareto frontier, which in the Figure are blue crosses that are on the lower edge of the cloud of blue crosses. We encircled in red some configurations on the Pareto frontier that have been selected for implementation.

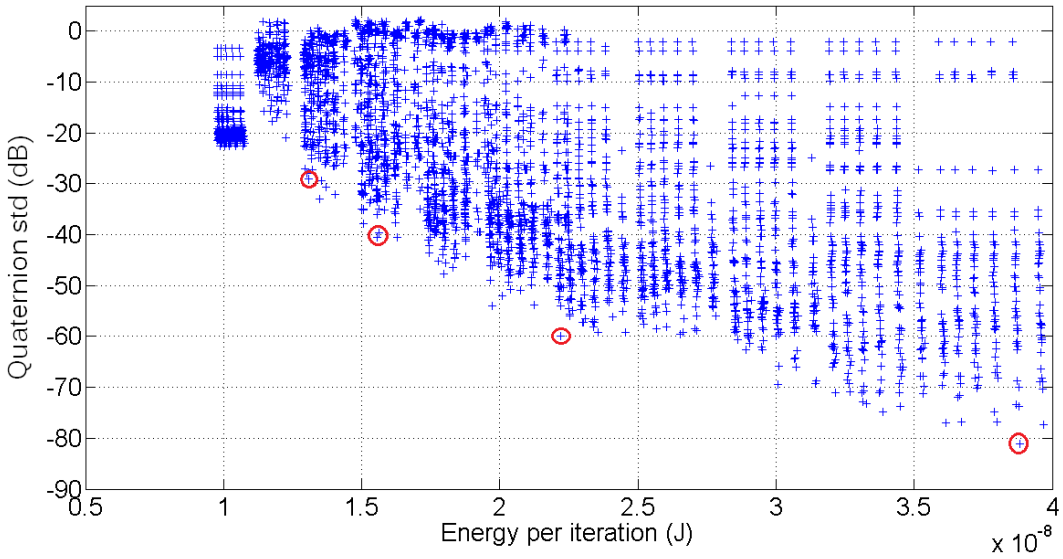


Figure 4.5: Accuracy energy tradeoff for a set of tested bitwidth configurations of the EKF.

We select several configurations corresponding to different accuracy levels. The first configuration is the Medium accuracy of $-40dB$, which is the accuracy of the static case (about 1°). The second level is High ($-60dB$) (about 0.1°), which is accuracy of high quality systems like [22]. We also select a Very High accuracy level ($-80dB$). The last configuration is Low accuracy ($-30dB$), which is the dynamic accuracy configuration. Table 4.5 gives the bitwidth values of each parameter for each configurations selected. We can see that for all configurations the extended inversion bitwidth is not really bigger than the default size since

we need $1bit$ or $2bits$ more. This is a good sign that stability of the filter is not very sensitive to our implementation of the matrix inversion. The same observation can be done for the extended quaternion normalization bitwidth and quaternion bitwidth, at most only $4bits$ are required to provide the accuracy. If we now compare all the bitwidths for different accuracy, we see that when we go from an accuracy configuration to the just below configuration we decrease the bitwidth of each parameter. As an example, the transition from VH to H is a decrease of $20dB$ of quaternion accuracy, which is a magnitude of order on quaternion value. The decrease of bitwidth between these two configurations is about $3bits$ for each parameter, which seems logic since we need to approximately $3bits$ to represent one magnitude of order in fixed-point data type. The same observations can be made for the other configuration transition. Lastly, we have to mention that the relative difference of bitwidth between of the VH configuration and the M or L configurations is big, $7bits$ compared to $29bits$ represent 24%. This means that the accuracy of the targeted application will have a real impact on energy consumption.

Table 4.5: Configuration selected for accuracy

Accuracy Bitwidth	Default	Quaternion	Ext. inv.	Ext. Quat.
Very High (-80 dB)	29	20	30	22
High (-60 dB)	26	17	26	19
Medium (-40 dB)	23	16	23	16
Low (-30 dB)	22	12	24	16

4.5 Architecture Exploration using HLS

4.5.1 HLS Tool

In this section we briefly explain how the HLS tool used addresses architectural issues to translate C++ code into RTL. The HLS tool we use is Vivado CatapultC.

RTL code generated by HLS is a description of the data-path, composed of wires and registers, and the basic operators to be instantiated. The basic operators are adder, multiplier, divider, multiplexer, etc. At this level of description, data transition through wires are considered perfect so that data need one clock cycle to reach the next register. The RTL takes into account the number of cycles an operator takes to deliver data but it does not specify how operators are technologically implemented.

In most cases, it is easy to translate a C++ instruction into RTL description, specially if the instruction is a basic operation as adding, multiplying, etc. But for a complex algorithm implementation like the EKF, the two following issues, among others, have to be addressed: 1) How to manage concurrent computing? 2) How to describe the hierarchy of an architecture?

Indeed, in C++ the concurrent computing is done using multi-threading. However this solution is not supported by HLS tool since it needs a consistent method to ensure the

behavior of the architecture for each clock cycle. Also, in C++ a function is an abstract object that can be used with a lot of flexibility, thinking of recursive functions that are impossible to implement as such with real object. If a function is used twice, do we implement two instances or do we reuse the same instance twice? The C++ does not need to solve this statement, so in HLS we need to adjust the coding style to control the way the code will be interpreted.

The solution proposed by Vivado is in two parts: the restriction of the coding style to functions and the use of blocking FIFOs as data channels. Each function must be, either composed of a hierarchical description that links functional blocks, or only composed of operation instructions. This allows to describe the block hierarchy of the architecture with no ambiguity. On the other hand, the use of blocking FIFOs allows to fully control the transfer of data between blocks. Blocking FIFOs means that, if a block needs a data from a particular FIFOs, it interrupts the process and automatically waits until the data is available.

Moreover, each blocking FIFOs is connected to only one input and one output. This way, each C++ function that produces and consumes data from FIFOs is considered as a hierarchical block. Then, the scheduling is reduced to a producer-consumer problem.

That is said, HLS tools are particularly efficient for the implementation of combinatorial circuits and to optimize loops. However, this efficiency has to be employed by an adapted source code. This requires to split the algorithm into function blocks, which will be separately optimized.

4.5.2 Hierarchical choices

The algorithm is split into functions, some are generic to EKF as matrix product or matrix inversion, and some are specific to our model as update equations or quaternion normalization. Each function is implemented in a separated block, then it is possible to establish a hierarchical structure for our architecture. If a function is used several times, the question of multiple instances of the block must be addressed. It is up to the designer to choose if a function call needs a new block instance. We have opted for the two extreme opposite approaches. The first configuration is a straight-forward data-flow, functional blocks are instantiated multiple times, one for each use. The second configuration implements a single instance of each functional block and manages the data-path to reuse blocks to provide the same results.

4.5.2.1 No-Reuse Architecture

The *No-Reuse* architecture has the advantage to allow concurrency and to be relatively easy to develop. This solution is more costly in term of area but has potentially a lower latency. This version of the EKF is not presented, but the block partition of the algorithm is the same as given in Section 4.5.2.2.

4.5.2.2 Reuse Architecture

In the *Reuse* architecture only, one instance of each functional block is used. This solution relies on a custom local FSM that controls the data-path. Figure 4.6 shows the EKF architecture that corresponds to this case. Note that the local FSM is developed by the designer and not generated by the tool.

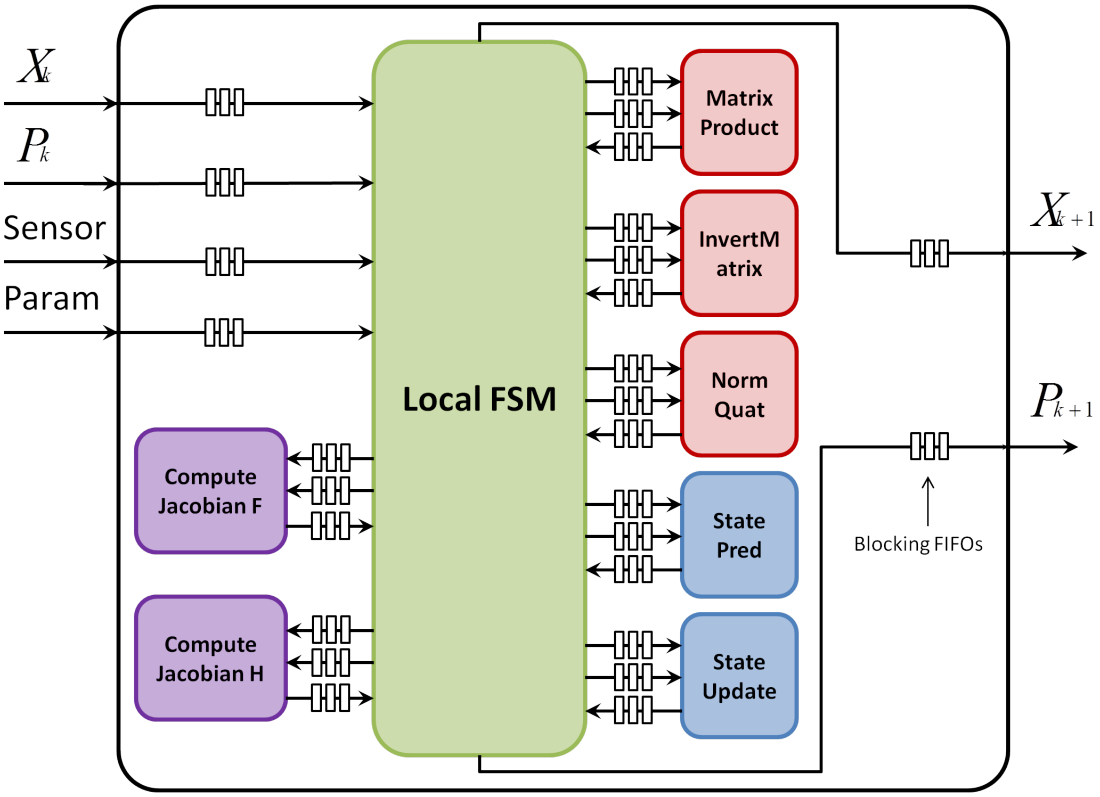


Figure 4.6: EKF architecture with sharing blocks (Reuse)

Blue blocks are units that provide Kalman-level functions as state prediction and state update. They correspond to Equations (3.19)-(3.21) and (3.22)-(3.23) detailed in Section 4.2.

Purple blocks are units that compute the Jacobian matrix of prediction and update functions, F and H.

Red blocks are units that provide arithmetical functions as matrix product or matrix inversion.

Green block is the FSM that manages the data-path. States of this FSM correspond to block-level scheduling.

With this partitioning of functions, it would be possible to adapt source code to implement a different EKF version with a minimum of architecture modifications.

4.5.3 Matrix Product

The matrix product is a central block of the EKF. Here we explore some synthesis options of the HLS tool to determine the best tradeoff between all configurations.

As said previously, the HLS tool provides RTL which is a functional description that do not specify the implementation. But, at the next step of development, the RTL will be transformed into Gate level, which is a description at logical level. So, the HLS tool has to take into account the characteristic of the targeted technology. The main characteristic used is the critical path, which is the longest time for a data to transition from a register to the next. For a given clock frequency, the HLS tool schedules operations adding register when critical path becomes bigger than the clock period. The critical path of all operators for a particular technology are collected in a library corresponding to the targeted technology. Each library also contains other characteristics for each operator like the number of cycles needed and the presumed area at implementation. In fact, each operator has several versions with different tradeoff between the critical path, the number of cycles and the area. This allows the HLS tool to choose operators according to different constraints defined by the user. The RTL generated by the HLS tool, that describes an architecture, is characterized by the number cycles it takes to produce data and its area.

In our case we try different implementation configurations for the matrix product. We use a straight-forward specification with 3 nested loops. HLS tools offer two main options to optimize loops: Unrolling and Pipelining. For clarity purpose we use a notation using three letters to call a configuration. The position of the letter represents the loop depth and the letter the option used N (Nothing), U (Unroll) or P (Pipeline). The default configuration is NNN. The clock frequency considered is 100MHz and target library is $65nmBULKCMOS$. Table 4.6 summarizes the results for the tested configurations. The area is given in μm^2.

Table 4.6: Matrix product option exploration.

Configuration	**Area (μm^2)**	**Cycle**
UUP	619k	111
NUP	285k	153
NNP	209k	453
NNN	209k	453
NNU	216k	447
UNN	326k	447
UUN	216k	447
NPP	216k	405
PPP	209k	399

The basic reference architecture is the NNN configuration, there is no loop optimization, only one Multiply And Accumulate (MAC) operator is instantiated. This configuration is

the biggest in terms of number of cycles with 453 cycles, but also the smallest in terms of area with 209kμm^2. Other configurations have to be shorter in cycles but with a cost by increasing the area. However, we can see that all others configurations do not necessary provide an improvement. NNP, NNU, UUN and UNN configurations have a number of cycles that is similar to the NNN configuration, so they are not selected. NPP and PPP configurations have a small improvement on number of cycles around 10%, with a very little cost on area. However, these solutions have not been selected neither. The reason is that the cycles gain is small and we prefer to keep the unchanged NNN configuration as a clear reference. The last two configurations NUP and UUP are selected since they provide good tradeoff on cycles and area. NUP configuration provides a number of cycles divided by three for an increase on area of 50% compared to NNN. UUP configuration provides a number of cycles divided by four for an increase on area of 200% compared to NNN. The 3 selected configurations are UUP, NUP and NNN and each of them provides a different Area/Latency tradeoff.

4.5.4 Early EKF Implementation Results

Here we present results of the synthesis of the 6 different implementations of our EKF. The 6 configurations are given by the 2 architecture alternatives (*No-Reuse* and *Reuse*) combined with the 3 matrix product implementation possibilities (NNN, NUP and UUP). All these results are made considering the Very High accuracy case, which has the strongest time constraint. The clock frequency is set to $100MHz$ (10ns critical path). The critical path of all configurations tested at this step are about $9ns$, which slightly corresponds to the clock period.

Table 4.7 summarizes results in terms of area and number of clock cycles for the 6 configurations. Note that the area is not representative of real results since the tool uses a standard library. As expected the Reuse configurations are smaller than the No-Reuse, which seems natural. However the number of cycles are similar considering the same matrix configuration. This is due to the fact that the EKF equations have dependency that does not allow to parallelize computations. Moreover, the number of cycles reduction of the total architecture using NUP and NUP is significant, around half, for a really small cost on area around 10%.

Table 4.7: Synthesis results Kalman (accuracy VH)

Configuration	**Area**	**Cycle**
No-Reuse/UUP	8121k	4111
No-Reuse/NUP	4940k	4489
No-Reuse/NNN	4243k	7189
Reuse/UUP	2834k	4092
Reuse/NUP	2481k	4470
Reuse/NNN	2403k	7170

The next section will provide more accurate results with gate level simulations of these 6 configurations.

4.6 Gate-level EKF Implementation Results

4.6.1 Architecture selection

Due to the simulation time and memory space required at this level, before exploring different technological possibilities, we select the most promising architecture from the 6 available candidates.

Here we set the clock at 100MHz and use the 65nm technology library low-power with low V_T transistor and $1V$ supply. The design is compiled at Gate level by Design Compiler and simulated by ModelSim. Power consumption is evaluated using PrimeTime.

Table 4.8 provides area results for the 6 configurations presented in the previous section, with a description at Gate-level. We can see that these results are different from HLS tool

ones. Again, solutions with Reuse of hardware are far better in terms of area than No-Reuse one.

Table 4.8: Area results at Gate-level (accuracy VH) in μm^2

Config. Matrix	**Reuse**	**No-Reuse**
UUP	1679k	7833k
NUP	1200k	3565k
NNN	1118k	2728k

Table 4.9 provides power results for the 6 configurations presented in the previous section. The total power is composed of the static power and the dynamic power. The static power is due to the supply of transistors, it is called the Leakage power. The dynamic power is due to energy dissipation during transition of state of transistors. We can see that for all configurations the leakage power is about 1% of the total power consumption. But, despite this low proportion, the absolute leakage power is bigger than the power budget objective of $100\mu W$. This is why we intend to use the power gating to switch off the supply of the hardware during inactivity time. We also give in this table the energy consumed for 1 iteration of the EKF by multiplying the total power by the operating time of an iteration.

Table 4.9: Power results at Gate-level (accuracy VH, 65LPLVT, 100MHz, 1.00V).

Configuration	**Leakage**	**Total**	**Energy of 1 iteration**
No-Reuse/UUP	$2233\mu W$	$112mW$	$4.48\mu J$
No-Reuse/NUP	$1167\mu W$	$95.8mW$	$4.31\mu J$
No-Reuse/NNN	$965\mu W$	$90.1mW$	$6.76\mu J$
Reuse/UUP	$617\mu W$	$50.8mW$	$3.05\mu J$
Reuse/NUP	$495\mu W$	$49.1mW$	$2.95\mu J$
Reuse/NNN	$473\mu W$	$46.1mW$	$4.15\mu J$

As expected, the Reuse solutions are more power efficient than the No-Reuse one. This can be explained by the fact that EKF equations do not allow a lot of parallelism. The lowest energy consumption is the Reuse with NUP configuration. It is interesting to see that the most energy efficient solution is the intermediate tradeoff between area and execution time. The solution Reuse/NUP is selected and will be explored in the following section using different technological implementations.

4.6.2 Technology exploration

4.6.2.1 CMOS65nm

Here we present the results of the selected configuration Reuse/NUP for a set of 3 different accuracy levels (Very High, High and Medium) which have been defined in Table 4.5 of Section 4.4.2 and two 65nm technologies. The difference between the two technologies presented is the V_T voltage, which is High or Low. Higher V_T means lower switching rate, so longer critical path, but a decrease of the leakage power. The clock is still $100MHz$.

Table 4.10 summarizes the area and power results for the 2 65nm technologies and for the 3 accuracy configuration. Results of the 2 technologies are similar. The V_T voltage has a small impact on area and total power consumption for our architecture.

Table 4.10: Area and Power results (Reuse/NUP/65nm/1.00V/100MHz).

Config.	Area	Leakage	Total	Energy of 1 iter.
VH/HVT	1258k	2.61µW	48.3mW	2.90µJ
H/HVT	1081k	2.32µW	43.4mW	2.60µJ
M/HVT	961k	1.99µW	38.4mW	2.31µJ
VH/LVT	1200k	495µW	49.1mW	2.95µJ
H/LVT	1040k	436µW	44.2mW	2.64µJ
M/LVT	925k	368µW	39.2mW	2.35µJ

On the other hand, accuracy has a large effect on area and power. Table 4.11 compares area, power and bitwidth ratio between all the 4 accuracy configurations considering Medium (M) accuracy as the reference. The bitwidth column is computed using Default bitwidth of Table 4.5 since we saw previously that it represents 90% of the operation load. We can see a strong correlation, almost linear, between the 3 parameters area, power and bitwidth. The values of the L configuration is not the result of simulation but extrapolation. However, the correlation of parameters allows us to assume that the reduction cost of the Low configuration should be about 5%. These results show us that the increase of accuracy is high compared to the increase in power and area. Indeed, increasing the accuracy by $40dB$ only cost an increase of around 30% in power and area.

Table 4.11: Increase ratio between accuracy configurations.

Accuracy	HVT		LVT		Bit-width
	Area	Power	Area	Power	
VH	30.9%	25.8%	29.7%	25.6%	26.1%
H	12.5%	12.8%	12.4%	12.8%	13.0%
M	−−	−−	−−	−−	−−
L[1]	−5%	−5%	−5%	−5%	−5%

[1]Low configuration is linearly extrapolated from the 3 other configurations

We can also see that all of the 6 configurations presented exceed the power budget objective of 100µW. Indeed, considering an iteration rate of 50Hz, the maximum energy allowed for an iteration is 2nJ, which is exceeded by the 6 configurations. That is the reason the motivate the use of a different technology in the following section.

4.6.2.2 FD-SOI28nm

Here we use 28nm FD-SOI technology. The RTL specification produced by the HLS tool is the same as for 65nm. In a first version we keep similar parameters than for the 65nm results, 1.00V Supply and 100MHz clock. In a second version we try to take advantage of the 28nm technology. Firstly we decrease the supply voltage to 0.75V. The second point is the tuning of the clock frequency. We increase the clock frequency to minimize the activity time and so the leakage power consumption, considering the use of power gating. Considering a safety margin, we have chosen a clock period of 6ns, so a frequency of 167MHz. Table 4.12 summarizes the area and power results for the two 28nm versions $1.00V/100MHz$ and $0.75V/167MHz$ and for the 3 accuracy configurations. We also give the critical path of configurations to ensure that it is lower than the $6ns$ period.

Table 4.12: Area and power results (Reuse/NUP/28nm).

Config.			Area	Crit. path	Leakage	Total	Ener./iter.
1.00 V	100 MHz	VH	334k	3.15ns	497µW	30.6mW	1.84µJ
		H	289k	3.11ns	437µW	26.9mW	1.61µJ
		M	256k	2.71ns	380µW	23.8mW	1.43µJ
0.75 V	167 MHz	VH	321k	5.74ns	96.4µW	27.0mW	1.01µJ
		H	278k	5.68ns	84.3µW	24.6mW	0.886µJ
		M	246k	4.88ns	74.1µW	21.8mW	0.784µJ

First we note that the area and power ratio between the accuracy configurations is the same than for 65nm, around 15% between M and H and 30% between M and VH, Secondly the area is about quarter of 65nm results. We observe that in the first configuration ($1.00V/100MHz$) the energy decrease is real but not tremendous since we don't take advantage of the opportunity to reduce the supply voltage, which has quadratic impact on Power. In the second configuration ($0.75V/167MHz$) the energy decrease is significant and around 45% as expected from the supply decreases. With a frame rate of $50Hz$ and the constraint of 100µW, these results show that we are two times beyond of our objective of 2µJ per iteration. We have reached our objective with a margin large enough to compensate the expected overhead due to power gating.

4.7 Conclusion

In this chapter, we used a design-flow strategy that implies a top-down process, from application level down to technological implementation. The design is configured in a way

that the implementation choices like bitwidth match with the accuracy requirement of the recognition applications. Several solutions have been explored along the process and few have been selected.

Figure 4.7 shows the comparison between results of the implementation and the expected values. In this Figure we added the real values $0.78\mu J$ for $28nm$ and $2.35\mu J$ for $65nm$. We also readjusted the expected values for $28nm$ and $65nm$ considering the bitwidths. The EKF algorithm has also been implemented in floating-point in the Cortex M4 of our prototype Zyggie V2. The execution time and the current consumption has been measured. One iteration of the algorithm uses $56.8\mu J$. In these conditions only the $28nm$ matches our objectives and requires a radio bellow $21nJ$ per bit.

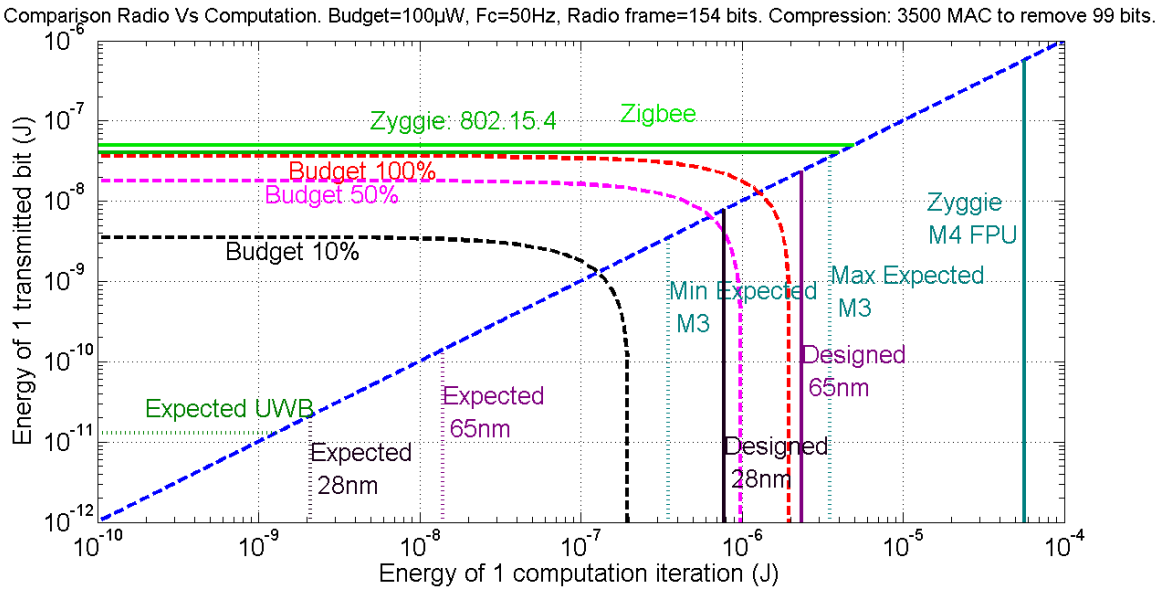


Figure 4.7: Comparison between Radio and implemented Computation results.

We also see on this figure that the energy of the implemented version is 300 times bigger than the lower bound for $28nm$, 150 times bigger for $65nm$ and 20 times bigger for the Cortex M4 version. That is the same for radio part.

These differences, are due to different aspects. First HLS does not provide the more optimal solution. Secondly, the solution is not fully custom and provide some flexibility. Finally, over a given number of MAC, control units, data transfers and additional logic are required to run the application. We do not have results of a dedicated radio implementation, but we can expect the same kind of gap for the radio. The ultimate IR-UWB bound corresponds to a high rate transmission of raw data considered at the physical level, so additional power consumption will emerge inevitably from the protocols layers with an impact that increase with low rate communication as in our BAN.

In the next chapter we focus on the aspect of adapting the configuration of the node to the motion. Indeed, here we only focus on the EKF consumption, next chapter will give an overview of the whole node consumption profile including sensors.

5

Adaptive system driven by motion activity

In this chapter we discuss and test algorithms that are required to manage computation modalities depending on the motion detection. This adaptation is motivated by the heavy energy constraint which does not allow to constantly use the full performance mode, and the possibility to adapt the performance mode to the real time needs.

5.1 System-Level management of the node

In this section we present our strategy to deal with the energy constraint for the whole node, this includes the sensors, the computing part, the radio and the memory. In the previous chapter we showed the possibility to implement an EKF that is compliant with energy budget. But currently, the use of MEMs sensors only is enough to burn out the energy budget. We also present here which decisions are made in terms of sensor rate and data rate.

5.1.1 The System-Level Finite State Machine

In previous chapters we developed algorithms that allow to produce applicative results as posture and gesture detection. We saw that there are a lot of configurations when we consider all the different applications. More than producing data the node has to consider all the high level decisions:

- Choose configuration for each type of application.
- Switch on/off and set the sampling rate of each sensor.
- Set the data production rate for the EKF/classification.
- Switch on/off the power gating of each function.

In our case we decided to implement this top management with a Finite State Machine (FSM). This FSM, depicted in Figure 5.1, is composed of the following basic states

- Activity detection.

- Sensors sampling.
- Compute applicative results.
- Process radio data.
- Configuration of the node.

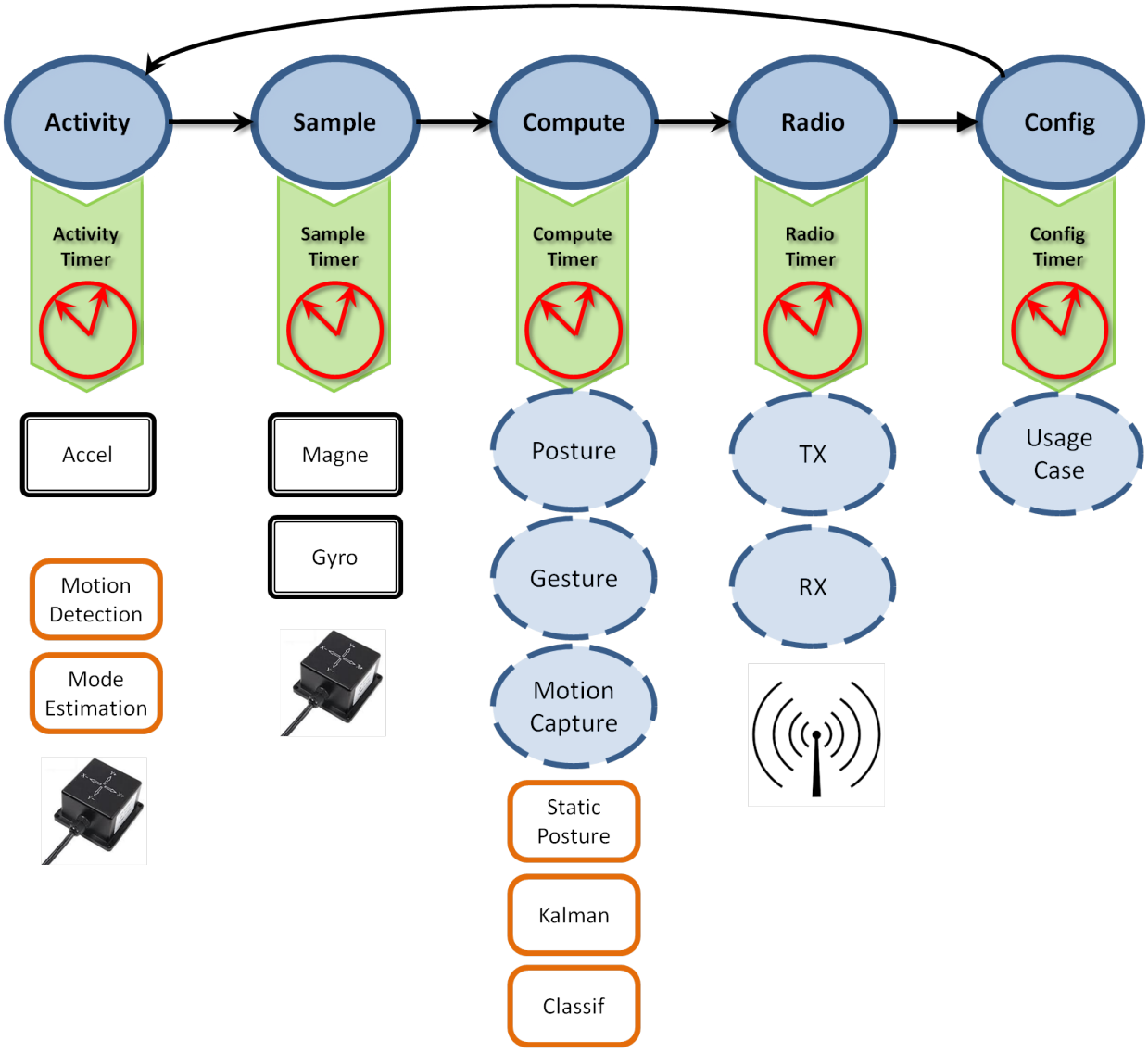


Figure 5.1: Top management Finite State Machine diagram.

Each state has its own timer which depends on the configuration. If a timer triggers, a bigger FSM of the corresponding state starts.

The activity FSM samples the accelerometer data and compute one iteration of the Motion detection and the Mode estimation. These two modules are addressed after in Section 5.2. The sample FSM samples data of magnetometer and, according to the configuration, can switch on/off or sample the gyroscope. The compute FSM fusions data given by MEMs, depending on the application, posture detection, gesture detection or motion capture. If static posture detection is required the static detection module and classification are used. If gesture recognition is required the EKF and classification are used. If the motion capture is required only the EKF is used. Result data are then written in memory waiting for radio to transmit. The radio FSM sends new data if there is one, and write in memory received data. The configuration FSM has two functions, interpret received data from radio related to usage configuration and manage timers in function of the activity mode.

Figure 5.1 is a representation of the top management FSM. Blue circles are basic states of the FSM. Most of the time, the node is on idle in top management FSM which power consumption is insignificant. While the FSM is on one of the basic states, all bottom FSM are switched off using power gating. Different states communicate using shared memory, since data rate is low it does not need a sophisticated BUS policy.

The energy saving strategy is mainly determined by the activity state. The activity management is the part that decides the sampling rate and the computing rate of the node depending on the detected activity . It uses accelerometer data since it is a relevant sensor for motion detection and the least energy consumer.

5.1.2 Computing management

The sampling and computing rates determine the energy profile of the node. The biggest concern is the gyroscope sensor. We showed in section 2.1, Table 2.1 that the power consumption of this sensor outclasses the power budget of our system (about 10mW consumption for a 100µW budget). It also outclasses the power consumption of other parts of the system. Moreover the consumption is almost constant regardless of the output rate. That is why when the gyroscope is on, we have to maximize the benefits of the data it provides. Thus, we set the maximum performance mode considering the sampling rate and computing rate of 50Hz and the gyroscope on.

When the gyroscope is off, the energy consumption mainly scales on the sampling rate of magnetometer and/or the radio frame rate (depending on the transmitter technology considered.). The accelerometer has a very low power consumption, it is always used at maximum rate to help as an activity detection.

This way, we choose a set of rates by considering the two extremes of the frame rate, the fastest 50Hz and the slowest 1Hz which corresponds to a barely no motion state. The last rate is an intermediate set to 10Hz. Table 5.1 summarizes the sampling rate of each sensor and the corresponding computing rate.

Table 5.1: Sampling rate of each sensor and computing rate for each mode.

	Sensor sampling rate (Hz)			Computation rate (Hz)
	Accelerometer	**Magnetometer**	**Gyroscope**	
Mode 1	50	1	–	1
Mode 2	50	10	–	10
Mode 3	50	50	–	50
Mode 4	50	50	50	50

5.1.3 Energy management

Considering the parameter of each selected mode we can estimate the power profile of each mode. In our case we consider a node with ($1\mu J/iter$) for computing and a ($1nJ/bit$) for the radio.

The radio frame considered in our case is depicted in Figure 5.2. The communication policy for our estimation is the following, we receive a frame at 1Hz and we emit a radio frame at the same rate as the computing rate.

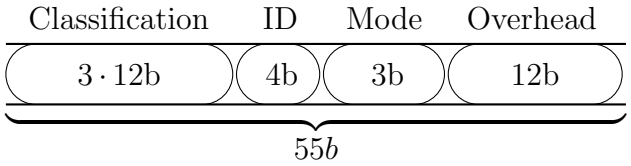


Figure 5.2: Radio frame considered composed of classification results, identification of nodes, the node configuration and an overhead.

Power estimation is based on the MEMs state of the art presented in Table 2.1.

Results are presented in Table 5.2. We use two indicators to present results, the power and the life-time. This is done because in case of power greedy mode the power does not make sense anymore. The life-time of the system is estimated considering a battery supply that should provide 16h (maximum daily use time considering 8h of inactivity) of use at the power budget of $100\mu W$.

Results show that for a day the maximum cumulative time of use of the gyroscope is about 6min. The adaptation of mode, dynamically considering motion and application needs, is clearly mandatory due to this constraint. We also see that for other modes the life-time is not really an issue with several hours.

Table 5.2: Power equivalence to the power budget of 100µW and life-time duration of each computing mode.

	Sensor			**Computing**	**Radio**	**Equivalent**	**Equivalent**
	Accel	**Magne**	**Gyro**	(1µ$J/iter$)	(1nJ/bit)	**Power budget**	**Life Time**
Mode 1	15µW	1.6µW	–	1µW	110nW	18%	90h
Mode 2	15µW	16µW	–	10µW	605nW	42%	38h
Mode 3	15µW	80µW	–	50µW	2.8µW	150%	11h
Mode 4	15µW	80µW	15mW	50µW	2.8µW	15100%	6min

5.1.4 Memory, data storage

Here we present separate results of the memory consumption.

We consider a single shared memory for all the node hardware. We have to characterize its area and power consumption. Firstly we list the sum of data we need to store:

- State variable X.
- Covariance variable P.
- Kalman configuration R, Q, $\vec{Acc}$, $\vec{Mag}$, dt.
- Sensors value $\vec{Acc}$, $\vec{Mag}$, $\vec{Gyr}$.
- Distances value and certitude.
- Classification matrix considering 3 eigenvalues.
- Result projection.
- Node configuration ID, mode, threshold.

The size of data and the expected memory is summarized in the table 5.3. We can see that the size of the memory is really small with at most 85 32b-words.

Then using the CACTI tool [60] we estimate the area of the memory, the energy consumption of a read/write and the power leakage. Estimations are done using 65nm and 32nm Low Standby Power (LSTP). Table 5.4 provides the area, the energy cost of a read/write and the leakage power of each configurations. Results in Table 5.4 show that the memory of our system is about 10% of the size of the EKF implementation, see Table 4.10 for 65nm and Table 4.12 for 28nm for comparison. Moreover the power consumption is negligible to other part of the system when we compare to magnitude of previous estimation in Table 5.2.

5.2 Activity detection

In this section we propose a simple solution to detect motion using accelerometer. It is then used to select computing mode of the node. Results are given using simulation with recorded

Table 5.3: Memory size necessary for algorithm storage considering different word size.

Type	**Size (bits)**	**Size (16b-words)**	**Size (32b-words)**
State variable X	$16 \cdot 7$	7	4
Covariance variable P	$23 \cdot (7 \cdot 7)$	$49 \cdot 2$	49
Kalman configuration	$23 \cdot (3+3+2+1)$	$9 \cdot 2$	9
Sensors value	$16 \cdot (3+3+3)$	9	6
Dist. value & certitude	$12 \cdot 10 + 4 \cdot 10$	10	5
Classification matrix	$16 \cdot (3 \cdot 4)$	12	6
Result projection	$16 \cdot 3$	3	2
Node configuration	$5 + 3 + 10 \cdot 3$	1+3	4
Total	$2028bits$	159	85

Table 5.4: Memory characteristic estimation using CACTI [60] for 65nm and 32nm Low Standby Power (LSTP).

Techno	**16b-words**			**32b-words**		
1.0V	Area	R/W Energy	Leakage	Area	R/W Energy	Leakage
65-LSTP	11051µm^2	1.6pJ	15.1nW	14290µm^2	2.2pJ	14.1nW
32-LSTP	2692µm^2	0.45pJ	14.5nW	3474µm^2	0.63pJ	13.2nW

gestures. Our objective is to implement a very simple and efficient motion detection that allows us to demonstrate the interest of dynamically change the computing mode of the node. Our solution matches the objective but other implementations may proposed.

5.2.1 Motion estimator

The activity management is done in two steps, the motion detection and the computing mode selection.

The goal of the motion estimator is to use the 3-axis accelerometer data to estimate a motion value that should reflect the compute need. Then using threshold we can determine the computation mode required.

The orientation of the node defines the projection of the earth gravity vector in the sensor frame. A rotation changes the projection of gravity over axis which is detectable. The only exception is the pure rotation along the vertical axis which passes through the sensor, but we can assume that this case is unlikely to happen in case of a node on a body. This means that pure rotation can easily be detected be a 3-axis accelerometer. Most of the time, any motion is composed of rotation and translation. A translation produces linear acceleration which is added to gravity. The linear acceleration does not imply a variation of orientation, but we can assume that this value is relevant of a motion velocity and so the potential need of a high computing rate. The special case of uniform motion is not an issue since first, a uniform motion is unlikely to happen a node on a body, and secondly it does not change the orientation anyway. Therefore, we can consider that the accelerometer is enough to detect

any motion with value that can reflect several level of activity.

The motion estimator is basically a digital filter. The output must be zero when the node is static, the norm must increase during a motion and must decrease to zero when back to static state. This means that, the filter has to remove the continuous component of the signal which is gravity when static, the filter has also to detect the acceleration variation which are both the linear acceleration and the change of gravity projection over axis. Explained that way, the filter has an high-pass behavior. We need to implement a simple filter so we choose a first order Infinite Impulse Response (IIR) filter. The equation of the filter is given by 5.1. y is the output of the filter (the motion estimation) and x is the input of the filter (accelerometer values).

$$y_{n+1} = y_n \cdot a_1 + x_n \cdot a_2 + x_{n-1} \cdot a_3 \tag{5.1}$$
$$0 < a_1 < 1 \tag{5.2}$$
$$0 < a_2 < 1 \tag{5.3}$$
$$-1 < a_3 < 0 \tag{5.4}$$

The setting of coefficient is not straightforward. We have to detect motions but what is considered as a motion for a human and what is a static case? Our system runs at the minimum frequency of 1Hz, motions under this frequency can be tracked using accelerometer and magnetometer. Above this frequency the sampling and computing rates must increase and the gyroscope may be necessary.

Considering human motion we came out with the filter coefficients 5.5.

$$y_{n+1} = y_n \cdot 0.65 + x_n \cdot 0.9935 - x_{n-1} \cdot 0.99 \tag{5.5}$$

The filter characterization is given in Figure 5.3. We see in this figure that the filter has a behavior close to a first order high-pass with a cut frequency about 0.1*Fe (Fe is the sampling frequency of accelerometer). The high frequencies are slightly amplified.

The choice of these coefficients is not optimum for all motions but still provides satisfying results in our experiments. The idea to set coefficients is to consider timing constraints. After a stimulation the time to return to rest state is estimated by the time constant $\tau = \frac{1}{2\pi f_c}$. With the approximation of a first order filter, the time to return to rest state at 95% is $\tau_{95\%} = 3\tau = 0.095$s at 50Hz (0.16s at 30Hz). This is coherent with the human scale motion. Moreover, when the node is static, the accelerometer value can be at most gravity which is about $g \approx 9.81 m/s^2$. The output converges to $y_\infty = g/100 \approx 0.0981$ which is a value about the noise level of accelerometer MEMS sensors.

This filter is used for each of the 3-axis separately. The norm of the 3-axis are added so it gives a single positive value. This allows to use a simple threshold for the computing mode selection.

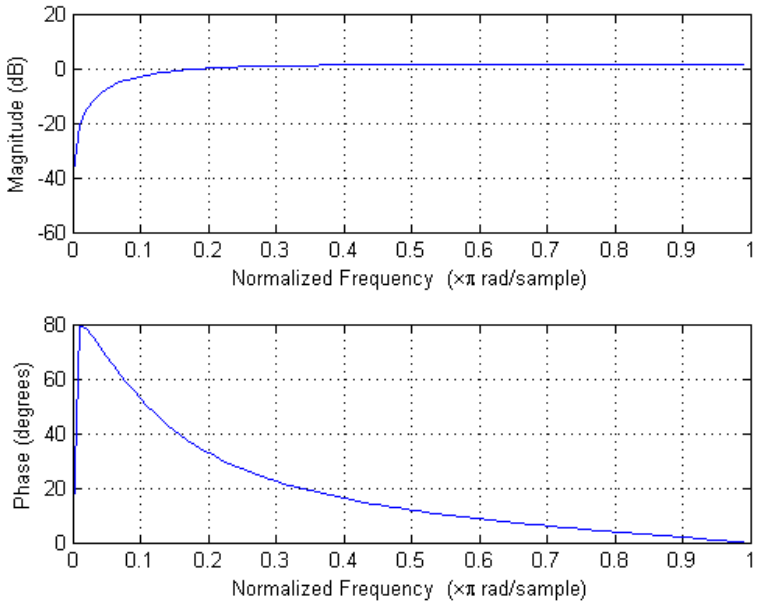


Figure 5.3: Baud diagram of the digital filter.

5.2.2 Computing Mode selection

5.2.2.1 Method

Thresholds are bounds that delimit the 'amount of motion' that reflect the need to change the computing mode.

There are several way to determine the values of the thresholds. The thresholds can be fixed or be computed dynamically.

The dynamic solution seems to be more adapted to our kind of applications because.

- The threshold value can depend on the node considered, hand, foot, chest, etc...
- The threshold value can depend on motion and user application.
- The motion activity is quantitatively subjective and dependent of the motion estimator, so thresholds can only be determined empirically.

The methodology is to use a policy to adjust threshold value in function of a feedback of orientation accuracy. Here the idea would be to use the EKF Error as a feedback which is an assumption of accuracy. This dynamic thresholds method looks like a closed loop control system. Figure 5.4 shows a schematic of how it should work.

But here the EKF feedback is not a direct read of the orientation accuracy but an assumption. Orientation error of EKF can be caused by any combination of :

- The maximum computing rate is too low compared to motion. That should not happen with standard human motions.

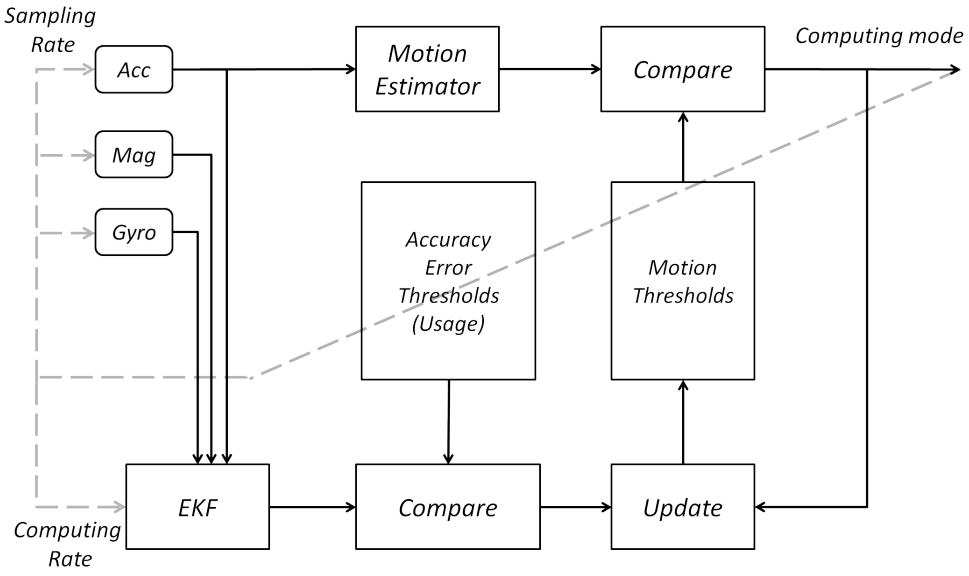


Figure 5.4: Synoptic view of the dynamic threshold method.

- The gyroscope is switched off so prediction state of EKF is wrong. With slow motions this error is low so should be corrected by the update state. This is here that thresholds values are important to ensure a good tracking.

- The motion added a linear acceleration which is confused with gravity. Here the EKF can detect an error while there is not necessary a error on the orientation estimation. A solution to suppress this error is to use the feedback of another reference system like a GPS that gives the linear acceleration.

The conclusion is that when the EKF estimates there is no significant error, it is almost always true. But the error it estimates do not generally reflect the orientation error. In fact, this method is an open loop system and the use of usual linear systems should not provide satisfying results This problem is more likely to be addressed by learning systems. That is why in our case we decided to no implement the dynamic threshold methodology. Instead we determine empirically fixed values of thresholds. The goal is to give results that show that even very simple implementation provides acceptable results for the application considered.

5.2.2.2 Determining threshold

We consider two ways to determine thresholds:

- A method should be to use records of existing motions. Simulations of the motion can be made while tuning thresholds values until the reach of an optimum. This method is application specific which has the advantage to be adapted to a given context, for each node on-body location and then more accurate.

- Another method is to use generic motions to keep generality. Firstly we consider the accuracy of the EKF for each computing mode for a range of generic motions to determine the limit of each mode. Then using motion estimator, we deduct the thresholds that delimit the computing modes. This method is not specific to an application but we just have to do it once.

We can see here that it is not reasonable to make an exhaustive study. There are too much parameters that can be changed

- The number of nodes and their on-body location.
- The application tolerance to gesture recognition error.
- The motion tested, recorded or artificial.
- The minimal or average orientation accuracy needed to reach in function of the application.
- The thresholds values.

That is why we decide to choose the generic approach to remove almost all application specific parameters.

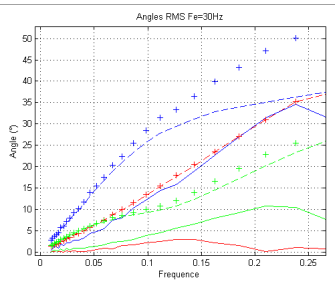
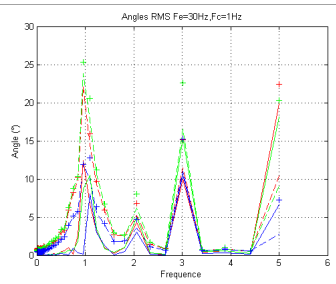
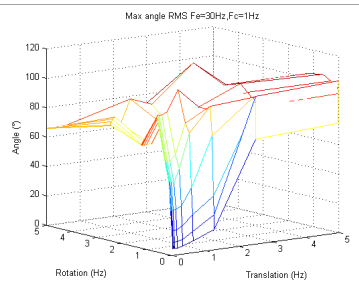
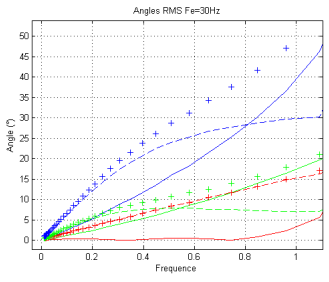
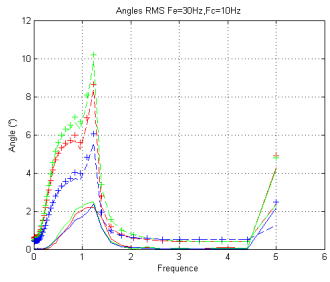
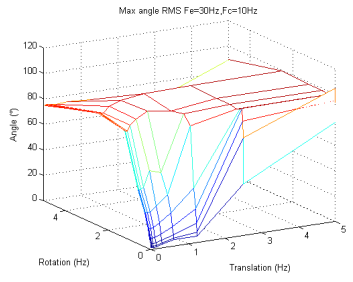
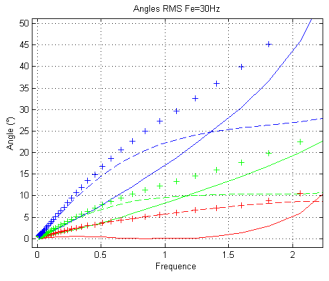
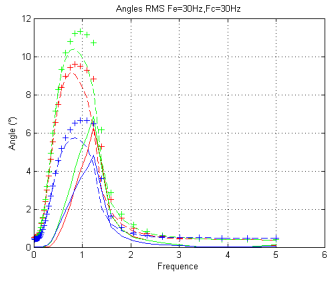
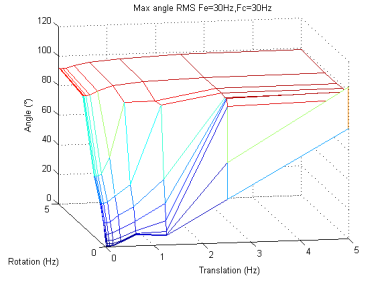
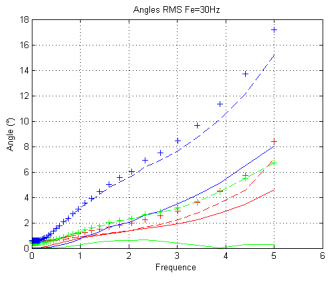
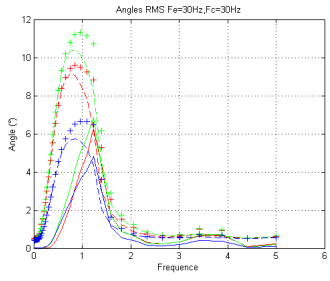
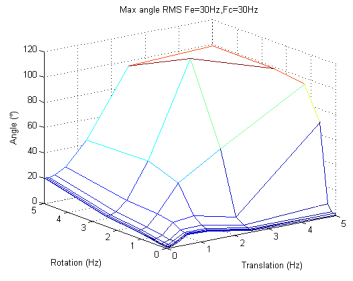
The methodology we use to determine thresholds values is, first determine the maximum 'motion frequency' that each computing mode can support without significant error. Then secondly use these bound frequencies in the motion estimation to determine the expected value.

The term of 'motion frequency' is used because we compute error using artificial motion. We create a motion that is composed of a translation and a rotation each one at its own frequency. The translation is about 0.30m and rotation of each axis is 90°. The EKF is tested for a range of frequency from 0Hz to 5Hz for each computing mode. For each of the 3 angles Pitch, Roll and Yaw, statistics on error are computed.

Table 5.5 collects results on the angles error. Each row represents the error for the corresponding computing mode, see Table 5.1. The two first columns of figures correspond to pure rotation and pure translation. For these two columns Pitch is in red, Roll is in green and Yaw is in blue. Full lines are the average error, dotted lines are the standard deviation and cross are the Root Mean Square (RMS) value. The last column corresponds to motion that are combination of both rotation and translation, for each couple of frequency the plotted error is the worst RMS of the 3 angles. For pure rotation, each computing mode has its own range of efficiency. Error seems almost linear with a different rate for each mode.

For pure translation, all modes seems to have the same behavior. In this situation what we can see is the effect of linear translation on angle error while the real orientation of is constant. This error increases until about 1Hz and then decreases. The increases is due as said in previous section that gravity and linear acceleration added are interpreted as only gravity by the EKF. The decrease is the results of the modification added by 3.31. If the linear acceleration norm is too high the confidence in accelerometer decreases then orientation error is removed. It works particularly well in the case of pure translation.

Table 5.5: Simulation for different motions (rotation, translation, both) at each of the 4 computing mode. In the two first columns: angle error average (full line), STD (dotted line) and RMS (cross) for Pitch in red, Roll in green and Yaw in blue. Last column: worst RMS among 3 angles for both translation and rotation motion.

Mode	Rotation	Translation	Combination
1			
2			
3			
4			

With the combination of the two motions, we can clearly see that each mode has its own range of efficiency. We can also note the huge difference between mode 3 and 4 which the only difference is the use of gyroscope or not.. One can argue that the error with gyroscope is too high for an AHRS. This would be true if this EKF was designed to always work at maximum speed with gyroscope. But here, the EKF parameters are tuned to give more confidence to update state that it should be. The reason is that we intend to use sparsely the gyroscope, so there is no prediction state most of the time.

Using these results, we extract the bound motion frequencies between modes. We use the pure rotation results and the RMS value of the 3 angles. We fixed the limit accuracy to 20° on average for the 3 axis. This accuracy value determines the tolerance of the system to orientation error and can be changed if we consider another application. Table 5.6 gives the extracted values of motion frequencies. For each of these frequencies the corresponding motion is given to the motion estimator. The thresholds are the average output values of each frequency.

Table 5.6: Motion frequency bounds and corresponding threshold value.

Motion Frequency	Threshold
0.1	2.3
0.65	6.5
1.4	8

5.2.3 Results of real time adaptation

In this section we collect results of the simulation of real time adaptation. We simulated a capoeira motion which is a case of a quick motion. The computing modes and the parameters of the activity estimation are set as determined in previous section. The idea is to compare results with the case of always using the maximum performance mode. We can compare the potential benefits in term of energy saving through the gyroscope and computing activity, and the cost in accuracy reduction.

5.2.3.1 Activity reduction benefits of using dynamic rates

Figure 5.5 shows the gyroscope usage rate during the Capoeira motion for each node. As we can see the gyroscope can be switched off most of the time for the majority of nodes, gyroscope is active around 30% of the time. Figure 5.6 shows the iteration rate usage compared to the full speed mode. It means the rate of iterations really performed relative to the maximum of iterations possible. Half of iteration are saved in our case. Refer to Table 5.7 to know the corresponding node of each value.

5.2.3.2 Accuracy reduction cost of using dynamic rates

To remind how the simulation works. Original motion is stored in a file as a list of orientations for each joints (22 here) at each frame (30Hz). The simulator creates sensors data

Table 5.7: Node names

Number	Name
1	Hips
2	Left Hip
3	Left Knee
4	Left Ankle
5	Left Toe
6	Right Hip
7	Right Knee
8	Right Ankle
9	Right Toe
10	Chest
11	Chest 2
12	Chest 3
13	Neck
14	Head
15	Left Collar
16	Left Shoulder
17	Left Elbow
18	Left Wrist
19	Right Collar
20	Right Shoulder
21	Right Elbow
22	Right Wrist

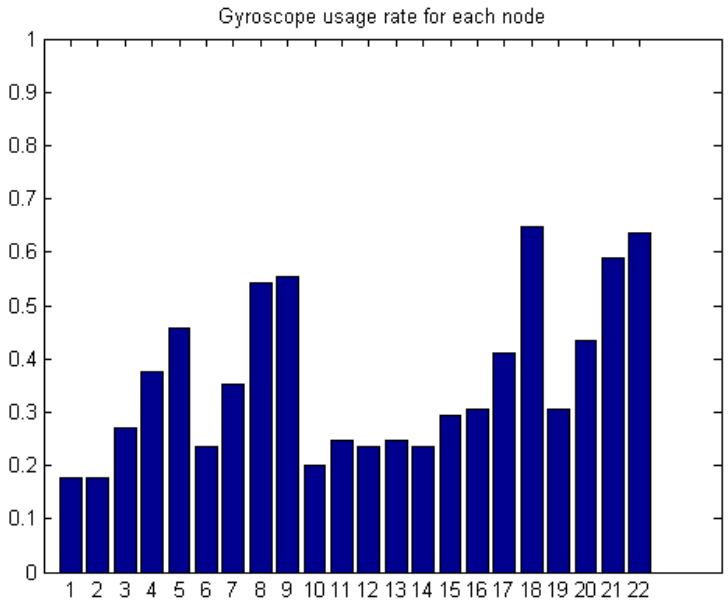


Figure 5.5: Gyroscope usage rate during a capoeira motion for each node.

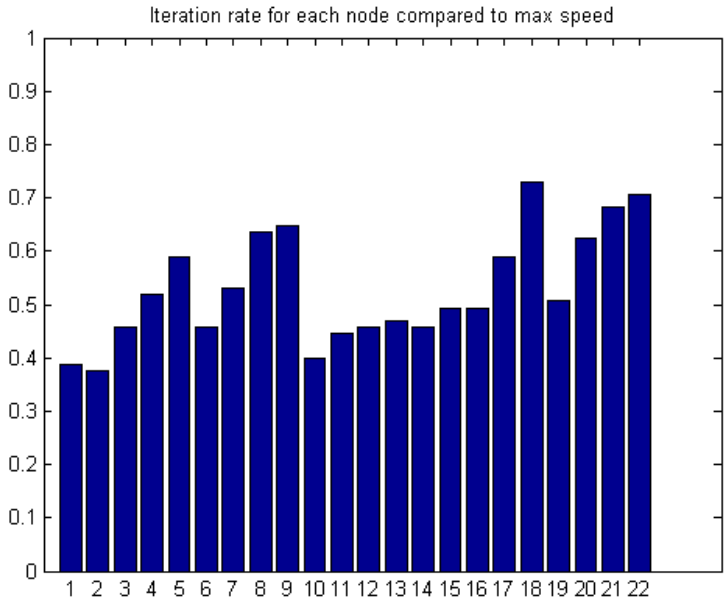


Figure 5.6: Iteration rate of Kalman computation during a capoeira motion for each node.

for each node that correspond to this scenario. Then we can test tracking of the EKF with custom configurations and compare results to original orientation of nodes. We also apply the gesture recognition to see the impact at application level. In this section we test two configurations, maximum performance with always gyroscope and the dynamic rate configuration as determined in previous section. Results of each of these configurations can be compared to the original or to each other.

To begin with, we compute the error of the angles estimation of each node using the RMS. Figure 5.7 show the error between the original orientation and the maximum performance estimation. This error is low, about 5° for almost all nodes.

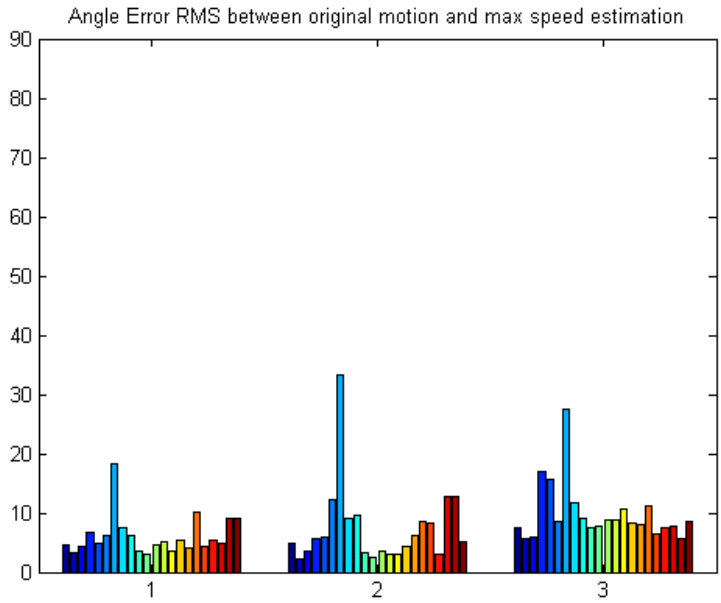


Figure 5.7: Angle Error RMS between the original motion and the maximum performance estimation for a capoeira motion for each node. Pitch graphic n°1, Roll graphic n°2, Yaw graphic n°3.

This value has to be compared to the Figure 5.8, which is the error between the original and the dynamic rate estimation. The error is almost doubled depending on nodes and angle. We can also see that the tracking fails for several nodes like 1 and 2. This is certainly due to the fact that all thresholds are the same for all node while the activity profile of all nodes are different. We expect that it could be partially solved using dynamic threshold.

The last Figure 5.9 compares the two estimations between each others. We can see that the profile of Figure 5.9 is similar to the Figure 5.8, but the average error is lower.

We see that angle error increases significantly with the use of dynamic rate. The error magnitude would be too high if we considered a motion capture application. In this case of application, we should reconsider the selection of computing mode and the thresholds to ensure better results and still benefits of activity detection. In our case, the gesture recognition is not as much affected by these errors.

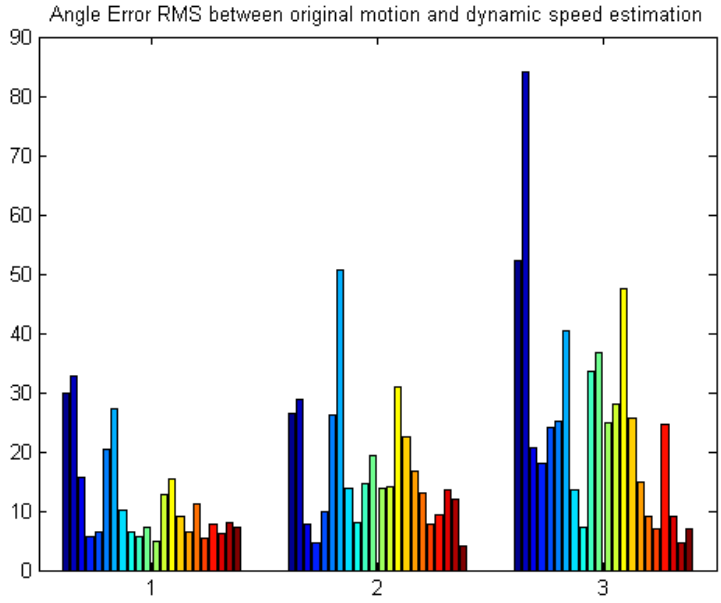


Figure 5.8: Angle Error RMS between the original motion and the dynamic speed estimation for a capoeira motion for each node. Pitch graphic n°1, Roll graphic n°2, Yaw graphic n°3.

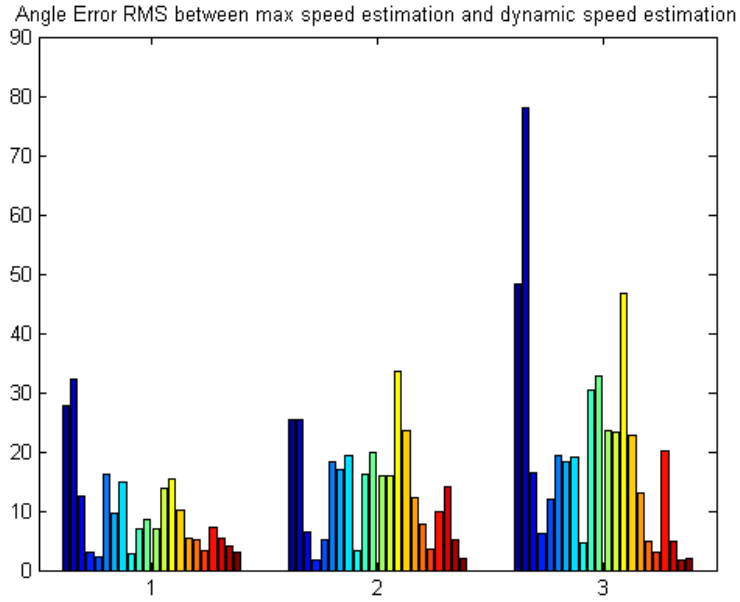


Figure 5.9: Angle Error RMS between the the maximum performance and the dynamic speed estimation for a capoeira motion for each node. Pitch graphic n°1, Roll graphic n°2, Yaw graphic n°3.

As explained in section 3.2.3, a gesture can be segmented in a set of reference postures that composes the motion. It is then possible to recognize a gesture by detecting this succession of postures. We now apply the same process as in section 3.4.3 with the new results that use dynamic rate.

The simulated experience is the gesture recognition of a capoeira motion using orientation of nodes. The nodes considered for these simulations are those located on ankles, knees, elbows, and wrists which are number 3,4,7,8,17,18, 21 and 22. The classification uses PCA and NCC method to decide the most likely posture among the set of reference postures. In previous section 3.4.1 we considered that the reference postures are the original posture orientation which are the correct data for a motion capture system. But in our case, the goal is not to recompose the gesture as the original but to be able to recognize the gesture. The idea is then to change the reference postures by using a record of our own system.

Figure 5.10 shows the gesture recognition profile of the motion considering the original orientation as reference postures. This figure has to be compared to Figure 3.21, we see that the dynamic rate decreases the accuracy of the recognition profile compared the solution with maximum rate and with gyroscope. Half of the frames are wrong,which, depending on application tolerance, can be prohibitive.

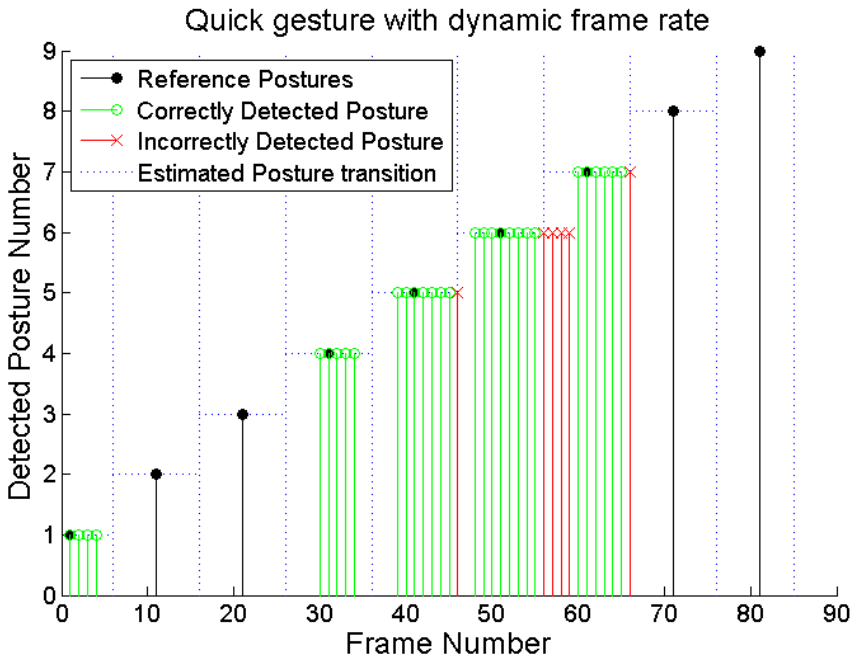


Figure 5.10: Gesture recognition profile of the a capoeira motion using classification. Data reference considered is the original motion.

However, if we consider that reference postures are the orientation estimation with the maximum performance record, see Figure 5.11, results are much better. Recognition profile is almost flawless in this condition. This means that even if the angle error introduced by dynamic rate increased significantly, the error at applicative level is not significant.

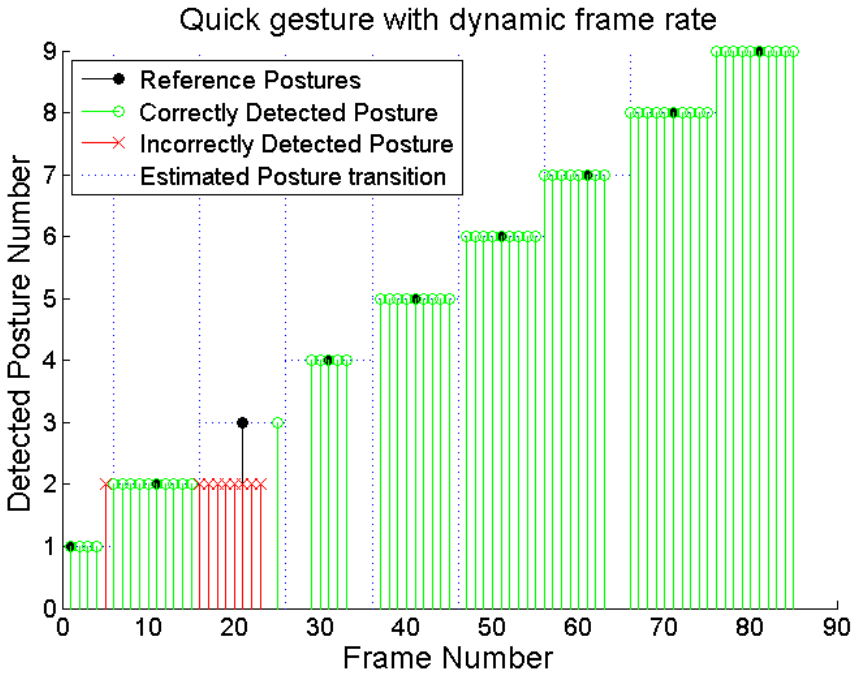


Figure 5.11: Gesture recognition profile of the a capoeira motion using classification. Data reference considered is the maximum performance estimation.

This result confirm the possibility to implement the rehabilitation application system. In a first step the correct rehabilitation gestures are recorded once with a patrician with the system at maximum performance speed . After that, the gesture recognition can be applied with dynamic rate which saves a lot of power and allows to monitor the exercise.

In real application we can imagine a way to set thresholds. When the user receive the new system it must be initialized. The user has to do some specific postures, gestures and motion that have been chosen to calibrate the system. This step can be used to determine several sets of thresholds depending on kind of activity. This way each system is calibrated for each user.

5.3 Conclusion

We showed the possibility to save energy using an activity detection system and dynamic rate. The cost in accuracy in our case doos not impact the application of gesture recognition. Our method could be adapted to different applications. We can change the current accuracy tolerance of 20° and the typical motion translation of 0.30m and rotation of 90° tested. The computing mode chosen can also be reworked to be adapted to specific applications. Our approach is generic but a study on the body activity could be done to identify the different computing needs according to the application and the on-body location of each node.

The importance of these results is to show the possibility to save a lot of energy using dynamic rate. The gyroscope usage rate is about 30% to 60% compared to maximum rate

which almost mean a extend in life time of a factor 1.7 to 3.3 only considering the motion sequence. In fact the real daily usage rate of the gyroscope could be a lot smaller. This impacts not only sensors and computing but also radio transmission.

Until a technological breakthrough happens with MEMs, adaptive rate is the only way to provide an outdoor/indoor (no equipped environment) gesture recognition system based on daily use (daily life time).

6 Conclusion and perspectives

6.1 Synthesis

6.1.1 Preliminary study

The BoWI ambition is to provide a generic system for BAN-based posture/gesture recognition. There are a lot of applications possible as games, MSD prevention, functional rehabilitation, etc.... We defined 3 generic applications that cover the different cases: the posture detection, the gesture detection and the motion capture.

Another aspect of the project is that the system must operate on a daily basis and with low ergonomic constraint. This means the absence of conventional battery and then an operating at ultra low power. Our objective is the possibility to use energy harvesting solution leading to the power budget of $100\mu W$.

After this, we made a study on the energy profile of all the components involved in the node architecture which are: the sensor part, the radio part and computing part. We can immediately note that all MEMs sensors, accelerometer, magnetometer and gyroscope, have different magnitude of power consumption (respectively $15\mu W$,$80\mu W$,$15mW$). This guides us toward an approach of using resources only when needed. The radio state of the art shows that it is not possible to use usual radio low-power transmitter with our power budget. The use of specific transmitter with IR-UWB energy efficiency is mandatory. Since it is not our expertise domain we did not focus on the radio transmitter, but we assume that transmitter, that compels with our requirements of power and bandwidth, will exist in a near future.

6.1.2 Simulation environment

In order to test our algorithms and different applications cases we use a simulator. This approach allows being fully in control of the data generated.

The simulator is based on motion capture files (bvh) to generate sensor data. This approach allows making some statistical analysis since we can compare results with original values while tuning parameters. Moreover, in a bvh file a gesture is a succession of postures. So we can construct a file that contains a library of postures and process it like static postures. Otherwise, using interpolation we can generate a gesture that links two postures. We can find a lot of website that freely share motion capture (MoCap) files at the bvh format

this allows to access to a big database of gesture and scenario. We can also make our own motion capture using an existing system if needed.

The data generated by the simulator are inertial data and distances. The MoCap file allows to compute the position and the orientation of each point of the avatar with a fixed frame rate. From here, it is possible to generate IMU data and distances. IMU are then noised with the desired model that corresponds to the scenario. In theory, distances can be used to generate radio data like RSSI with a proper channel model. This point has not been investigated. In our case we did not include the shadowing effect of the body. But, some papers studied the channel of the body in specific motion condition which results could be integrated as generic estimation values.

6.1.3 Algorithm Study

The study of the algorithms is composed of two steps. Firstly, we determine the best method to address each application case. Secondly, we determine the algorithms used to implement the method.

The common points of the methods are the use of orientation of node and the classification. Algorithms that determine the orientation with inertial measurements are Attitude and Heading Reference System (AHRS). In case of static posture this can be computed by a really simple algorithm but, in case of gesture we adopt an Extended Kalman Filter (EKF). The proposed EKF is of a relatively low complexity since it only uses additions, multiplications, and 6 divisions for 1 iteration. This solution has been selected since it reduces the implementation energy and area cost. The classification method proposed is the combination of a Principal Component Analysis (PCA) and a Nearest Centroid Classifier (NCC). This method is simple to implement and provides good results.

We saw in this chapter that the static posture detection can be done using different combinations of data, not only orientation but also RSSI or raw sensor data. The gesture recognition is possible using orientation of nodes and provides good results, even if the performance depend on the accuracy requirement and so the application. The motion capture is not implemented since it requires the use of beacons or the use of complex algorithms that can not be implemented on the node.

6.1.4 Architecture Implementation

The implementation of the architecture focuses on the EKF. Our objective is to characterize an Application Specific Integrated Circuit (ASIC) implementation of the algorithm. The design flow used includes a High Level Synthesis (HLS) tool. The fixed-point implementation brought to a multi-level study of the error tolerance of the system. The error rate of the gesture recognition is influenced by the accuracy of the orientation estimation of the EKF which depends on the fixed-point format of the implementation. An exploration of the fixed-point parameters is done to find the better accuracy/cost tradeoff. Then we propose an architecture of the EKF composed of several blocks with the HLS tool. Several implementation options are explored like the implementation of the matrix or the blocks hierarchy. Best

solutions are selected to produce the Register Transaction Level (RTL) description of the architecture. These solutions are compiled to gate level with the technological library $65nm$ and $28nm$ to provide an accurate estimation of the EKF implementation. These solutions are characterized by their area and energy consumption. Results show that with $28nm$ technology we can reach the energy consumption of $1\mu J$ per iteration of the EKF. This means that with the use of power gating we can reach the power consumption of $50\mu W$ which is half of the power budget objective.

6.1.5 Top management

In this chapter we study the operating for a whole node. First we describe the global management of the node and how it can be implemented with a minimal overhead using FSM. Then, considering the necessity to save as much energy as possible we define several operating modes. These modes correspond to different performance levels of the node, which are determined by the sampling rate, computing rate and the use or not of the gyroscope.

In a second part, we propose and show the results of a dynamic operating mode. Driven by the motion of the node, the operating mode changes in real-time to be in accordance with computing needs.This is done using accelerometer data, which are filtered to detect motion. When the node is motionless the mode must be at minimal performance to save the maximum of energy. When the node moves the sampling and computing rates increase to provide accurate enough results. The results in this section showed the decrease in accuracy compared to the case that operates at maximum rate and the potential gain in computing activity and gyroscope activity reduction. We saw that at application level the decrease of orientation accuracy has no impact which is encouraging for our application case. The robustness to local orientation errors is a strong point of our method that takes advantage of nodes redundancy by means of a classification method.

6.1.6 Publication

These contributions have been supported by the participation to 2 conferences. One participation to IPSN15 for the paper Radio Signature Based Posture Recognition Using WBSN [61]. Anther participation to BSN15 for the paper Low-Complexity Energy Proportional Posture/Gesture Recognition Based On WBSN [62]. The content of these papers have been used in the Chapter 3. There is a current submission of a paper for the architecture implementation that relies on Chapter 4 results. A submission for a journal is also in writing with results of the dynamic rate adaption of Chapter 5.

6.2 Perspectives

In the simulator, the consideration of the shadowing effect could help to improve modeling the radio data. The information of Line of Sight (LoS) and No Line of Sight (NLoS) between two nodes could be used to, at least, choose the proper channel model to use. With a

Computer Simulation Technology (CST) tool that provides accurate channel modeling we can pre-estimate models that depend on typical scenarios. Integrating these generic scenario models could be a solution to provide a simulation model that is reasonably accurate with a short execution time. Another point to discuss is the lack of standard in the domain of motion capture. The BVH format is mainly used but is not a standard, despite the attempt of [63] to propose a standard format. We can find a lot of motion capture files available on websites but, there is no standard in terms of parameter setting like the avatar biomechanical hierarchy, the frame rate, the scale of data, the naming, etc.... This is the same for posture/gesture recognition or motion capture, there is no database or benchmark to evaluate the performance of methods and algorithms.

The elaboration of the EKF is tricky since it needs to set parameters values with no given methodology. We can assume that some adjustment of the parameters is possible with a thorough study of the filter. We can also consider two configurations of the EKF depending on the use of gyroscope or not. For the classification we can note that the use of PCA and NCC is a tremendously simple solution. This solution is proposed since it is intended to be implemented on the node. But, if we reach one day 1nJ per bit at low rate and including all protocol layers, we can consider to send data to a central node, which is a smartphone.That way we can implement a more complex classification algorithm like a learning machine, which seems a good choice since data of the same class are strongly correlated even between individuals.

If we consider the system as a final product, nodes can be embedded into clothes due to the high integration of an ASIC. This kind of technology is still in development and highly anticipated since it would allow a lot of applications. The issue is that the solution requires that clothes and the electronic system resist to the washing machine.

Now we address the issue of calibration of the system. The system has several calibrations to make in order to be functional. Firstly, sensors must be calibrated which can be done during the production process. Then, the user must initialize the rest posture to match the orientation of nodes with the biomechanical model. Finally, the thresholds must be set up for the activity detection. This can be done by the analysis of a set of postures and gestures that the user has to perform. But, we can also image that these values can be previously estimated for typical application and individuals parameters.

A BAN prototype called Zyggie was developed during the project. This prototype allowed to record real experimental inertial and radio data. The important radio data recorded is the Received Signal Strength Indicator (RSSI). Records have been made for a set of postures with different users. In parallel, another PhD student of this project, Rizwan Masood, proceeded to radio channel simulation using CST tool. Figure 6.1 shows some results of comparison between the experimental and the simulated values. Graphic at the right depict the normalized power value received by the node 4 from other 3 nodes for all experimental record and the simulation. We can clearly see a correlation between these values which again encourage the idea of the existence of a human body channel model. This model should gather channel models for several typical postures which could be tuned with individuals users parameters like size, weight, etc.... This human body channel model would help to design adapted radio antennas and communication protocols that take advantage of the

spatial arrangement of nodes to minimize energy consumption.

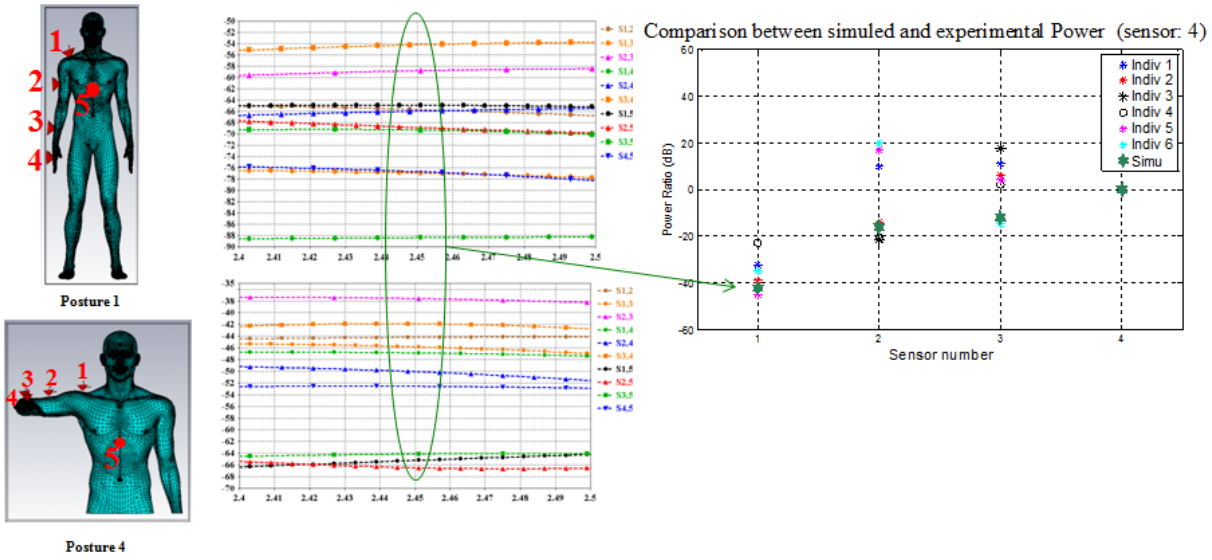


Figure 6.1: Simulation results of radio channel for two postures. Comparison between simulated and experimental Power.

On the other hand, we can consider the reverse situation. Assuming the existence of such a model, can we determine the individual parameters of the user? If enough people use the BoWI system we can consider to gather data. The parameters of all the users like size, weight, age, thresholds, etc... can be used to make a database. This database could be exploited to improve the human body channel model using a larger sample of real data. A the end, during a calibration period, the system can provide raw data to the global model, like postures and corresponding radio channel. Then, a reverse model can guess the individual parameters of the user and eventually confront the data given by the user if any.

Figure 6.2 depicts the overall system we can consider. In a first step, we use simulated results to create a database that contains typical values of individual parameters, postures information and radio channel data, see red arrows. This database is used to generate the global human body channel model. If the user agrees and gives his/her individual parameters the database can be enhanced with real data to improve the global model. Then, for each user, an adapted channel model can be made, see blue arrows. This user model links the postures and the corresponding radio channel. If we know the current posture we can determine the current radio channels, see green arrows. This can be used for fast simulation to create radio data or, this can be used by the system to adapt its communication strategy. But, on the opposite, if we know the current radio channels we can determine the current posture of the user, see purple arrows. This is like a classification by radio measurements. Finally, if the user do not give his/her individual parameters, we can combine the user postures, radio channels and the global model to reversely determine the individual parameters, see yellow arrows. This method allows to provide a user model better than a

generic one in any case.

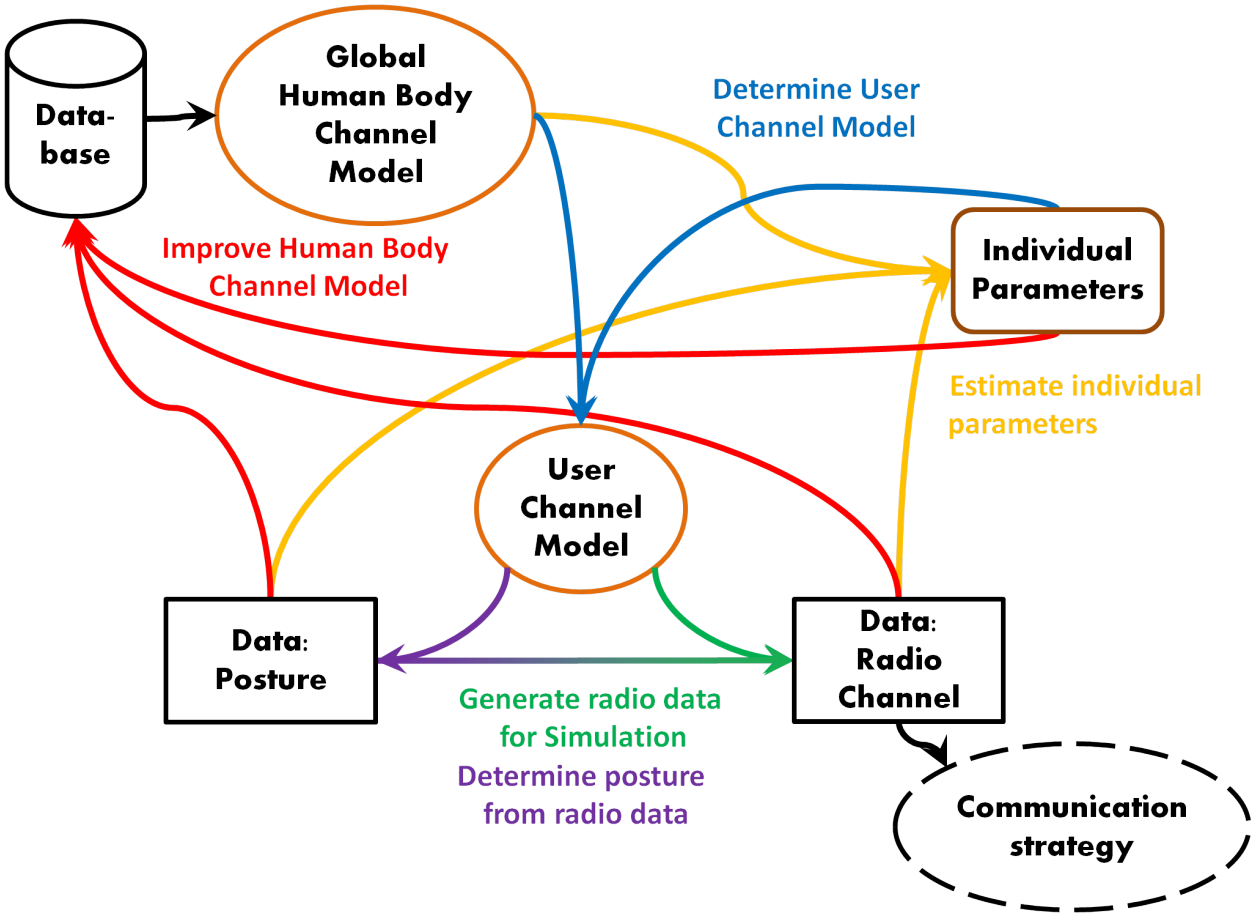


Figure 6.2: Diagram of human body radio channel model and parameters interactions.

In 2017, the Université de Bretagne Sud will use the second version of the BAN prototype Zyggie V2, which includes UWB transceiver for Time Of Arrival (ToA) measurement, to detect and quantify bad postures/gestures. This is encouraged by the new French law that intends to take into account the drudgery at work.

Bibliography

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292 – 2330, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128608001254

[2] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless body area networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1658–1686, Third 2014.

[3] M. Swan, "Emerging patient-driven health care models: An examination of health social networks, consumer personalized medicine and quantified self-tracking," *International Journal of Environmental Research and Public Health*, vol. 6, no. 2, p. 492, 2009. [Online]. Available: http://www.mdpi.com/1660-4601/6/2/492

[4] Y. Hao and R. Foster, "Wireless body sensor networks for health-monitoring applications," *Physiological Measurement*, vol. 29, no. 11, p. R27, 2008. [Online]. Available: http://stacks.iop.org/0967-3334/29/i=11/a=R01

[5] C. M. N. Brigante, N. Abbate, A. Basile, A. C. Faulisi, and S. Sessa, "Towards miniaturization of a mems-based wearable motion capture system," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 8, pp. 3234–3241, Aug 2011.

[6] S. J. Roundy, "Energy scavenging for wireless sensor nodes with a focus on vibration to electricity conversion," Ph.D. dissertation, University of California, Berkeley), 2000.

[7] S. Roundy *et al.*, "Micro-electrostatic vibration-to-electricity converters," ASME International Mechanical Engineering Congress & Exposition, November 2002.

[8] M. Stordeur and I. Stark, "Low power thermoelectric generator-self-sufficient energy supply for micro systems," in *Thermoelectrics, 1997. Proceedings ICT '97. XVI International Conference on*, Aug 1997, pp. 575–577.

[9] T. Starner, "Human-powered wearable computing," *IBM Systems Journal*, vol. 35, no. 3.4, pp. 618–629, 1996.

[10] N. S. Shenck and J. A. Paradiso, "Energy scavenging with shoe-mounted piezoelectrics," *IEEE micro*, vol. 21, no. 3, pp. 30–42, 2001.

[11] A. Mehra, X. Zhang, A. A. Ayon, I. A. Waitz, M. A. Schmidt, and C. M. Spadaccini, "A six-wafer combustion system for a silicon micro gas turbine engine," *Journal of Microelectromechanical Systems*, vol. 9, no. 4, pp. 517–527, Dec 2000.

[12] U. Olgun, C. c. Chen, and J. L. Volakis, "Design of an efficient ambient wifi energy harvesting system," *IET Microwaves, Antennas Propagation*, vol. 6, no. 11, pp. 1200–1206, August 2012.

[13] J.-M. Tarascon and M. Armand, "Issues and challenges facing rechargeable lithium batteries," *Nature*, vol. 414, no. 6861, pp. 359–367, 2001.

[14] M. Swan, "Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0," *Journal of Sensor and Actuator Networks*, vol. 1, no. 3, pp. 217–253, 2012.

[15] "Zyggie platform," http://www.bowi.cominlabs.ueb.eu/fr/zyggie-wbsn-platform.

[16] C. Bachmann *et al.*, "Low-power wireless sensor nodes for ubiquitous long-term biomedical signal monitoring," *Communications Magazine, IEEE*, vol. 50, no. 1, pp. 20–27, January 2012.

[17] A. Siligaris *et al.*, "A 60 GHz UWB impulse radio transmitter with integrated antenna in CMOS 65nm SOI technology," in *11th IEEE Topical Meeting on Silicon Monolithic Integrated Circuits in Rf Systems (SiRF)*, Phoenix, USA, 2011, pp. 153–156, 9799 9799.

[18] B. Marr, B. Degnan, P. Hasler, and D. Anderson, "Scaling energy per operation via an asynchronous pipeline," *Very Large Scale Integration (VLSI) Systems, IEEE Trans. on*, vol. 21, no. 1, pp. 147–151, Jan 2013.

[19] A. D. Young, M. J. Ling, and D. K. Arvind, "Imusim: A simulation environment for inertial sensing algorithm design and evaluation," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, April 2011, pp. 199–210.

[20] P. Asare, R. F. Dickerson, X. Wu, J. Lach, and J. A. Stankovic, "Bodysim: A multi-domain modeling and simulation framework for body sensor networks research and design," in *11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2013, pp. 72:1–72:2.

[21] M. Meredith and S. Maddock, "Motion capture file formats explained," *Department of Computer Science, University of Sheffield*, vol. 211, pp. 241–244, 2001.

[22] "Xsens product," http://www.xsens.com/en/general/mvn.

[23] "Vicon system," http://www.vicon.com.

[24] P. Aggarwal, Z. Syed, X. Niu, and N. El-Sheimy, "A standard testing and calibration procedure for low cost mems inertial sensors and units," *Journal of Navigation*, vol. 61, no. 2, pp. 323–336, 03 2008. [Online]. Available: https://www.cambridge.org/core/article/a-standard-testing-and-calibration-procedure-for-low-cost-mems-inertial-sensors-and-units/E74E0CE76FB3B267F2E5115E4E3603EE

[25] I. Skog and P. Händel, "Calibration of a mems inertial measurement unit," in *XVII IMEKO World Congress*, 2006, pp. 17–22.

[26] K. Benkic, M. Malajner, P. Planinsic, and Z. Cucej, "Using rssi value for distance estimation in wireless sensor networks based on zigbee," in *2008 15th International Conference on Systems, Signals and Image Processing*, June 2008, pp. 303–306.

[27] T. Ayhan, T. Redant, M. Verhelst, and W. Dehaene, "Towards a fast and hardware efficient sub-mm precision ranging system." in *SiPS*. IEEE, 2012, pp. 203–208.

[28] M. Bocquet, C. Loyez, M. Fryziel, and N. Rolland, "Millimeter-wave broadband positioning system for indoor applications," in *Microwave Symposium Digest (MTT), 2012 IEEE MTT-S International*, June 2012, pp. 1–3.

[29] A. Muhammad, "Evaluation of tdoa techniques for position location in cdma systems," Ph.D. dissertation, Faculty of the Virginia Polytechnic Institute and State University, 1997.

[30] "Computer simulation technology tool," https://www.cst.com/products.

[31] R. D'Errico and L. Ouvry, "A statistical model for on-body dynamic channels," *Int. Journal of Wireless Information Networks*, vol. 17, no. 3-4, pp. 92–104, 2010.

[32] M. Mackowiak and L. Correia, "A statistical model for the influence of body dynamics on the gain pattern of wearable antennas in off-body radio channels," *Wireless Personal Communications*, vol. 73, no. 3, pp. 381–399, 2013.

[33] S. S. Haykin *et al.*, *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[34] J. L. Marins, X. Yun, E. R. Bachmann, R. B. McGhee, and M. J. Zyda, "An extended kalman filter for quaternion-based orientation estimation using marg sensors," vol. 4, pp. 2003–2011 vol.4, 2001.

[35] M. Wang, Y. Yang, R. R. Hatch, and Y. Zhang, "Adaptive filter for a miniature mems based attitude and heading reference system," in *Position Location and Navigation Symposium, 2004. PLANS 2004*, April 2004, pp. 193–200.

[36] R. Mahony, T. Hamel, and J. M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, June 2008.

[37] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, June 2011, pp. 1–7.

[38] E. M. Diaz, F. de Ponte Müller, A. R. Jiménez, and F. Zampella, "Evaluation of ahrs algorithms for inertial personal localization in industrial environments," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, March 2015, pp. 3412–3417.

[39] G.-E. D. *et al.*, "A gyro-free quaternion-based attitude determination system suitable for implementation using low cost sensors," in *Position Location and Navigation Symposium, IEEE 2000*, 2000, pp. 185–192.

[40] N. T. Trung, Y. Makihara, H. Nagahara, Y. Mukaigawa, and Y. Yagi, "Inertial-sensor-based walking action recognition using robust step detection and inter-class relationships," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, Nov 2012, pp. 3811–3814.

[41] A. Lin, J. Zhang, K. Lu, and W. Zhang, "An efficient outdoor localization method for smartphones," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, Aug 2014, pp. 1–8.

[42] X. L. Meng, Z. Q. Zhang, S. Y. Sun, J. K. Wu, and W. C. Wong, "Biomechanical model-based displacement estimation in micro-sensor motion capture," *Measurement Science and Technology*, vol. 23, no. 5, p. 055101, 2012. [Online]. Available: http://stacks.iop.org/0957-0233/23/i=5/a=055101

[43] M. A. Brubaker, L. Sigal, and D. J. Fleet, "Estimating contact dynamics," in *2009 IEEE 12th International Conference on Computer Vision*, Sept 2009, pp. 2389–2396.

[44] J. A. Costa, N. Patwari, and A. O. Hero, III, "Distributed weighted-multidimensional scaling for node localization in sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 1, pp. 39–64, Feb. 2006.

[45] T. F. Cox and M. A. Cox, *Multidimensional scaling*. CRC press, 2000.

[46] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

[47] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[48] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, July 2004, pp. 985–990 vol.2.

[49] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[50] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[51] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[52] H. L. Daniel Roetenberg and P. Slycke, "Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors," White paper, April 2013.

[53] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *JOSA A*, vol. 4, no. 4, pp. 629–642, 1987.

[54] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[55] A. J. Hanson, "Visualizing quaternions," in *ACM SIGGRAPH 2005 Courses*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005. [Online]. Available: http://doi.acm.org/10.1145/1198555.1198701

[56] M. D. Ercegovac, L. Imbert, D. W. Matula, J. M. Muller, and G. Wei, "Improving goldschmidt division, square root, and square root reciprocal," *IEEE Transactions on Computers*, vol. 49, no. 7, pp. 759–763, Jul 2000.

[57] W. Li and J. Wang, "Effective adaptive kalman filter for mems-imu/magnetometers integrated attitude and heading reference systems," *Journal of Navigation*, vol. 66, no. 1, pp. 99–113, 007 2012. [Online]. Available: https://www.cambridge.org/core/article/effective-adaptive-kalman-filter-for-mems-imu-magnetometers-integrated-attitude-and-heading-re 01C2D8CA9E64F7ED36DE47B920CBCB80

[58] V. Daniel *et al.*, "Practical motion capture in everyday surroundings," *ACM Trans. on Graphics*, vol. 26, p. 35, 2007.

[59] H. P. Bruckner, C. Spindeldreier, and H. Blume, "Energy-efficient inertial sensor fusion on heterogeneous fpga-fabric/risc system on chip," in *Sensing Technology (ICST), 2013 Seventh International Conference on Sensing Technology*, Dec 2013, pp. 506–511.

[60] "Cacti 5.3." [Online]. Available: http://quid.hpl.hp.com:9081/cacti/

[61] A. Aulery, C. Roland, J.-P. Diguet, Z. Zheng, O. Sentieys, and P. Scalart, "Radio signature based posture recognition using wbsn," in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, ser. IPSN '15. New York, NY, USA: ACM, 2015, pp. 322–323. [Online]. Available: http://doi.acm.org/10.1145/2737095.2737141

[62] A. Aulery, J. P. Diguet, C. Roland, and O. Sentieys, "Low-complexity energy proportional posture/gesture recognition based on wbsn," in *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, June 2015, pp. 1–6.

[63] H.-S. Chung and Y. Lee, "Mcml: motion capture markup language for integration of heterogeneous motion capture data," *Computer Standards & Interfaces*, vol. 26, no. 2, pp. 113–130, 2004.