ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD DE MÁLAGA

PROYECTO FIN DE CARRERA

RECONOCIMIENTO DE LAS ACTIVIDADES RELACIONADAS CON EL MOVIMIENTO HUMANO A TRAVÉS DE SENSORES

RECOGNITION OF HUMAN MOTION RELATED ACTIVITIES FROM SENSORS

INGENIERÍA DE TELECOMUNICACIÓN

MÁLAGA, *2010* *MARÍA JOSEFA VERA NADALES*

Este proyecto fin de carrera se ha desarrollado en las instalaciones en Munich de la Agencia Aeroespacial Alemana (DLR), en el departamento de Comunicación y Navegación, bajo la supervisión de Korbinian Frank y Patrick Robertson.

This master thesis has been carried out in the installations in Munich of the German Aerospace Center (DLR), in the Communication and Navigation department, under the supervision of Korbinian Frank and Patrick Robertson.

Agradecimientos

Quisiera agradecer y dedicar este proyecto, en primer lugar, a mis padres, María José y Fali, y a mi hermano Rafa, que nunca han cesado de demostrarme su amor y su confianza en mí. Su apoyo y comprensión en los malos, muy malos, buenos y muy buenos momentos durante todos estos años ha sido clave en mi vida, y, gracias a ellos, muchos de mis sueños, que nunca imaginé podría llegar a vivir, se han hecho realidad. Nombrar también a mis abuelos, a los que no tengo suficientes palabras de agradecimiento por todo lo que me han apoyado todos estos años. Su amor incondicional ha sido de un valor incalculable para mí.

Y agradecer, en general, a toda mi familia y a mis amigos. Sin duda, lo poco o mucho que tengo, logro y soy se lo debo, en gran medida, a todos ellos. Desde hace años me han brindado su cariño, otorgándome la fuerza y optimismo para afrontar todos los obstáculos que se han presentado en mi vida. También incluyo a Elena y Jesús, mis queridos compañeros de batalla, al igual que a Karan, Antonio, Fabián, Uday y Mihai, por los cuales los buenos recuerdos durante la realización de este proyecto son innumerables.

Este trabajo no habría sido posible sin el convenio establecido entre la Universidad de Málaga y la Agencia Aeroespacial Alemana (DLR). Por ello, agradezco a ambas instituciones y a las personas que han hecho posibles este convenio la oportunidad que me han brindado. En especial, agradecer a mis supervisores, Korbinian Frank y Patrick Robertson, su paciencia, apoyo y confianza en mi capacidad y trabajo, que han sido de un valor inestimable para mí. Agradecer a mi tutor Antonio Díaz Estrella su ayuda y sus ánimos, al igual que todas aquellas personas que participaron y nos ayudaron a obtener los datos necesarios para este proyecto.

Finalmente, no quisiera olvidar a todos los maestros y profesores que despertaron en mí el amor al conocimiento. A ellos les debo la pasión por aprender y el amor por enseñar, que, hasta ahora, han guiado mi vida.

Acknowledgment

I would like to thank and dedicate this project, firstly, to my parents, María José Fali, and to my brother Rafa, who have never ceased to prove their love and confidence in me. Their support and comprehension in bad, very bad, good and very good times over all these years have been a key factor in my life, and, thanks to them, lots of my dreams, that I never imagined I could live, have become real. I would like to thank also to my grandparents, whom I have not enough words of gratitude for all their support these years. Their wholehearted love has been invaluable to me.

And I would like to give my thanks, in general, to all my family and my friends. Doubtlessly, how little or much that I have, achievement and I am I owe, mainly, to all of them. For years, they have given me their love, giving me the strength and optimism to face all obstacles that have arisen in my life. I also include Elena and Jesús, my dear colleagues, as well as Karan, Antonio, Fabián, Uday y Mihai, of which the good memories during the work on the thesis are endless.

This work would not have been possible without the agreement established between the University of Málaga and the German Aerospace Center DLR. Therefore, I am grateful to both institutions and individuals who have made this agreement possible, the chance they have given me. In particular, to thank my supervisors, Korbinian Frank and Patrick Robertson, their patience, support and confidence in my ability and work, that have been invaluable to me. Thanks to my tutor Antonio Díaz Estrella his help and encouragement, like all those who participated and helped us to obtain the data necessary for this project.

Finally, I would not like to forget all the teachers and professors who awakened in me the love for knowledge. To them I owe my passion for learning and my love for teaching, which, until now, have guided my life.

Contents

Acronyms

2D 2 Dimensions

3D 3 Dimensions

ADL Activity of Daily Living

ANN Artificial Neural Network

ARS Activity Recognition System

BF Body Frame

BN Bayesian Network

BPF Band Pass Filter

CM Center of Mass

CPT Conditional Probability Table

DAG Directed Acyclic Graph

DBN Dynamic Bayesian Network

DFT Discrete Fourier Transform

DLR Deutsches Zentrum fűr Luft- und Raumfahrt - German Aerospace Center

FIR Finite Impulse Response

FFT Fast Fourier Transform

FN False Negative

FP False Positive

FSS Feature Subset Selection

GF Global Frame

GPS Global Positioning System

GUI Graphical User Interface

HHMM Hierarchical Hidden Markov Model

HMM Hidden Markov Model

IMU Inertial Measurement Unit

IQR Interquartile Range

JPD Joint Probability Distribution

kNN K-Nearest Neighbor

LDA Linear Discriminative Analysis

LPF Low Pass Filter

MB Markov Blanket

MEMS Micro machined ElectroMechanical Systems

MIT Massachusetts Institute of Technology

PDF Probability Density Function

PDR Pedestrian Dead Reckoning

PPTC Probability Propagation in Trees of Clusters

RFID Radio Frequency Identification

RMS Root Mean Square

RV Random Variable

SF Sensor Frame

SVM Support Vector Machine

TE Test Set

TN True Negative

TP True Positive

TR Training Set

UML Unified Modeling Language

List of Tables

List of Figures

Chapter 1

Introduction

This chapter presents the context and the topic of this master thesis, showing first the main reasons that have led to the development of it. Next, the problem statement that includes the objectives of this Master Thesis and its contributions of this work to the state of the art in the recognition of human motion related activities is explained. Finally, the structure of this report is given.

1.1 Motivation

Telecommunication systems, computers and electronic devices are everywhere. Thanks to the advances in technology, today we have an incalculable number of electronic systems that supply different needs of human beings. This scenario supposes the beginning of a new era in the history of computers, called by Mark Weiser in the early nineties as *Ubiquitous Computing*[1]. First, people shared mainframes. After that, each person has his/her own personal computer. Today, several computers could be integrated and work together to meet the needs of one individual.

These computers will perform much better in supplying the user with the required services, if they are designed to integrate the user's current *context*, i.e. all circumstances relevant for his/her current situation. An example is already patented and develop by Google [1] where mobile phones are aware of the context so they can act in consequence.

[1]See http://www.ubiq.com/hypertext/weiser/UbiHome.html for more information in this topic.

Therefore, *Context-Awareness* is one of the most important elements in Ubiquitous Computing [2]. Context awareness has been defined by Dey in [3]:

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

Concretely, the inference of the current user's activity have became in the last decades in an relevant research area for its importance in the user's context inference. If computers are aware of the person's current activity, a wide variety of new services could be provided:

- Employees of rescue services could provide information of their activity to preserve their safety or to increase the efficiency of the operation. A context aware system could have other industrial applications such as in the office environment, where each employee can provide information so other colleagues could know what he/she is doing. If somebody is in a meeting, the person who requests to talk to him/her does not need to look for the schedule or try to contact him/her. If other employee does not want to be disturbed, the system could inform his/her colleagues to not disturb him/her.

- These systems could be used for assisted living. People who have a grade of dependency or are not able to fend for themselves could provide information related to his/her current activity and relevant events, such as an accident or a fall that could be reported to the emergency services or the person in charge of the patient. Not only elderly people could benefit these systems, also insanes or handicapped persons that require more attention and special cares.

- Today, governments invest millions of euros to preserve the safety of drivers and to decrease the number of traffic accidents. If the car has the ability of reasoning and is aware of the driver's activity and context, the car could inform emergency or police services or even take decisions to preserve the driver's and other users' safety. For example, if the driver is falling asleep, the car could emit an emergency sound or even park in the proximity of the road where it does not disturb others.

- Another concern in today's world is health. New applications such as UbiFit [4] which encourage the user to exercise everyday have been develop for the user's healthcare.
- Apart from these applications, a system awares of current user's activity could be used for entertainment, games and video games, mobile context advertising, etc.

Besides, recent improvements in the performance of small, lightweight and cheap Micro machined ElectroMechanical Systems (MEMS) inertial sensors have made the application of inertial techniques to such problems possible. Consequently, activity recognition with wearable sensors has been an active research topic that dates back to the 1990s [5]. For instance, in 1993, members of the Massachusetts Institute of Technology (MIT) Wearable Computing Project incorporated wearable computing into their everyday lives, describing their experiences with such devices [6].

The proposed Master Thesis outlines a small, ligthweight and wereable system capable of recognizing human motion related activities for user's context inference due to the great possibilities that such system provides to current society.

1.2 Problem Statement

Context information requires the availability of sensors providing the neccessary information. There are however kinds of information, like for instance a person's activity that cannot be measured directly by one sensor. In this cases, such information has to be *inferred* from the combination of information obtained from the available sensors. This field named *sensor fusion* is a very active field of research nowadays due to Ubiquitous computing, the advance in technology and the proved usefulness in the sense of gained information due to integrating several kinds of sensors [4].

Therefore, the design of this activity recognition system consists in the selection and design of three subsystems [4]:

- Sensing module: one or several sensor platforms that could be placed on the wearer or his/her environment in order to collect information of the user and/or its interaction with the environment. Accelerometers are used very often for activity recognition, but also microphones, light sensors, etc.

- Feature processing and selection module: it processes the raw sensor data in order to extract some characteristics or *features* that will help to discriminate activities. The features can be for instance a mean value of an output signal of a sensor or information like the moment of the day (morning, afternoon, evening or night).

- Recognition module: it will use the features to infer the activity an individual is engaged in. Related work on Activity Recognition System (ARS) has implemented this module using different algorithms depending on the feature set and characteristics of the system.

The objectives and contributions of this Master Thesis to the recognition of human motion related activities are pointed out in the following sections.

1.2.1 Objectives

The objectives of this Master Thesis is the design of an activity recognition system that fulfills the following requirements:

- The system should be extensible and has to infer the following activities: *standing, sitting, lying, walking, running, jumping* and *falling*.

- User acceptance will not be reachable if the sensor platform or system is heavy or it does not provide easy mobility. So, it should be not obtrusive and easy to carry. One IMU is used in one location of the body.

- For recording and inference of long term activities, long battery life is required. This constraint implies, in general, to increase the complexity of the system or to reduce the computational burden of the algorithms to get the highest energy efficiency achievable.

- The processing must be done online and in real-time because the system must *immediately* respond to the user's actions. Consequently, low computational burden is needed.

1.2.2 Contributions

This Master Thesis contributes to the research field for the recognition of human motion related activities by the following:

- The sensing module consists of an IMU placed on one location of the body to infer human motion related activities.
- The feature processing uses different window lengths for the discrete-signal processing involved in the feature computation, including a brief physical explanation of the relevant characteristics of the signals obtained through the sensor for every activity in order to achieve a better understanding of why the features could characterize the activities and how robust they are.
- With respect to the recognition module, Bayesian inference for activity recognition has been analyzed. Naïve Bayes and machine-learnt Bayesian networks have been compared for this problem. Besides, an optimal Bayesian state estimation algorithm has been used, Grid-based filter to compare the results of the inference including the dynamic information of human motion related activities into the system. Finally, static and dynamic classifiers have been compared and evaluated.

1.3 Methodology

This Master Thesis has been developed in different phases:

1. Previous study and documentation. This phase consists of the study of the state of the art in activity recognition and the different techniques that can be used for the inference. The status of the state of the art and the contributions to it have been defined in this phase, as well as possible approaches for the activity definition, feature computation and classification. The definition of the activities and the transitions required have been done in this phase.
2. Next, the study of the most adequate body location for the sensor from the point of view of the recognition of human motion related activities has been done. Different body locations have been analyzed and observations about possible applications in activity recognition or location using inertial sensors have been pointed out.
3. In the third phase, sensor data have been recorded and observed for every activity. The relevant and useful outcome information of the sensor for activity recognition have been identified. Hypothesis related

to a physical explanation of the phenomenas that happen during the performance of these activities are given. This physical explanation has been done in order to identify the features that could represent these phenomenas. The aim of this phase is giving to the recognition algorithm all the relevant information about the activities considered through the feature computation in order to characterize every activity, and not only distinguish them. This phase also includes the computation of other features considered in the state of the art, to add these features to the ones identified by the researcher.

4. The following stage consists of the collection of a data set from different subjects considering all the activities and transitions required in the system. This data set is available for the scientific community to encourage research in activity recognition.

5. Once the features are computed, the process of the feature reduction and selection is done. A final set of features has been built in which the most representative features are included.

6. After the process of the identification of the most relevant features, the recognition algorithm is designed taken into account the data obtained in the feature computation as well as the requirement of real-time activity inference. Discrete Bayesian networks have been chosen for the recognition of the user's activity. The feature discretization process is explained.

7. The next phase consists of the development of the Java framework that implements the activity recognition system designed to evaluate real-time inference of user's current activity.

8. Finally, a complete evaluation of the system including the evaluation of real-time inference has been performed. A comparison of Naïve Bayes and unrestricted Bayesian networks as well as static and dynamic exact inference have been provided. Conclusions have been extracted from these results as well as future research topics and outlook.

1.4 Thesis Outline

This Master Thesis contains six more chapters. Chapter 2 gives an overview of the State of the Art in the field of activity recognition. Different approaches and solutions are shown. Appendix A complements this chapter giving detail information about some related work referenced in the bibliography and included in the State of the Art. Chapter 3 explains the background theory required in order to understand and justify the results and the approach designed for activity recognition in this Master Thesis. References that can be consulted for more information are given in every section of this chapter. Chapter 4 describes in detail the approach proposed in this work. This chapter includes from the definition of activities and transitions until the description of the possible recognition algorithms that have been evaluated. Appendices from B to F provide extra information related to the processes described in this chapter. Chapter 5 shows an overview of the program developed in this project as well as its functionality. Details of the program are shown in order to facilitate future work in activity recognition using this Java framework. Appendix G gives information about the format of the data files required for the algorithms of the Java framework. Chapter 6 describes the evaluation process and shows the results achieved for the different recognition algorithms. Different scenerios are defined and tested. Appendices H and I complement this chapter providing details about the results obtained during the process of the evaluation. Finally, Chapter 7 draws conclusions and gives an outlook to possible future work.

The web links provided in this report were all checked and tested on the 2nd of June 2010.

Chapter 2

State of the Art

In this chapter, a review of sensors, activities and recognition algorithms used for activity recognition is provided as well as a description of the aim and general architecture of an Activity Recognition System (ARS). For more details about some of the activity recognition approaches, see Appendix A.

2.1 Activities

Before designing an ARS, it is important to have a clear objective of the final application for what you will use it, as depending on the final application, the sensors are completely different as well as the requirements of the system.

For example, an ARS for house care should be designed for recognizing activities such as *cooking* or *leaving home*. In contrast, an ARS for industrial applications can be designed for activities like *assembling* or *painting car*. In both cases, the sensors to use and the features are different. For house care, if the activity to infer is *cooking*, a temperature sensor over the hotplate is very useful. However, if the activity to infer is *painting car*, a Radio Frequency Identification (RFID) tag on the paintbrush and a RFID reader on the user will give the information whether the user has taken the paintbrush or not [7].

Apart of health care and assisted living, a *context awareness* electronic device could be used for entertainment, games and video games, mobile context advertising, etc. Industries can apply as well this system to inference worker's activities [8].

Designing an ARS for human motion related activities will be useful for

health care encouraging people to do sports or for assisted living as an old person can be monitorized detecting if she has fallen or is going into a vehicle apart of other applications as said in Section 1.1. Research in this area is not recent [9]. Most of the related work tried to infer human motion and posture related activities [10, 5, 11, 12, 13, 14, 15, 16, 17, 18].

There are several kinds of activities that are studied in the literature [5]:

- Human motion activities like *walking, walking upstairs and downstairs, jogging, standing* or *sitting*.
- Activity of Daily Living (ADL): *bathing, dressing, opening a drawer* are examples of this kind of activities. They are activities necessary for people to live independently, primary need activities.
- Instrumental activity of daily living: *using the phone, shopping, food preparation, housekeeping, doing laundry, transportation, taking medications* or *handling finances*. They are more complex than ADL and both terms are used in health care and assisted living.
- Sports activities such as *cycling, rowing, calisthenics* or *martial arts moves*.
- Gestures such as *open door* or *close door* mainly for industrial environments.

Several examples of the activities and sensors used in the related work are shown in Table 2.1.

The nature of the activities that an ARS wants to recognize will decide also the nature of the sensors used: if the activity to infer is related to human motion, the most suitable option is to use sensors related to motion. In contrast, if the activities to infer are related to social interactions as *being in a meeting* or *talking on the phone*, another kind of sensor as a microphone might be required.

2.2 Sensing Module

Multiple kinds of sensors can be used for activity recognition depending on the activities to infer. Reference [5] includes a detailed review of the sensors used until now:

Ref.	Activities	Sensors	Sensor Placement	Data set	Recognition Accuracy
[10]	Human motion	Accelerometer X-Y, GPS	Pocket	10 people (L)	85% - 90%
[11]	Human motion	Accelerometer 3D	Chest & thigh	5 people (S)	89.3%
[14]	Human motion, ADL	Accelerometer 3D, compass, ambient light, force sensor	Right thigh, necklace, right & left wrist	13 people (S)	90.61%
[19]	10 ADL	Video	-	1 person (S)	95%
[20]	20 ADL	Accelerometer X-Y	Left thigh, right ankle,	20 people (S)	84.26%
[21]	Human motion, ADL	Accelerometer	Dominant wrist	7 people (L)	92.86% +/- 5.91%
[22]	ADL	Accelerometer, light sensor, microphone	Chest, wrist & shoes	7 people (L)	$\geq$ 90%
[8]	5 activities to repair a bike	Accelerometer	Torso, left & right sleeve, left upper & lower arm & left hand	3 people (L)	82.7%
[15]	Human motion	Accelerometer	Wrist & thigh	1 person (N)	86% - 93%
[23]	3 Kung Fu movements	Accelerometer	2 at wrist	1 person (L)	96.67%
[16]	Human motion	Accelerometer 2D	Left upper leg	6 people (L)	42% - 96%
[17]	Human motion	Accelerometer	2 at hip	1 person (L)	83% - 90%
[18]	Human motion	Accelerometer	2 at thigh	8 people (L)	92.85% - 95.91%

Table 2.1: Examples of work on activity recognition. The data set column indicates the conditions of the data collected: under laboratory (L), semi-naturalistic (S) or naturalistic conditions (N).

- Ball switches (for details, see [24]).
- RFID tag readers [7] and RFID tags placed on objects.
- Humidity and barometric sensors [5] that will get information about the environment.
- Microphones [5, 25, 22] that record user's voice and ambient sound.
- Video data [19].
- Digital Compass [5] which provides the orientation relative to the magnetic north.
- Light sensors [25, 14, 5, 22] sensitive to the ambient light.
- Fiber optical sensors to measure posture [26].
- Foam pressure sensors to measure respiration rate [27].
- Force sensitive resistors [14, 7].
- Physiological sensors such as oximetry sensors [28], skin conductivity [29] and heart rate sensors [14, 12, 13].
- Stretch sensor, as it was used in [30] in a jacket.
- Manifold air temperature sensors, cabinet door and other sensors useful for ADL at home [31].
- Accelerometers [10, 5, 11, 12, 13, 14, 20, 21, 25, 32, 22, 33, 15, 23, 16, 17, 18].
- Inertial measurement unit (IMU) [8].

Accelerometers are the most common sensor used in the related work. The main reason is that the most popular activities to infer are related to human motion.

The majority of the related work on activity recognition has used accelerometers in multiple locations of the body [20, 11, 12, 14, 25, 32, 22, 8, 15, 17, 18]. However, as explained in Chapter 1, recent work has demonstrated that using multi-modal sensor boards in one single location can offset the

information lost when sensor readings are collected from a single location [4]. If we can really achieve high recognition rates taking measurements from a single location, the system will be less intrusive, more comfortable to wear and the user will not be reluctant to use it. Related work examples of using several sensors in one single location are [7, 33, 13, 5, 10].

Other kind of information as maps, Global Positioning System (GPS) information, bus schedules, people's habits or the use of some objects are studied to increase the results (e.g. [10, 7, 31]).

2.3 Data Set

In the related work about ARS, researchers had to build their own data sets taking the information they required from the sensors and for every activity. The amount of data or the conditions in which they were collected varies from one work to another.

This data set could be collected from the researcher himself [15, 23, 17] or from different people [10, 5, 11, 14, 20, 21, 22, 8, 16, 18] in order to get a more realistic and representative data set.

The conditions under which the data set was collected (see Table 2.1 for some examples) is another factor to take into account:

- Laboratory conditions. In these, the best recognition results will be achieved, but they are less reliable under real conditions.

- Semi-naturalistic. For example, [19] prepared a room as a living room in order to collect the required data. Examples of related work on activity recognition under semi-naturalistic conditions are [11, 14, 20].

- Naturalistic. If the conditions are naturalistic, it means the information from the sensors is collected during a normal day of a person, under real conditions. Very few of the related work collected data under naturalistic conditions [13, 15].

Problems to face when the data set is built is the self-reporting of the activities done. It is not easy to solve it and requires the design of special devices, protocols or human effort to get a reliable data set [34]. Some data sets are currently available, such as the data set used by Bao in [20] and other work by Stephen Intille at MIT. In this work, as human motion

related activities could depend highly on the person, data will be taken from different people under semi-naturalistic conditions.

2.4 Feature Computation

The features computed in the related work depend on the sensor used and its location.

For example, in [5], the features computed from the accelerometers were the mean, variance, energy, spectral entropy, three correlation coefficients between the axis *x,y,z*, the Discrete Fourier Transform (DFT) coefficients in six exponential frequency bands and joint in pairs. In contrast, variance was calculated from the digital compass and visible light sensor. For more details about the features used in the related work, see Appendix A.

In summary, features computed from accelerometers in the related work are among others [5]:

- Mean.
- Variance or standard deviation.
- Peak frequency.
- Amplitude of the peak frequency.
- Slope of the peak frequency.
- Energy of the signal in certain frequency bands.
- Energy of the signal near the peak frequency.
- Maximum value minus minimum element.
- Power of wavelet coefficients.
- Frequency domain entropy or spectral entropy.
- Correlation between the axes.
- Integrated value.
- Root Mean Square (RMS) for each axis.

If the sensor platform is an IMU, these features can be calculated from the pitch, roll or yaw angle as well as from the gyroscope. Also, start and stop angle are taken into account as features in [8]. Other related work that included microphones, used the Cepstral Coefficients for voice characterization [35].

For feature extraction and computation, the signal can be processed by windowing with some overlap. The length of the window in related work is usually fixed, and the length can vary from 1 second [36] up to 6.7 seconds [20]. The overlap of the windows is usually 50% [21, 20], but could also be 70% [25] or even 80% for light sensors [22].

The same set of features and the same window length is usually used for inference of all activities. Only a few works studied the need of having different features and different window lengths to infer different activities [5].

Another requirement of the feature computation (see Chapter 1) is that the computational burden of the system should be controlled in order to get a real-time, accurate and reliable system with long battery life. So, the compromise between real-time processing and long battery life that required low computational burden and accuracy and required feature computation should be studied. Therefore, even if a large set of features is studied in the beginning, it should be reduced in order to fit the requirements of the system. Some techniques used for the feature selection and reduction are:

- Linear Discriminative Analysis (LDA) [21].
- Feature Subset Selection (FSS) [21].
- Forward and backward sequential search algorithms [14].
- Evaluation of information gain from attribute and correlation-based feature subset selection (used in [19]).

Another possibility to improve battery life time of the system is implementing a dynamic sensor selection algorithm as was done in [32].

2.5 Recognition Module

The last phase in an ARS is to finally recognize the activity of a person or group of people. To achieve this, classification techniques can be implemented to determine the current activity through the observation of a set of features.

Techniques based on Bayesian state estimation such as Particle filters [37] or Rao-Blackwellization Particle filters [38] are used. Decision trees [20, 33, 22], K-Nearest Neighbor (kNN) classification [25, 22, 20, 14], Bayesian Network (BN)s [8, 22, 32, 25, 20, 19, 33, 22, 25, 19], Support Vector Machine (SVM)s [19, 33] or neural networks [16, 19, 14, 10] are other examples of classifiers used in related work.

Training of the classifiers used in the literature could be based on *supervised* or *unsupervised* machine learning algorithms.

Supervised learning requires labeled data from each activity called the *training set*. Once the algorithm is trained, it will be able to classify unknown data. The main disadvantage is the effort to label the data. For that reasons, unsupervised approaches are desired. Unsupervised learning implies to construct models from unlabeled data, e.g. by discovering groups of similar examples called *clustering* [5]. Semi-supervised approaches are methods that can be applied when parts of the available data are labeled, while for other parts there exist no labels [5].

The machine learning techniques most commonly used are [5]:

- **Supervised learning** methods:
  1. Decision trees [20, 33, 22].
  2. kNN [25, 22, 20, 14].
  3. Bayesian networks [8, 22, 32, 25, 20, 19, 33, 22, 25, 19].
  4. SVMs [19, 33].
  5. Neural networks [16, 19, 14, 10].
  6. String-matching techniques (for details, see [8]).
- **Unsupervised learning** methods:
  1. Multiple eigenspaces (for further details, see [5]).
  2. Graphical models conbined with Hierarchical Hidden Markov Model (HHMM) [39].
  3. String-matching techniques combined with Hidden Markov Model (HMM) classifiers [40].
- **Semi-Supervised** approaches: semi-supervised approaches for activity recognition are used in [5, 41, 42].

In this work, an explanation of the classifiers that can be used will be given in Section 3.3.

2.6 Test and Evaluation

Results of the classification are commonly given by the classifier accuracy. But e.g. as an unbalanced classifier could provide a good classification accuracy even if one of the activities is not recognized properly, direct 1:1 comparison of results is not possible.

For that reason, precision and recall for every activity are preferred. As an example, [43] presents the classified activities for several classifiers in terms of precision and recall. Some works provide the confusion matrix, so all parameters can be computed from it [20].

Besides, the evaluation results can vary depending on the tests performed. As an example (see Section 2.3), better results are achieved if the classifier is trained and tested with the data from the same person [20], but the aim in general is to achieve an ARS able to get a good performance even if the training and testing is not done individually.

2.7 Conclusions

An overview of the related work aspects on activity recognition has been done. The main conclusions that can be extracted from it is that comparing related work is not as simple as comparing recognition accuracy rates. There is no standard in order to evaluate, test or define an ARS:

- Different activities are considered.
- Different sensors are used depending on the needs.
- Different locations on the body that will give different signals and relevant features.
- Different data sets, usually not very representative and collected from different persons and under different environments, that make very difficult to compare and evaluate the results properly.
- Different parameters in order to evaluate the results: accuracy or precision and recall.

To get a representative data set of the activities that could be standardized or used for the scientific community is not trivial. This is one of the needs of this field now, but the number of activities you can infer, as well as the different kind of sensors and locations on the body or in the environment where you can place them, make it nearly impossible to build a unique and complete data set for activity recognition. So, the lack of reference data and evaluation procedures is one of the main problems in order to evaluate the recognition of very short physical activities. For that reason, some data sets used in previous works are available [20].

Another challenge is the recognition of long-term and high-level activities [5]. Related work have done the recognition of maximum several minutes long activities. Inference of activities that can take hours or days such as *having holidays* or *staying in a conference* have not been done.

A more complete activity recognition state of the art can be found in [5, 44].

Chapter 3

Background Theory

This Chapter explains the theory necessary for this work. Little examples for a better comprehension of the concepts and algorithms are provided.

3.1 Biomechanics

Biomechanics is the field of science that applies mechanics to living organisms. Winter in [45] explains how measurements about the acceleration or movement of the human body can be done by cameras, accelerometers and other methods. Several works about the frequency content of human gait has been done such as [45] and even for different segments of the human body [46].

The following sections shows the most relevant results and considerations concerning human body mechanics that are required for this project. A complete reference in biomechanics is provided by Winter in [47].

3.1.1 Human Body

Figure 3.1 extracted from [47] describes the human body in three planes that intersect in the Center of Mass (CM).

The body Center of Mass (CM) changes dynamically and is not easy to determine. Consequently, the position or trajectory of the body CM is often a parameter of interest when studying posture or movement, specially in balance control [48]. However, physics of human body around the body CM are not required in this project.

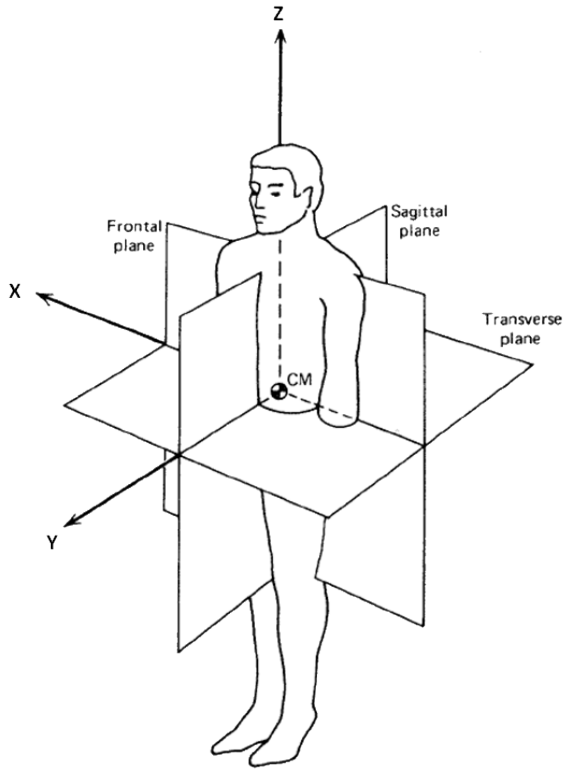


Figure 3.1: Description of Human Body extracted from reference [47]. Three planes are defined from the center of mass of the body. The center of mass of the body changes as soon as the human moves or changes its attitude. Also it depends on human body constitution. Anyway, for *standing* position, the center of mass is located close to the hip.

What is more relevant from the point of view of the acceleration that the body will suffer due to a movement, is how, due to the constitution of the human body, this acceleration propagates through the joints of the body [47]. It is easy to imagine that, as soon as you move your arm considerably, your trunk is affected by the acceleration of your arm. Analogously, the acceleration felt in the foot is transmitted through the leg until the hip.

On the one hand, imagine for a moment an accelerometer located in the hand. In this case, the acceleration that is felt by the foot rarely will be noticed in the hand. Also, the acceleration of the other hand will not be transmitted, almost not significantly, to the hand that wears the accelerometer. On the other hand, if the accelerometer is located near the body CM,

in the hip, the acceleration of the hand will be probably noticed. Maybe not significantly, but because it is transmitted through the arm and shoulders to the trunk - as well as the acceleration of the foot.

3.1.2 Gait in Frequency Domain

One of the most studied movements in biomechanics is human gait [45, 46]. To study human gait without using accelerometers, kinematic data is acquired from position changes of the segments of the body by the double differentiation in order to produce acceleration signals. As any measurements contains noise that should be subtracted or whose effects must be mitigated, the general technique to mitigate its effects is to use a low-pass filter or its equivalent [45].

Angeloni in [46] analyzed the frequency content of camera image plane data for markers on each body segment during gait. This works determines the low-pass filter cutoff frequency that balanced the amount of signal distortion and the amount of random noise passed for every segment. Results conclude that the highest harmonic below the cutoff frequency for the horizontal and vertical camera are below or around 10 Hz and supports the hypothesis that the frequency content of the kinematics of segments decreases caudal to rostral [46].

Antonsson in [45] uses a force platform to measure the acceleration of the foot during heel strike. Results indicate that the frequency band of interest is below 10 Hz containing most of the power of the signal.

Similar results are expected from the use of accelerometers.

3.2 Inertial Navigation

Even if pedestrian navigation is not the objective of this project, as an IMU is used, some concepts related to Inertial Navigation Systems are explained below. A complete reference this section is based on is given by Titterton in [49].

Navigation consists of traveling and finding the way from one place to another and the methods to achieve this are very different.

Navigation following a map is one of the most common and well known methods. In this case, the navigator will determine his or her position by

observation of geographical features defined on the map respecting a grid system or *reference frame*. Another technique for navigation is to take sightings of certain fixed stars to which the navigator can relate his or her position. The fixed stars define a reference that is fixed in space. Such a reference is commonly referred to as an *inertial* reference frame where, given the motion of the Earth and the time of the observation, the navigator is able to use the celestial measurements to define his or her position on the surface of the Earth.

Latitude and longitude are coordinates used for defining the position of the navigator. The problem is that, even if latitude can be computed from the celestial bodies, longitude requires an accurate time reference to be determined.

Inertial navigation consists of using inertial sensors such as gyroscopes and accelerometers, to sense rotational and translational motion relative to an inertial reference frame. So, the position may be calculated from knowledge of initial position and measurements of acceleration and direction.

Inertial navigation is based on the laws of classical mechanics formulated by Sir Isaac Newton. Basically, an object will continue moving uniformly in a straight line unless an external force acts on it disturbing its motion. If an external force is applied to the object, it will produce a proportional acceleration of the body.

Accelerometers are able to measure that acceleration. Performing successive mathematical integrations of the acceleration over time will provide velocity and position of the object. But, as a vectorial magnitude, direction is also required to define the motion of the object. Gyroscopic sensors will measure the rotational motion of the object relative to the inertial reference frame so they can also help to determine the orientation of the accelerometers at all times.

An IMU could consists of three accelerometers, three gyroscopes and three magnetometers placed such a way that they are perpendicular to each other. Setting the sensors perpendicular to each other defines a reference frame that will be called sensor frame (SF). If it is attached to the navigator, acceleration and rotational motion of the object can be measured. However, it is required to know the acceleration and rotational motion of the object relative to the inertial reference frame. In order to obtain them relative to the reference frame, a rotation has to be performed.

There are several methods to represent the attitude of the sensor relative to the reference frame and to perform the rotation [49]

- Direction cosine matrix.

  The direction cosine matrix is a 3×3 matrix whose columns represent unit vectors in body axes projected along the reference axes [49]. The direction cosine matrix is represented as follows:

$$C_s^g = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} \tag{3.1}$$

  where the index g represents the system of reference, the global frame. The index s represents the sensor frame. C_s^g is the direction cosine matrix that rotates from the sensor frame to the global frame and each element c_{ij} in the ith row and the jth column represents the cosine of the angle between the $i - axis$ of the global frame and the $j - axis$ of the sensor frame.

  Given a vector quantity defined in the sensor frame, r^s, it can be expressed in global frame by pre-multiplying the vector by the direction cosine matrix (see reference [49] for more details):

$$r^g = C_s^g r^s \tag{3.2}$$

  An interesting property of the direction cosine matrix is that it is an orthonormal matrix, so the inverse of it is equal to the transpose [50].

- Euler angles.

  Euler angles define the attitude of a body respecting a reference frame based on rotations around different axes. These angles are called yaw (ψ), pitch (θ) and roll (ϕ). Every angle represents a rotation that has to be done over one axis:

  1. rotation of angle ψ around the z-axis of the reference frame.
  2. rotation of angle θ around the new y-axis.
  3. rotation of angle ϕ around the new x-axis.

  Mathematically, every rotation can be expressed as direction cosine matrix as follows [49],

$$rotation\ \psi \text{ around } z-axis,\ C_1 = \begin{pmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.3)$$

$$rotation\ \theta \text{ around } y-axis,\ C_2 = \begin{pmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{pmatrix} \quad (3.4)$$

$$rotation\ \phi \text{ around } x-axis,\ C_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{pmatrix} \quad (3.5)$$

The order in which the rotations are performed can change the final results of the rotated measurements vector [49]. Consequently, to rotate from the reference frame to sensor frame, the whole rotation is expressed as

$$C_g^s = C_1 C_2 C_3 \quad (3.6)$$

This means that to rotate from sensor frame to the reference frame, the inverse operation has to be performed:

$$C_s^g = C_g^{sT} = C_1^T C_2^T C_3^T \quad (3.7)$$

- Quaternions.

  Quaternions are four element vectors that allow to transform from one coordinate to another through a single rotation about a vector $\vec{\mu}$ defined with relative to the reference frame [49]:

$$q = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} cos(\mu/2) \\ (\mu_x/\mu)sin(\mu/2) \\ (\mu_y/\mu)sin(\mu/2) \\ (\mu_z/\mu)sin(\mu/2) \end{pmatrix} \quad (3.8)$$

where μ_x, μ_y, μ_z are the components of the angle vector $\vec{\mu}$ and μ the magnitude of $\vec{\mu}$.

The direction cosine matrix of a quaternion can be obtained as (see [49] for the prove of this expression):

$$C_s^g = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & a^2-b^2+c^2-d^2 & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & a^2-b^2-c^2+d^2 \end{pmatrix} \tag{3.9}$$

Using the equation 3.2, any vector can be rotated from the sensor frame to the global frame.

3.3 Classification Techniques

Artificial intelligence is a scientific discipline that designs and searches for algorithms that allow machines to reason, to give intelligent decisions based on input data.

These algorithms often include a training phase for the reasoning process. They can be classified depending on how the learning or training process is done:

- Supervised learning, where input-output examples are given. It generates a function that maps inputs to desired outputs. Basically, the algorithm tries to model and find the desired function that provides the correct output given the input. Supervised learning is used in classifiers.
- Unsupervised learning that models a set of unlabeled inputs. It performs the clustering of the data.
- Semi-supervised learning that combines both, labeled and unlabeled data to generate an appropriate function or classifier.

The data used for learning and obtaining the classifier is called *training set (TR)* and part of the data set is used for testing and called *test set (TE)*. Details about the evaluation of the classifier are given in Section 3.3.3.

Supervised machine learning, as said before, produces general hypotheses that models the process as a relation which, given the input data, is able to make predictions about future instances. Classifiers use supervised learning in order to find the relation that models the input-output examples provided, but the choice of which specific learning algorithm or classifier should be used is a critical step as the behavior and characteristic of every classifier is different. A review of the most well known classification techniques that can be used including its main features is given in the reference [51] and will be explained here briefly.

3.3.1 Problem of Classification

The classification problem consists of finding a classifier $K : \mathrm{X} -> C$ that maps any instance x_i to its true classification label c_j where C is the set of all the classes c_j, $c_j \in C = c_1, c_2, ..., c_k$ and X is the set of objects or features in the format $\mathrm{x} = (o_1, o_2, ...o_d)$ with their values or attributes A_l, for $1 \leq l \leq d$

The classifier uses data $(\mathrm{x}_1, c_1), ...(\mathrm{x}_n, c_n)$ where x_i are the instances or the vector with the feature values to define the relation $K : \mathrm{X} -> C$.

The data $(\mathrm{x}_1, c_1), ...(\mathrm{x}_n, c_n)$ is used for training the classifier. A well known problem of some classifiers techniques is *overfitting* of data. This problem occurs when the classifier fits the data set, modeling and learning the relation of all pairs (x_m, c_m) including possible *outliers* not significant of the class c_m. Following sections provide a general description of the most used classifiers in brief and their characteristics.

3.3.2 Classifiers

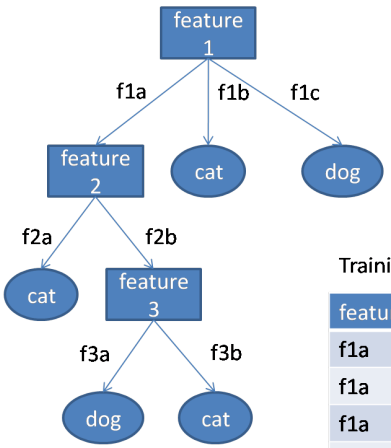
As described in [51], the different kinds of algorithms used for classification are logic based algorithms such as decision trees; neural networks; instance-based like k-Nearest Neighbor (kNN) classification [52]; Support Vector Machines (SVMs) and statistical algorithms such as Bayesian networks (BNs). A reference this section is based on is given by Kotsiantis in [51].

3.3.2.1 Decision Trees

Decision Trees are logic based algorithms that classify instances by sorting them based on feature values [51]. So, each node in a decision tree represents

a feature in an instance to be classified, and each branch represents a value of the node. An example of a decision tree is given in Figure 3.2.

The classification algorithm starts from the root node, following the value of the features or input data. The final decision is made when a leaf is reached.



Training Set:

feature 1	feature 2	feature 3	class
f1a	f2a	f3a	cat
f1a	f2b	f3a	dog
f1a	f2b	f3b	cat
f1b	f2a	f3b	cat
f1b	f2b	f3b	cat
f1c	f2b	f3a	dog
f1c	f2b	f3b	dog

Figure 3.2: An example of a decision tree. Using the value of three features, it is able to classify the class *pet* whose values are *cat* and *dog* based on the input data given by the training set that is shown. The structure of the decision tree is one of the possibles structures learnt by the training set. The Nodes of the tree corresponds to the features. The branches are the value of these features and finally, the leaves of the tree are values of the class to classify.

There are several decision trees possible. The root node should be the feature that best divides the training data and different methods and parameters are defined in order to identify this feature [51]. Overfitting of data by the learning algorithm should be avoided.

Most decision tree algorithms cannot perform well if the instance space is not orthogonal to the axis of one variable and parallel to all other axes conforming hyperrectangles [51].

3.3.2.2 Artificial Neural Networks (ANNs)

Artificial Neural Network (ANN) were inspired by nature. They simulate biological findings relating to the behavior of the brain as a network of units called neurons (see Figure 3.3 for details).

They are based on the following:

1. The information processing is done in lots of simple elements called neurons.
2. The signals are transmitted among neurons through synaptic connections.
3. Each link between two neurons has a weight assigned and called *synaptic weight* that has a multiplier effect over the transmitted signal.
4. Each process unit applies an activation function or *squashing function* to the input signals in order to determine the output signal.

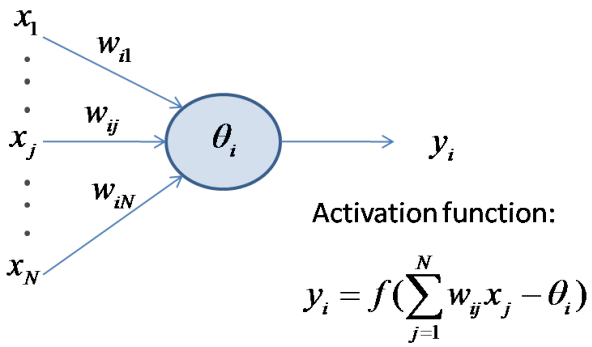


Figure 3.3: Model of a neuron for an Artificial Neural Network. Each neuron is a processor unit that receives the input signals x_i weighted by the synaptic weights w_{ij}. Each neuron applies an activation function to the input signals considering a threshold θ_i. The output signal y_i of the neuron can be the input of other neurons of the network.

Frank Rosenblant developed a simple neuron model called *Perceptron* in 1958. Well-known algorithms are based on the notion of perceptron. The perceptron assigns to each input an output following a learning process of error correction in order to determine the correct synaptic weights. Basically, given the feature values $x_1, x_2, ...x_N$ as input signals and the weights w_{ij} of every link to the perceptron, it computes $\sum x_j w_{ij}$ and output goes through

an adjustable threshold θ_i [51]. If the sum is above threshold, output is 1, else it is -1.

Single layered and multi layered perceptrons neural networks are both used in classification. Radial Basis Function networks are also widely applied in many science and engineering fields (see reference [51] for more details about these networks).

Researchers have compared decision trees and neural networks [51] with the conclusion that neural networks usually perform as well as decision trees, but seldom better. The main disadvantage of ANNs is their lack of ability to reason about their output in a way that can be effectively communicated, so the neural network is a *black box* which results are difficult to interpret [51]. Also, the structure of the network should be defined before the learning process is performed. An exhaustive study about the topology of the network should be done previously [51].

3.3.2.3 K Nearest Neighbors Classification (kNN)

Instance-based learning algorithms require less computation time during the training phase than eager-learning algorithms (such as decision trees) but more computation time during the classification process [51]. One of these algorithms is kNN classification [52].

kNN is based on the principle that the instances within a dataset will generally exist in close proximity to other instances that have similar properties. This algorithm considers that every feature is a dimension of a n-dimensional space where every instance, that corresponds to a set of value of all the features, is a point within that n-dimensional space. kNN considers the relative distance between instances. There are different approaches that defines different distances between instances such as Euclidean, Minkowsky, Manhattan or Chebychev [51]. So, the final decision of the algorithm given an instance is determined by the most frequent class label of the k nearest instances around the input data (see Figure 3.4).

The main disadvantages of these algorithms are related to the storage requirements, their sensitivity to the choice of the definition of the relative distance and the lack of a good method for choosing k [51]

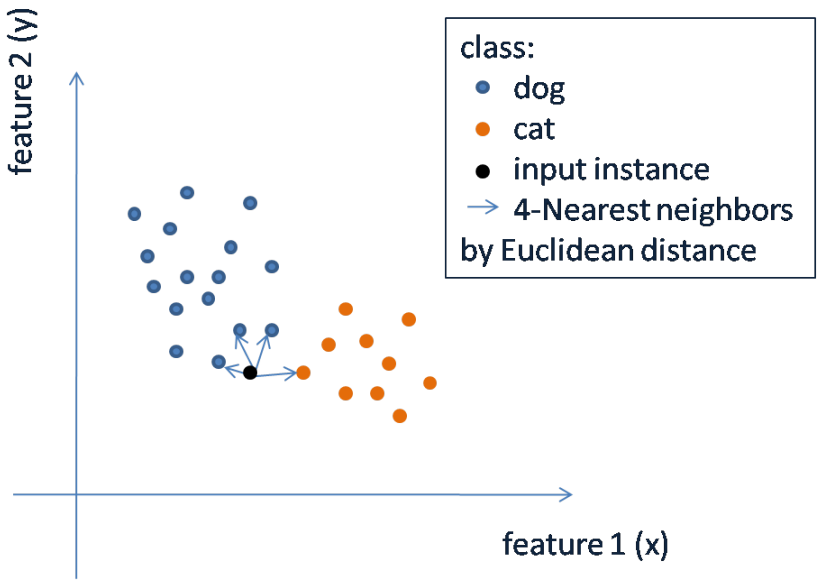


Figure 3.4: An example of application of the k-Nearest Neighbors algorithm. The number of features considered are 2, so the 2-dimensional space is represented. The classifier should determine if the class is *dog* or *cat* given the input data represented as a black point. Using the Euclidean distance and for a value of k of 4, as most of the 5 nearest neighbors using the Euclidean distance definition are from the class *Dog*, the final decision of the classifier given that instance is *Dog* as well.

3.3.2.4 Support Vector Machines (SVMs)

SVMs are a supervised machine classification technique [51]. The condition is that, if the data of different classes are linearly separable, an optimum separating hyperplane can be found. The data contained in that hyperplanes are called support vector points and the solution is represented as a linear combination of only these points [51]. Other data points are ignored, so the complexity of this algorithm depends on the data points found in the margins [51]. A graphical representation of this process is given in Figure 3.5.

As the final solution is a linear combination of the support vector points, there is no need of a large number of training instances and it does not matter if the number of features is large respecting the size of the training set [51].

If the data set contains misclassified instances, a *soft margin* that accepts missclassifications of the training instances can be used (see reference [51] for more details).

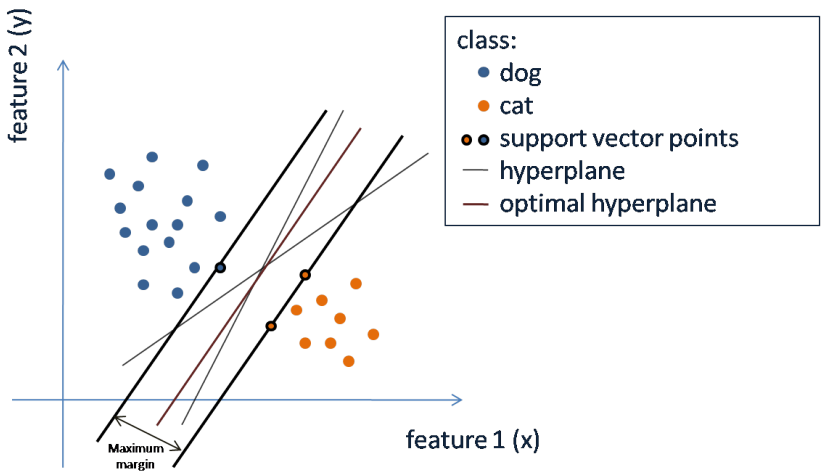


Figure 3.5: Support vector machine algorithm. This algorithm tries to find the optimum hyperplane that separates the different classes to be classified. Support vector points lies on its margin and are shown as well in the figure. The final solution is presented as a linear combination of these points. There are multiple candidate hyperplanes that can be selected. The maximum margin allows the support vector machine algorithm to select one of them.

In case data are not linearly separable, it can be represented on a higher-dimensional space called *transformed feature space* and define a separating hyperplane there. *Kernel functions* are used in order to map new points into the feature space for classification, and the selection of the appropriate kernel function is not trivial [51].

3.3.2.5 Bayesian Networks (BNs)

The classification problem can be considered as an estimation problem, where the goal is to estimate the posterior class probabilities $p(c_j|\mathbf{x})$ (see Section 2.5).

A BN, also called *belief network*, is a probabilistic graphical model that represents the relationships among a set of random variables (RVs) in a directed acyclic graph (DAG) (see Figure 3.6). Every node is a Random Variable (RV) that contains a probability table. In the context of classification, the node *class* is the cause of the observation of a set of feature values, and the network obtained is a causal network.

Every node also contains the likelihood of everyone of its values given its

parents. If the node does not have any parent, these probabilities are the prior probabilities of the states of the RV considered. However, if a RV or node is not conditional independent, the probability table is a conditional probability table (CPT) and contains the probability of a state given its parents. If RVs are not discrete, every node has a conditional probability distribution (CPD). Examples in this section are given for discrete BNs.

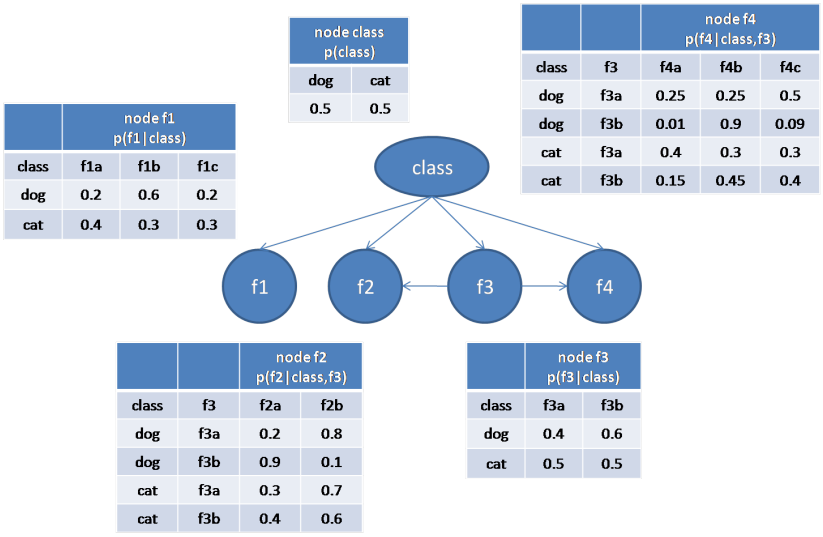


Figure 3.6: An example of a bayesian network used for classification where the dependencies of the features are considered. The node *class* corresponds to the random variable to classify. The other nodes are the features used in order to infer the *class* value. It is a cause network where *class* is the *responsible* of observing the value of the other features. The *class* node does not have any parent and the prior probabilities are given in its probability table. The other nodes have a conditional probability table where the probability of a value of the node given the value of the parents is provided.

Naïve Bayes is a kind of BN where the independences of the features is assumed. It is a strong assumption, but still this approach shows incredible results in classification problems [51] with much lower complexity (see Figure 3.7).

BNs are explained in the next Section due to its importance in this project.

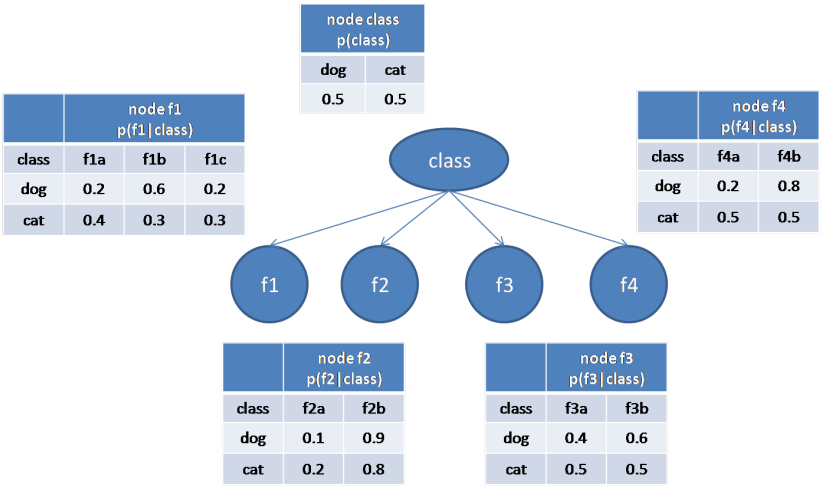


Figure 3.7: Naïve Bayes is a bayesian network where the features are considered to be independent. The algorithm complexity decreases obtaining good results in classification for many applications.

3.3.3 Evaluation of the Classifier

Once a classifier is chosen, the next issue to define is how the evaluation of the classifier will be done. To perform this evaluation, the test set (TE) is presented to the classifier without the ground truth. These instances will produce an output class that should be compared to the true class or ground truth. The result of this process is represented in a confusion matrix with the format shown in Figure 3.8 (a).

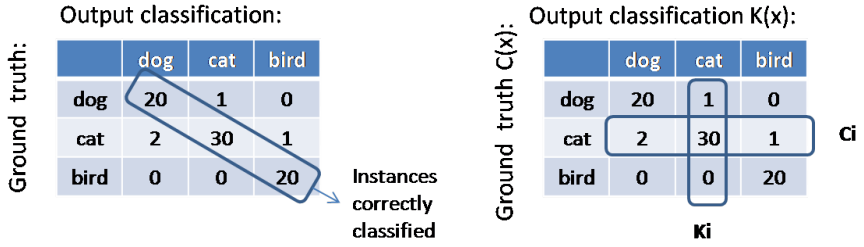
Results can be summarized as shown in Figure 3.9 based on the concepts of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN).

In the following paragraphs, several parameters will be defined and computed from the confusion matrix data.

- Accuracy

  The accuracy of a classifier is a measure of the rate of success of such classifier. It is called a success if an instance of a class is predicted correctly.

$$A_{TE}(K) = \frac{|\{x \in TE | K(x) = C(x)\}|}{|TE|}, \tag{3.10}$$



(a) (b)

Figure 3.8: (a) An example of a confusion matrix where the results of a classifier are shown. The first column contains the ground truth or true classification. The first row indicates the output of the classification. As an example, thirty out of thirty-three instances of the class *Cat* were well classified. Two instances out of thirty-three were misclassified as *Dog* and one as *Bird*. (b) Example with the notation used for computing the parameters that measure the classifier performance. The definition of C_i and K_i is shown graphically for a better comprehension of the notation given above.

where A_{TE} is the accuracy of the classifier given the test set TE, x is the instance given to the classifier, $K(x)$ is the output of the classifier and $C(X)$ is the class label or ground truth of that instance.

- Classification error

  The classification error gives percentage of errors made over the whole set of instances of the test set (TE). An error is produced when an instance of a class is predicted incorrectly.

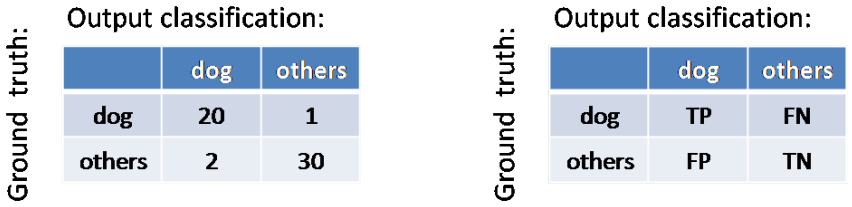
$$E_{TE}(K) = \frac{|\{x \in TE | K(x) \neq C(x)\}|}{|TE|}, \tag{3.11}$$

  where E_{TE} is the classification error of the classifier given the test set TE, x is the instance given to the classifier, $K(x)$ is the output of the classifier and $C(x)$ is the class label or ground truth of that instance.

- Recall

  Recall is a criterion measured for every class label. A high recall implies that the classifier is able to recognize the class when its instance is given as an input.

  Defining $C_i = \{x \in TE | C(x) = i\}$ (see Figure 3.8), recall is computed as



(a) (b)

Figure 3.9: Definition of true positive, true negative, false postivie and false negative for a class based on the results of a process of classification (a) Output of the classifier (b) Definition of true positive, true negative, false positive and false negative

$$Recall_{TE}(K,i) = \frac{|\{x \in C_i | K(x) = C(x)\}|}{|C_i|}, \tag{3.12}$$

where $Recall_{TE}(K,i)$ is the recall of the classifier for the class label i given the test set TE, x is the instance given to the classifier, $K(x)$ is the output of the classifier and $C(x)$ is the class label or ground truth of that instance.

It can be defined as well as:

$$Recall_{TE}(K,i) = \frac{TP_i}{TP_i + FN_i}, \tag{3.13}$$

where TP_i and FN_i are the true positives and false negatives of class i.

- Precision

  A high precision of the classifier for a class i implies that other class $j \neq i$ is not classified as the class i.

  Defining $K_i = \{x \in TE | K(x) = i\}$ (see Figure 3.8), precision is computed as

$$Precision_{TE}(K,i) = \frac{|\{x \in K_i | K(x) = C(x)\}|}{|K_i|}, \tag{3.14}$$

where $Precision_{TE}(K, i)$ is the precision of the classifier for the class label i given the test set TE, x is the instance given to the classifier, $K(x)$ is the output of the classifier and $C(x)$ is the class label or ground truth of that instance.

It is also defined as:

$$Precision_{TE}(K, i) = \frac{TP_i}{TP_i + FP_i}, \tag{3.15}$$

where TP_i and FP_i are the true positives and false positives of class i.

- F-Score

  F-Score or F-measure combines both, precision and recall to get a final score for a class label. This score goes from zero to one and is defined as,

$$F_{\beta,TE}(i) = (1 + \beta^2) \cdot \frac{Precision_{TE}(K, i) \cdot Recall_{TE}(K, i)}{\beta^2 \cdot Precision_{TE}(K, i) + Recall_{TE}(K, i)}, \tag{3.16}$$

  where $Precision_{TE}(K, i)$ is the precision of the classifier for the class label i given the test set TE, $Recall_{TE}(K, i)$ is the recall of the classifier for the class label i given the test set TE and $F_{\beta,TE}(i)$ is F-Score for the value β given a test set TE.

  If $\beta = 1$, $F_1 - Score$ is the harmonic mean of precision and recall. If $\beta = 2$, $F_2 - Score$ weights recall twice as much as precision, and $F_{0.5} - Score$ weights precision twice as much as recall.

To measure these parameters, as said before (in section 3.3.1, the whole data set is divided into a training set for learning and a test set for the evaluation. On the one hand, training set is important as the classifier will be better if the training set is bigger. On the other hand, the evaluation of the classifier will be more accurate as the test set is bigger. Both ideally should be large, but the data set is limited. The dilemma of how to split these data is dealt with differently depending on the technique used.

If N is the number of instances in the data set, the techniques used for the evaluation of the classifier are defined in function of the portion of the data set used for training and for testing:

- Resubstitution $(N; N)$ where the whole data set is used as the training and test set.

  This test will give the apparent classification error of the classifier as the test set is equal to the training set. This classification error is not zero because usually algorithms involve different parameters and measures that lead to uncertainties.

  This error on the training data is not a good indicator of performance on future data, but this error rate indicates only how good or bad are our results on the training data. It evaluates the algorithm, rules and patterns used.

- Holdout $(2N/3; N/3)$, $2N/3$ of the instances of the data set are used for learning and the rest for testing.

  This method uses a certain amount of data for testing and the remainder for training. It could be more reliable by repeating the process with different subsets of data, so in each iteration a certain proportion is randomly selected for training and the rest of data is used for testing.

  The error rates or predictive accuracy on the different iterations are averaged, so an overall error rate and accuracy is achieved.

  Repeated holdout is not still optimum as the test sets overlap.

- X-fold cross-validation $(N - N/x; N/x)$. The data set is split in x parts and one of it is used for testing and the rest for training the classifier.

  This technique avoids the overlap of the test sets. Firstly, it splits the data into x subsets of equal size and after that, each subset is used for testing in turns while the remainder is used for training.

  So the process consists on the following:

  1. Splits the data set into X subsets.
  2. Perform an iterative process where one subset is used for testing and the remainder for training. Repeat the process until all the subsets have been used for testing.
  3. Make the average of the classification error and accuracy.

  The standard cross-validation is ten-fold cross-validation, but in the literature four-fold cross-validation is used as well.

- Leave-one-out $(N-1;1)$, just one instance is out of the data set and used for testing.

  Leave-one-out is a particular cross-validation process where $x = N$ being N the number of instances of the data set.

  In this procedure, all the data set except one record is used for training. That record is used for testing and the result of the evaluation is stored. This process is repeated for all the instances of the data set.

  This technique does not involve random sub-sampling but it is computationally very expensive.

- Leave-one-subject-out is also a technique used in the field of activity recognition in order to evaluate an approach. In this case, the data taken from one subject is not used in the training set. This data will be used to evaluate the classifier as the test set.

3.4 Bayesian Networks

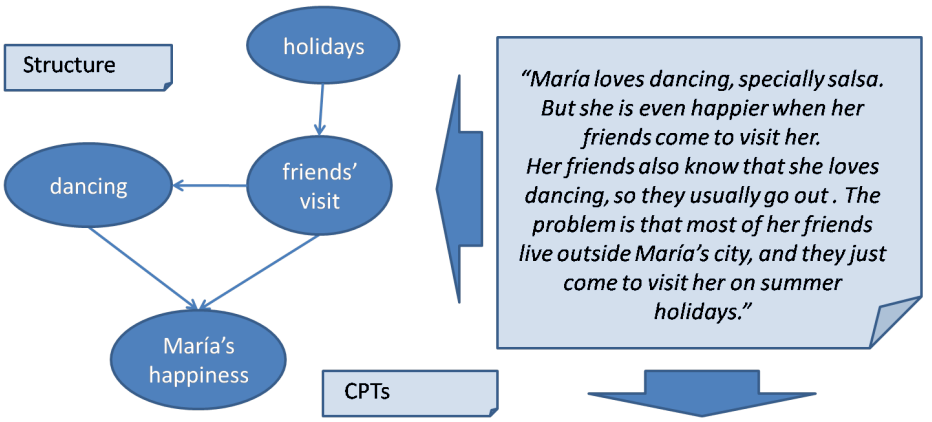
In this Section, BNs are explained as well as learning and inference algorithms that can be used. A complete reference this section is based on is given by Jensen in [53].

3.4.1 Introduction

On the one hand, a causal network is a graph that represents causal relations between events [53]. Figure 3.10 shows an example where the knowledge about a process is represented in a causal network built by human reasoning. The structure of causal networks follows a *directed graph*.

On the other hand, a BN is a probabilistic graphical model that represents the relationships among a set of RVs (nodes) in a Directed Acyclic Graph (DAG) as it cannot contain cycles [53]. In general, the edges between variables do not respond to causality as in a causal network.

However, even if the edges of a BN do not respond to causality, it is assumed that if two events are *strongly* related, even if human reasoning does not see the relation between them, it is possible that this relationship between these two nodes comes from hidden nodes that were not considered in the network, so the nodes are related to compensate the missing information



CPT		María's happiness		
dancing	**friends' visit**	very happy	happy	unhappy
salsa	yes	0.99	0.01	0
salsa	no	0.2	0.6	0.2
others	yes	0.95	0.04	0.01
others	no	0.05	0.7	0.25
none	yes	0.9	0.1	0
none	no	0	0.6	0.4

Figure 3.10: An example of a causal network built by human reasoning. A bayesian network can come from this causal network as soon as these events could be represented as random variables with mutually exclusive states where the events are connected through directed edges following a directed acyclic graph. Each random variable has a conditional probability table where the probability of that variable given evidences in its parents is given according to the process statistics. So, the information we have about María's happiness is represented in the bayesian network as well as the variables or events involved in the process. As an example, the conditional probability table of the random variable *María's happiness* is given where the states of several random variables can be observed.

of the unknown nodes. Besides, concepts applicable to causal networks like *d-separation* can be applied to BN [53].

BN can be discrete or continuous depending on the RVs involved.

A BN represents the Joint Probability Distribution (JPD) in a compact manner. The Joint Probability Distribution (JPD) can be easily computed from a BN by the chain rule [53]:

$$p(x_1, ...x_n) = \prod_{i=1}^{n} p(x_i|Parents(x_i)) \tag{3.17}$$

3.4.1.1 Propagation of Evidences

Causal and BN can be used to follow how a change of certainty in one variable may change the certainty for other variables [53]. Suppose it is known the state of a RV of the network. This knowledge over the state of a RV is called *hard evidence*. If the variable is not instantiated but a certainty of its states is known, it is called *soft evidence*. The evidence given to the network produces an impact on the rest of the RVs depending on the connections between both. The propagation of the evidence depending on the connection between a set of RVs is explained in Figure 3.11 [53].

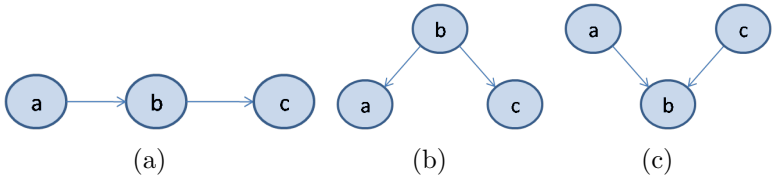


Figure 3.11: Possible connections in a causal network (a) Serial connection of three variables. If b has got hard evidence, a and c are independent. If not, any hard evidence in a or c influences the certainty of c or a respectively through b. So, given hard evidence on b, a and c are *d-separated* (b) Diverging connection. If b has got hard evidence, a and c are independent. If not, any hard evidence in a or c influences the certainty of c or a respectively through b. So, given hard evidence on b, a and c are *d-separated* (c) Converging connection. If b has not got evidence, a and c are independent as the causes of the consequence are independent. But, if b has got evidence, and also it is known evidence in c, the certainty of a decreases. This connection so is open if the children of the causes has got any kind of evidence.

3.4.1.2 *D*-Separation

The definition of d-separation is given in [53]. Two distinct variables a and b are d-separated if, for all paths between a and b, there is an intermediate variable v such that either the connection is serial or diverging and v is instantiated or the connection is converging, and neither v nor any of its descendants have received evidences.

The importance of d-separation resides on the fact that if two variables a and b are d-separated, changes in the certainty of a have no impact on the certainty of b [53].

3.4.1.3 Markov Blanket (MB)

The MB of a node contains the knowledge needed to predict and model the behavior of that node. It is formed by its parents, children and children's parents because they can be used to explain away the node in question [53].

In Figure 3.10, the whole network is part of the MB of the target node *María's happiness* except the node *holidays*. Imagine that evidences over the nodes *dancing* and *friends' visit* are given. If we know that María's friends are visiting her and that they are dancing, no other information is required to know that María is, probably, very happy. In this case, the node *María's happiness* is d-separated of the rest of the network being its MB the minimal set of nodes which d-separates it from the rest of the network.

3.4.2 Learning Algorithms

BNs are characterized by their structure and the parameters or Conditional Probability Table (CPT)s of the network. Example 3.10 shows a BN where the structure was learnt by human reasoning and the parameters estimated by observation of the process. However, knowing the structure of a network given data about a process as well as the parameters involved is not trivial. Different learning algorithms can be used depending on the data set [54]

- Incomplete data set. If some variables in some cases are not observed methods such as Monte-Carlo or the Gaussian approximation can be used.
- Fully observed data set. If data of all the random variables are provided in the data set for all the cases, algorithms such as *Greedy Hill Climber* can be chosen.

A Greedy Hill Climber algorithm starts defining a structure of all the possible structures that can be built with the random variables given in the data set. Inside the space of all the possible BNs, Greedy Hill Climber tries to find the BN that best fits the data.

In order to achieve that, the starting BN is then modified adding, deleting or swapping an arc of the current network. To compare two BNs, the Cooper and Herskovits Log score function is used [54, 55]: it provides a fitness in function of the structure of the network and the data set given to the learning algorithm. This fitness is used by the algorithm to find the best BN over all

the other possibilities. In case that no better BN is found, a random restart occurs, so the algorithm starts again and chooses randomly another BN.

This algorithm does not find the maximum in general, it finds local maxima of the space of all possible BNs.

3.4.3 Inference Algorithms

There are two main types of BN inference tasks [56]: *belief updating* or *probabilistic inference* and *belief revision* or *maximum a posteriori explanation*.

On one hand, *belief updating* calculates the posterior probability $p(x|e)$ where x is a set of query nodes and e is a set of observed values of evidence nodes. On the other hand, *belief revision* finds out the most probable instantiation of some hypothesis variables, given the observed evidence. Even if the inference tasks are different, inference algorithms can be used for both tasks with small modifications. A survey of BN algorithms used for inference is provided in the reference [56].

Inference algorithms for BN can be either algorithms for exact or approximate inference. The categories of these kind of algorithms are shown in Figure 3.12 and extracted from reference [56].

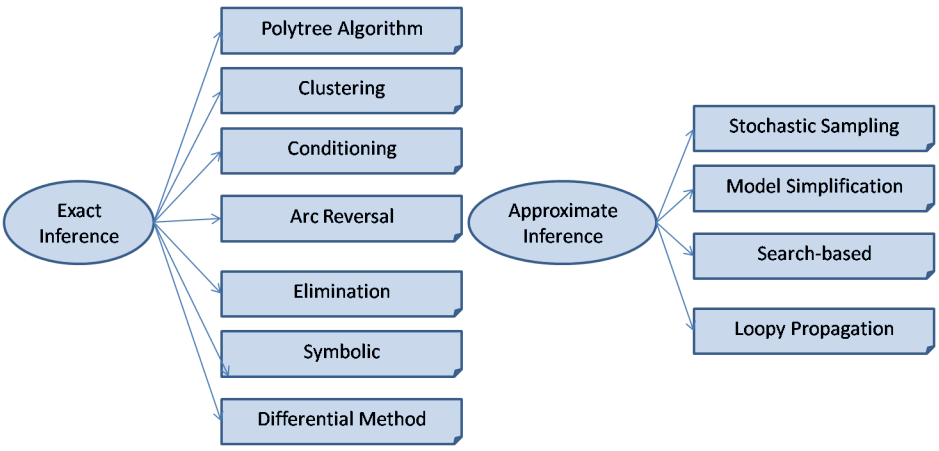


Figure 3.12: Categories of the different inference algorithms for Bayesian networks depending on the kind of inference they perform: exact or approximate. This classification is extracted from reference [56].

One of the most famous algorithms for exact inference that can be used for real-time inference is Lauritzen and Spiegelhalter's *clique-tree propagation*

algorithm of the category of *clustering* algorithms called *Probability Propagation in Trees of Clusters (PPTC)*. It takes advantage of proven fast inference algorithms in tree-like structured BNs [57].

3.4.3.1 Joint Probability

BNs require complex inference algorithms that are not all suitable for real-time inference. However, the problem of exact inference can be simplified hugely if the required probability $p(x|e)$ needs to be computed of a single RV x_{target} whose MB evidence in all its nodes as this variable is d-separated of the rest of the network (see Section 3.4.1.2 for details).

Given evidence in the MB of this RV, the inference problem is now simplified to compute

$$arg_i max[p(x_{target}|e_{MB})] \tag{3.18}$$

where e_{MB} are the evidences of the RVs of the MB.

Using the Bayesian theorem,

$$p(x_{target}|e_{MB}) = \frac{p(x_{target}, e_{MB})}{p(e_{MB})}, \tag{3.19}$$

where $p(e_{MB})$ is constant for all the states of the target node, so the problem is simplified to compute

$$arg_i max[p(x_{target}, e_{MB})] \tag{3.20}$$

Finally, it is required to compute the JPD of the nodes of the MB for all the possible states of the target variable and to find the state that gives the maximum probability. This operation consists of multiplying the correspondent value of the CPTs as explained in Equation 3.17.

This method can be used for real-time applications with the disadvantage that all the nodes of the MB of the target RV should be observed.

3.4.3.2 Probability Propagation in Trees of Clusters

PPTC performs the inference via a secondary structure described in [58]:

- An undirected tree where each node is a *cluster* or *cliques*. Each cluster has a potential associated.

- *Sepsets* that labels given to the edges between the adjacent clusters. The potential of the intersection of the adjacent clusters is assigned to it.

To construct this secondary structure or *join tree*, a sequence of operations should be done to the DAG of the BN [58]. First, the DAG has to be moralized (linking all parent nodes of a node) and triangulated (connecting all nonadjacent nodes in cycles with length greater or equal to four) [57]. Cliques are identified in the triangulated graph and the join tree can be built starting with these cliques as clusters taking care of inserting the appropriate sepsets [58].

The process of inference consists of the propagation of the probabilities through the *message passing* algorithm. Propagation of the probabilities is twofold, it is separated in a *collect-evidence* and a *distribute-evidence* step, each passing and processing once every cluster [57]. Marginalization over a cluster potential that contains the target RV should be performed and normalized by the probability of the evidence to obtain the queried result [57].

This approach works efficiently for sparse networks, but has problems with dense networks, as its complexity is exponential in the size of the largest clique [56].

3.5 Bayesian State Estimation

Two complete references this section is based on are given by Simon in [59] and Arulampalam in [60]. A complete tutorial on HMM is given by Rabiner in [61].

Consider the system described by

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{3.21}$$

where $k \in \mathbb{N}$ is the time index, x_k is the state of the system on time k, v_k is an independent and white process noise associated to the nonlinear system. The function $f_k(\cdot)$ defines the time-varying nonlinear system. The objective of a Bayesian estimator is to recursively track and estimate the state x_k by measurements

$$z_k = h_k(x_k, n_k), \tag{3.22}$$

where z_k is a measurement of this process at time k, and n_k is the measurement noise associated to the measurement process done. The function $h_k(\cdot)$ models the measurement process. The noise sequence n_k is assumed to be independent and white with known Probability Density Function (PDF)s [59, 60].

This process can be modeled as a first-order Hidden Markov Model (HMM) as shown in Figure 3.13.

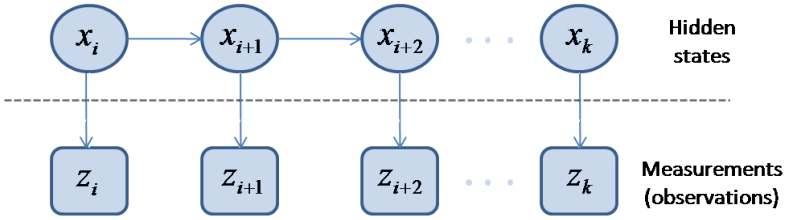


Figure 3.13: First-order Hidden Markov Model. In this process, the states x_k can not be observed but can be estimated through the observations or measurements z_k. A Bayesian estimator is based on Bayes' rule and tries to obtain the conditional probability $p(x_k|z_{1:k})$ to estimate x_k.

A Bayesian estimator approximates the conditional PDF of the state x_k based on the measurements $z_{1:k} = (z_1, z_2, ...z_k)$ obtained up to time k. This conditional probability distribution is denoted as $p(x_k|z_{1:k})$. It is assumed that the initial PDF $p(x_0|z_0) \equiv p(x_0)$ of the state vector, also known as the prior, is available (z_0 being the set of no measurements) [60].

The goal is to find a recursive way to compute the conditional PDF $p(x_k|z_{1:k})$ [59]. Before computing it, the conditional PDF $p(x_k|z_{1:k-1})$ can be obtained using the Chapman-Kolmogorov equation as

$$\begin{aligned} p(x_k|z_{1:k-1}) &= \int p(x_k, x_{k-1}|z_{k-1})dx_{k-1} \\ &= \int p(x_k|x_{k-1}, z_{k-1})p(x_{k-1}|z_{k-1})dx_{k-1} \end{aligned} \tag{3.23}$$

But from Equation 3.21, it is known that x_k is entirely determined by x_{k-1} and v_{k-1}; therefore $p(x_k|x_{k-1}, z_{k-1}) = p(x_k|x_{k-1})$. So

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{k-1})dx_{k-1} \tag{3.24}$$

$p(x_k|x_{k-1})$ is given by the Equation 3.21 and the known statistics of v_{k-1}. The second PDF $p(x_{k-1}|z_{k-1})$ is available at the initial time.

Now, let's try to get the *a posteriori* conditional PDF of x_k. Following Simon in [59], it can be written as:

$$\begin{aligned} p(x_k|z_{1:k}) =& \frac{p(z_{1:k}|x_k)}{p(z_{1:k})}p(x_k) \\ =& \frac{p(z_k, z_{1:k-1}|x_k)}{p(z_k, z_{1:k-1})} \underbrace{\frac{p(x_k|z_{1:k-1})p(z_{1:k-1})}{p(z_{1:k-1}|x_k)}}_{p(x_k)} \\ =& \frac{p(x_k, z_k, z_{1:k-1})}{p(x_k)p(z_k, z_{1:k-1})} \frac{p(x_k, z_{1:k-1})p(z_{1:k-1})}{p(z_{1:k-1})p(z_{1:k-1}|x_k)} \end{aligned} \tag{3.25}$$

This equation can be extended by $p(x_k, z_{1:k})$ [59] to get

$$\begin{aligned} p(x_k|z_{1:k}) &= \frac{p(x_k, z_k, z_{1:k-1})p(x_k, z_{1:k-1})p(z_{1:k-1})}{p(x_k)p(z_k, z_{1:k-1})p(z_{1:k-1})p(z_{1:k-1}|x_k)} \frac{p(x_k, z_{1:k})}{p(x_k, z_{1:k})} \\ =& \frac{p(x_k, z_k, z_{1:k-1})}{p(x_k, z_{1:k})} \frac{p(x_k, z_{1:k-1})}{p(z_{1:k-1})} \frac{p(z_{1:k-1})}{p(z_k, z_{1:k-1})} \frac{p(x_k, z_{1:k})}{p(x_k)} \frac{1}{p(z_{1:k-1}|x_k)} \end{aligned} \tag{3.26}$$

Applying Bayesian theorem,

$$p(x_k|z_{1:k}) = \frac{p(z_{1:k-1}|x_k, z_k)p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})p(z_{1:k-1}|x_k)}; \tag{3.27}$$

z_k is a function of x_k given by the Equation 3.22, so $p(z_{1:k-1}|x_k, z_k) = p(z_{1:k-1}|x_k)$. Substituting and simplifying in Equation 3.27,

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{3.28}$$

All factors are known in Equation 3.28 except $p(z_k|z_{1:k-1})$ that can be obtained with Equation 3.23:

$$\begin{aligned} p(z_k|z_{1:k-1}) &= \int p(z_k, x_k|z_{1:k-1})dx_k \\ &= \int p(z_k|x_k, z_{1:k-1})p(x_k|z_{1:k-1})dx_k \end{aligned} \tag{3.29}$$

But from Equation 3.22, it is known that z_k is entirely determined by x_k and n_k; therefore $p(z_k|x_k, z_{1:k-1}) = p(z_k|x_k)$ and

$$p(z_k|z_{1:k-1}) = \int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k \tag{3.30}$$

Summarizing, a Bayesian estimator recursively estimates $p(x_k|z_{1:k})$ by the following [59]:

1. Assuming that the pdf of the initial state $p(x_0)$ is known, initialize the estimator as follows:

$$p(x_0|z_0) = p(x_0) \tag{3.31}$$

2. For $k = 1, 2, ...$, perform the following:

   (a) The prediction equation gives the *a priori* PDF

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{k-1})dx_{k-1} \tag{3.32}$$

   (b) The update equation gives the *a posteriori* PDF and is obtained with Equations 3.28 and 3.30 by

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{\int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k} \tag{3.33}$$

Equations 3.32 and 3.33 cannot be solved analytically unless some assumptions are made. Kalman filter and Grid-based filters are solutions to these equations for two particular cases. They are optimal algorithms if the assumptions made about the process are given. If not, there are approaches, sub-optimal algorithms such as extended Kalman filters, approximate Grid-based filters and particle filters that can be implemented when the analytic solution is intractable (see reference [60] for details of these algorithms).

3.5.1 Kalman Filter

The Kalman Filter implements a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance when the following assumptions are met.

These assumptions are [60],

- $f_k(x_{k-1}, v_{k-1})$ is known and is a linear function of x_{k-1} and v_{k-1}. So is a linear system where x_k at time k is given by the equation

$$x_k = F_k x_{k-1} + v_{k-1} \tag{3.34}$$

- $h_k(x_k, n_k)$ is a known linear function of x_k and n_k such as

$$z_k = H_k x_k + n_k \tag{3.35}$$

- v_{k-1} and n_k are Gaussian distributions of known parameters so they can be characterized by a mean and covariance as follows

$$\begin{aligned} v_{k-1} &\sim \mathcal{N}(0, Q_{k-1}) \\ n_k &\sim \mathcal{N}(0, R_k) \end{aligned} \tag{3.36}$$

Given these assumptions, if $p(x_{k-1}|z_{1:k-1})$ is Gaussian, then $p(x_k|z_{1:k})$ is also Gaussian. Note that the parameters F_k, H_k, Q_{k-1} and R_k are allowed to be time variant.

Using Equations 3.32 and 3.33 explained above for these assumptions, the PDF involved can be characterized as

$$\begin{aligned} p(x_{1:k-1}|z_{1:k-1}) &\sim \mathcal{N}(x_{k-1}; m_{k-1|k-1}, P_{k-1|k-1}) \\ p(x_k|z_{1:k-1}) &\sim \mathcal{N}(x_k; m_{k|k-1}, P_{k|k-1}) \\ p(x_k|z_{1:k}) &\sim \mathcal{N}(x_k; m_{k|k}, P_{k|k}) \ , \end{aligned} \tag{3.37}$$

where

$$\begin{aligned} m_{k|k-1} &= F_k m_{k-1|k-1} \\ P_{k|k-1} &= Q_{k-1} + F_k P_{k-1|k-1} F_k^T \\ m_{k|k} &= m_{k|k-1} + K_k(z_k - H_k m_{k|k-1}) \\ P_{k|k} &= P_{k|k-1} - K_k H_k P_{k|k-1} \ , \end{aligned} \tag{3.38}$$

being $\mathcal{N}(x; m, P)$ a Gaussian density with argument x, mean m, and covariance P.

K_k is the Kalman filter *gain* or *blending factor* that minimizes the error covariance $P_{k|k}$ and is computed as follows

$$K_k = \frac{P_{k|k-1} H_k^T}{H_k P_{k|k-1} H_k^T + R_k} \tag{3.39}$$

Kalman Filters are used for many applications even if the assumptions do not hold, as good results are usually obtained. This filter can be derived as well from the Least Squares estimation (see reference [59] for the procedure).

3.5.2 Grid-Based Filter

Grid-based methods provides the optimal recursion of the filtered density $p(x_k|z_{1:k})$ if the state space is discrete and consists of a finite number of states. No assumptions about the noise characteristics involved in the process are done. In this case, the hidden Markov Model considered is shown in Figure 3.14 where there are only a finite number of states possible. In general, any state transition from time $k-1$ to time k is possible. These transitions can be represented in a trellis diagram as well.

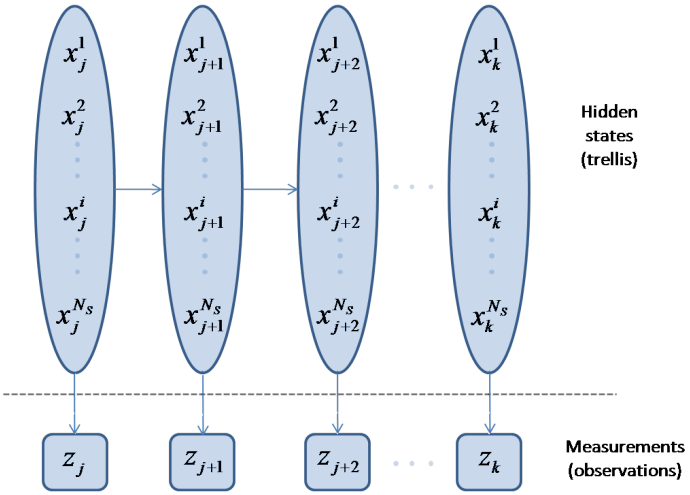


Figure 3.14: Hidden Markov Model of order 1. In this process, a discrete number of N_s states are possible. Grid-based filters can be used to perform the estimation of the current state given the measurements $z_{1:k}$ up to time k.

Being $x_{k-1}^i, i = 1, ..., N_s$ the state space at time $k-1$, so N_s is the number of possible states of the considered process at time $k-1$, the conditional probability of each state x_{k-1}^i given measurements up to time $k-1$ is denoted by

$$w_{k-1|k-1}^i = p(x_{k-1} = x_{k-1}^i | z_{1:k-1}) \tag{3.40}$$

So, the posterior PDF at time $k-1$ can be written as

$$p(x_{k-1}|z_{1:k-1}) = \sum_{i=1}^{N_s} w^i_{k-1|k-1}\delta(x_{k-1} - x^i_{k-1}) \tag{3.41}$$

And substituting in Equations 3.32 and 3.33, we obtain the prediction and update equations for the Grid-based filter

- Prediction:

$$p(x_k|z_{1:k-1}) = \sum_{i=1}^{N_s} w^i_{k|k-1}\delta(x_k - x^i_k), \tag{3.42}$$

- Update:

$$p(x_k|z_{1:k}) = \sum_{i=1}^{N_s} w^i_{k|k}\delta(x_k - x^i_k), \tag{3.43}$$

where

$$\begin{aligned} w^i_{k|k-1} &\triangleq \sum_{j=1}^{N_s} w^j_{k-1|k-1} p(x^i_k|x^j_{k-1}) \\ w^i_{k|k} &\triangleq \frac{w^i_{k|k-1} p(z_k|x^i_k)}{\sum_{j=1}^{N_s} w^j_{k|k-1} p(z_k|x^j_k)} \end{aligned} \tag{3.44}$$

The transition model defines $p(x^i_k|x^i_{k-1})$ and the measurement model give information about $p(z_k|x^i_k)$. They have to be known.

Chapter 4

Design

Making a complete description of every measurement from the inertial measurement unit (IMU) for every activity is difficult as well as useless. If someone is interested in getting a complete and detailed description of the physical phenomenons that happen during these activities, video data are highly recommended. In this work, this is not the priority, so no video data were recorded during the measurements and development of the current data set of activities. However, a general physical explanation of the measurements provided by the sensor is required in order to understand how robust and meaningful the features extracted from the sensor signals will be. For this reason, relevant signals recorded during the performance of every activity considered in this project are shown in Appendix B. This Appendix will help future researchers on activity recognition to understand why, for example, the mean value of the acceleration norm is a good feature when the sensor is located on the belt, but it does not have to be a very good one otherwise.

Consequently, this chapter focuses on the relevant aspects of the approach itself: assumptions, activities, description of the sensor and features used as well as the recognition algorithm designed for real-time inference.

4.1 Assumptions

Two assumptions are made in order to get the required information for activity recognition:

1. The sensor is attached to the person, so the sensor movement corresponds to the person's movement. As soon as the sensor can move freely,

its measurements do not represent the wearer's motion and cannot be considered for wearer's activity recognition.

2. The wearer of the sensor is *standing* at the beginning of the inference so the attitude of the sensor regarding the wearer's body is known.

4.2 Activities

The desired activities are classified in three mutual exclusive categories:

- **Static**:
  - ***Standing***: this does not imply standing straight. It could be in a natural attitude and moving lightly.
  - ***Sitting*** on a chair, on the ground or on any surface.
  - ***Lying***, where the part of the trunk and the hip are lying on the floor, but legs could be up or even the upper part of the trunk could be up right as well.
- **Motion**:
  - Walking upstairs, walking downstairs and walking horizontally are classified as ***walking***.
  - ***Running*** or jogging.
- **Short-term**:
  - ***Falling*** is considered as an involuntary change of body attitude produced by the action of an external force. As the result of *falling*, the subject's body hits the floor, staying in *lying* or *sitting* position.
  - ***Jumping*** includes jumping forward, backward, vertically once or repeatedly.

Besides, the transitions between all these activities are needed for the use of dynamic classifiers. Instead of considering all the possible transitions between all these activities, the transitions can be classified in four main groups:

- ***Accelerating***: occurs from *standing* to *walking* or from *walking* to *running*.
- ***Decelerating***: is the opposite of *accelerating*. In this case, the wearer decreases the rhythm of the motion, so the acceleration of human body during the motion gets lower e.g. from *walking* to *standing*.
- ***Getting up***: any movement that is done in order to *get up*. This transition is done when the wearer is *lying* and gets up or from *sitting* to *standing*. It is a transition between *static* activities.
- ***Getting down***: opposite of *up*. In this case, this transition is done from *sitting* to *lying* (in case the person is on the floor for example or in a sofa) or from *standing* to *sitting*.

Transitions are required to include the dynamic information of human motion, but are not the main objective of this project. On the one hand, *accelerating* and *decelerating* are subtle to detect and not really relevant and they are not considered. On the other hand, *getting up* and *down* are important for the transitions between *static* activities, so they will be taken into account and labeled together.

Finally, once activities and transitions are identified, some considerations about every activity are given in the following paragraphs:

- ***Static*** activities are considered once the sensor wearer is performing such activity. As an example, *lying* happens as soon as the wearer is *lying* on a surface and finishes when he starts to *get up*.
- ***Motion*** activities include *accelerating* and *decelerating* transitions, so any motion activity starts when the wearer is *starting* the motion and finishes as soon as the wearer starts another activity.
- ***Short-term*** activities such as *jumping* and *falling* are defined as *motion* activities: as soon as the wearer is starting to *fall*, the activity is labeled as *falling*.

These considerations are specially relevant for activities such as *jumping* and *falling* that can be divided in different phases. For example, *jumping* could have phases such as *taking impulse*, *flying in the air*, *hitting the floor*. *Falling* has also phases like *starting to fall*, *going down* and *hitting the floor*.

Observing these phases, another approach could consider that an activity is the sequence of gestures or movements and the approach and system model will be different.

4.3 Sensor Module

The following sections explain in detail the sensor used, its characteristics and information provided as well as the location to wear it for activity recognition.

4.3.1 XSens MTx Sensor

The XSens MTx sensor (see Figure 4.1 (a)) is an IMU with integrated 3D magnetometers (3D compass). It has an embedded processor capable of calculating the orientation of the sensor in real time, and returns calibrated 3D linear acceleration, turn rate and magnetic field data [62].

(a) (b)

Figure 4.1: (a) XSens MTx sensor [62]. (b) XSens MTx sensor frame representation [62].

4.3.1.1 XSens MTx Physical Properties

The sensor used, MTx-28A53G25 uses micro machined electromechanical systems (MEMS) solid state technology for accelerometers and gyroscopes. Magnetometers are made of thin film magnetoresistive. Besides, temperature sensors are included to compensate temperature dependencies of the other sensors [62].

Physical properties of MTx-28A53G25 are shown in Table 4.1.

	MTx-28A##G##
Communication Interface	Serial digital (RS-232)
Operating Voltage	4.5-15 V
Power consumption	360 mW
Temperature Operating Range	0°C - 55°C
Dimensions	38 x 53 x 21 mm (W x L x H)
Weight	30 g

Table 4.1: Physical properties of MTx-28A##G## [62].

4.3.1.2 XSens MTx Sensor Fusion

The XSens MTx sensor provides the 3D orientation by the fusion of the sensor output signals using a Kalman filter [8]. The orientation information can be obtained as Euler angles, direction cosine matrix or quaternions [62] (see Section 3.2 for details). Figure 4.2 summarizes the information available from the sensor. For optimal operation it is advised that the MTx remains quasi-static (no motion) for 5 seconds after plugging [63].

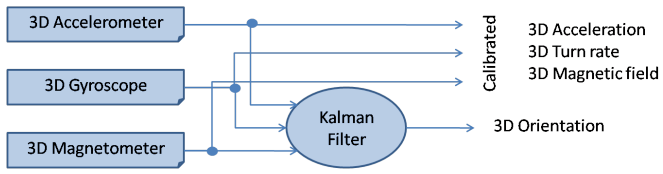


Figure 4.2: The XSens MTx sensor fusion scheme [8]. The fusion of the information of the 3D accelerometers, gyroscopes and magnetometers give the 3D orientation of the sensor regarding a reference frame (see Section 4.3.1.3).

4.3.1.3 XSens MTx Reference Frames

The MTx sensor frame (SF) is shown in Figures 4.1(b) and 4.3. MTx provides the orientation between the Sensor Frame (SF), S and a Earth-fixed reference co-ordinate system, G, that is the global frame (GF). The Earth-fixed reference frame is defined in Cartesian co-ordinate system as [62],

- X is positive when pointing to the local magnetic North.
- Y according to right handed co-ordinates.

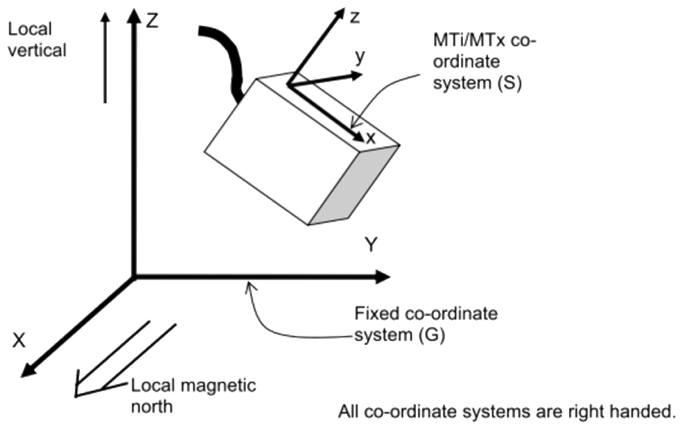


Figure 4.3: The XSens MTx sensor frame, S and the Earth fixed reference frame, G [62]. The direction cosine matrix C_{SG} provided by the sensor is able to rotate any measurement vector from any sensor from S to G.

- Z is positive when pointing upwards. It is the local vertical axis.

Both frames are right handed and are represented in Figure 4.3.

4.3.1.4 XSens MTx Settings

The received data are calibrated by the manufacturer. The calibrated data and orientation performance specification are provided in Tables 4.2 and 4.3 [62].

The orientation output format selected for this project is the direction cosine matrix and the sample frequency is set to 100 Hz.

4.3.2 Location

Different locations for the IMU on the body were studied as the wearer can carry electronic devices on him/her:

- The hand is another possible location where sensors could be contained in a watch.
- The foot or the ankle is the first location that have shown very good results for pedestrian navigation [65] and the sensors could be integrated on the shoe.

		Rate of turn	**Acceleration**	**Magnetic field**	**Temp.**
Unit		$[deg/s]$	$[m/s^2]$	$[mGauss]$	$[^{\circ}C]$
Dimensions		3 axes	3 axes	3 axes	-
Full Scale (FS)	units	$+/-300$	$+/-50$	$+/-750$	$-55...125$
Linearity	% of FS	0.1	0.5	0.2	<1
Bias stability	$units 1\sigma$	5	0.02	0.5	0.5
Scale factor stability	$\%1\sigma$	5	0.02	0.5	0.5
Noise density	$units/\sqrt{Hz}$	0.1	0.002	$0.5(1\sigma)$	-
Alignment error	deg	0.1	0.1	0.1	-
Bandwidth	Hz	40	30	10	-

Table 4.2: Calibrated data performance specification [62]. Bias stability is given temperature compensated and the deviation of 1σ occurs over operating temperature range. Magnetometer noise density can be susceptible to electro-magnetic radiation, so the noise density can be increased several times the typical value. Alignment error is given after compensation of non-orthogonality (after the calibration) [62] (see [64] for details about the origin of these errors).

	XSens MTx
Dynamic Range	all angles in 3D
Angular Resolution	0.05^{o} RMS
Static Accuracy (roll/pitch)	$<0.5^{o}$
Static Accuracy (heading)	$<1^{\circ}$
Dynamic Accuracy	2^{o} RMS
Update Rate	user settable, max 120 Hz

Table 4.3: Orientation performance specification [62]. Static accuracy of heading is given for homogeneous magnetic environments. Dynamic accuracy may depend on type of motion. Angular resolution is the value of the standard deviation of zero-mean angle random walk (see [64] for details about the definition of random walk).

- The sensor placed on the belt or in a pocket is also possible as these places are the locations of mobile telephones and other electronic devices or could be integrated in a belt.

Figures 4.5, 4.4 and 4.6 show the results for different locations of the sensor.

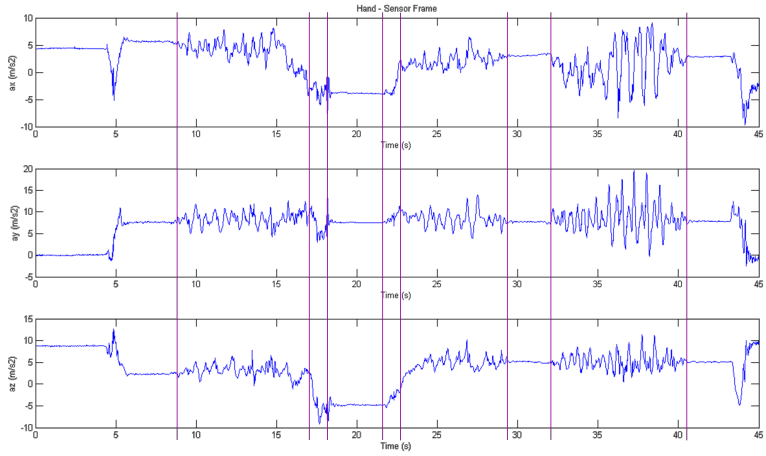


Figure 4.4: Records of the acceleration of every axis of the sensor frame during the performance of *standing - walking - getting down - sitting - getting up - walking - standing - running - standing* for the IMU sensor placed on the hand. During *standing* and *sitting*, movement of the hand depends on the current human motion not related activity. For example, the measured acceleration during the first *standing* is produced for the user's movement to carry the laptop. Hand movement could be also independent of other human motion related activities such as *walking* and *running*.

Foot and ankle are very good for location but not the most desirable options for the recognition of human motion related activities because it has a lack of information of the upper part of the body. If the sensor is in a watch, the movement of the watch around the wrist produces vibrations undesirable for human motion related activities recognition. Also, movement of the hand could be independent of human motion. However, this location is good for the recognition of ADL activities. The belt or the pocket are the best locations for two main reasons:

- The hip is the connection of the upper and lower part of the body. As it was explained in Section 3.1.1, it will sense the acceleration and it will be sensitive to any movement of the trunk and the legs and feet of the human body.
- The second reason is related to the center of mass (CM), and even if it is not compulsory or required, it is a physical fact that could be taken

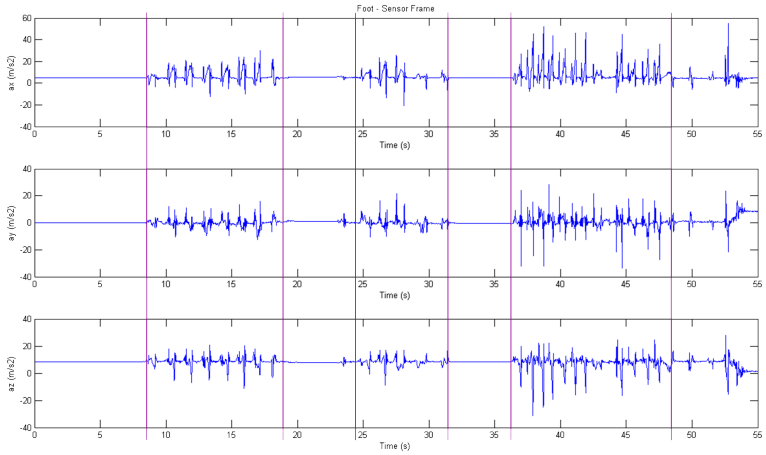


Figure 4.5: Records of the acceleration of every axis of the sensor frame during the performance of *standing - walking - sitting - walking - standing - running - standing* for the IMU sensor placed on the foot. *Standing* and *Sitting* cannot be distinguished from the accelerometer data. Transitions such as *getting up* and *getting down* are not distinguishable as well.

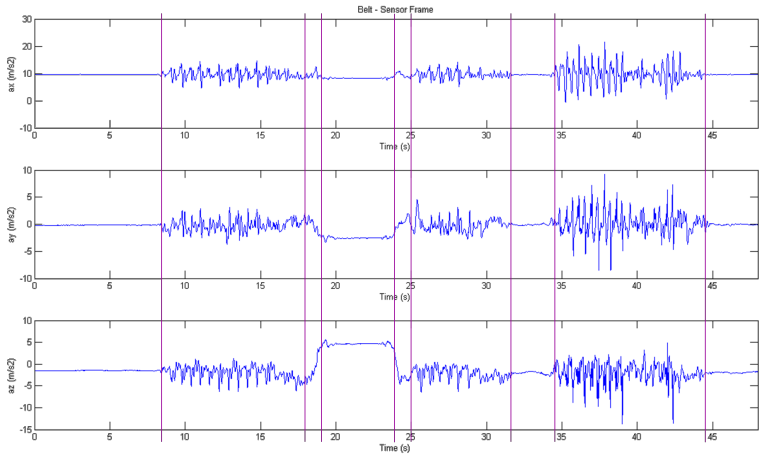


Figure 4.6: Records of the acceleration of every axis of the sensor frame during the performance of *standing - walking - getting down - sitting - getting up - walking - standing - running - standing* for the IMU sensor placed on the belt. Pattern of the signal changes from one activity to another.

into account for future work. Imagine that the person is *standing*. At this position, human body CM is near the hip (see Section 3.1.1). So, any external force strong enough applied to the human body in *standing* position that provokes a rotation of the whole body makes human body rotates over its CM. Gyroscopes should be able to sense this rotation as they are placed near the center of the rotation of the body. An example of this scenery is the activity *falling*. However, human body CM varies during this activity and it cannot be claimed that gyroscopes will sense the angular velocity of the CM during the execution of it, but still outputting gyroscope data could be useful.

4.4 Data set

Sensor output for the different activities and transitions were recorded during the realization of this work.

A total of 16 people, 6 females and 10 males of different ages, body build and constitution participated in the development of the data set. They all had instructions about the activities they had to do but individual freedom was allowed during the performance of every activity. Data were recorded in indoor and outdoor environment under semi-naturalistic conditions. Figure 4.7 shows photos taken during the collection of data. The sensor was placed on the belt either on the right or the left part of the body, at the subject's choice. In order to check orientation performance of the sensor, measurements in different headings were done.

The final data set contains over 4 hours and 30 minutes of activity data. Table 4.4 shows the data per activity. Appendix C provides the data set structure.

4.5 Labeling the Data

Supervised learning requires the labeling of the data that will be used to train the classifier. Figure 4.8 shows the program developed in Java that responsible of labeling the data.

Two people were required for collecting the data (see Figure 4.7. The subject whose data was being collected should perform the activities required. An observer was the responsible for telling the subject what he or she has to

(a) (b)

(c) (d)

Figure 4.7: Photographs taken while recording sensor data for the data set from one of the subjects (a) Walking (b) Running (c) Falling (d) Lying

do and labeling the data (see Figure 4.8).

Once data recording was finished, labeling was checked and modified if necessary in order to obtain reliably labeled data for all activities and transitions considered in this approach.

4.6 Sources of Information

Most related work in biomechanics and recognition of human motion related activities identified the norm of the acceleration as the main source of information. However, IMU sensor can provide acceleration and angular velocity regarding different reference frames as well as angle information between the

Activity	Duration (minutes)
Standing	107
Sitting	55
Lying	25
Walking	70
Running	15
Jumping	7
Falling	2
Transitions	**Duration (minutes)**
Up	3
Down	1

Table 4.4: Constitution of the data set per activities and transitions. Duration is given in minutes.

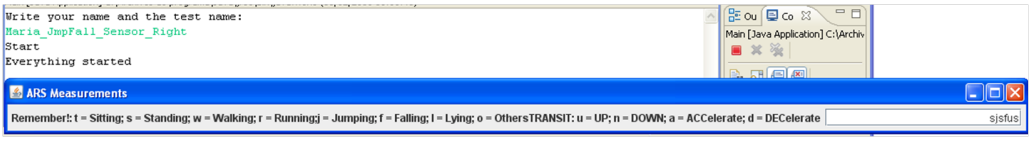


Figure 4.8: Java program used to collect the data set. This program stores the sensor data in a file and a main panel remembers the observer which key it has to be pressed in order to label the data. The labeled activities are also stored in another file. In this example, the sequence *sjsfus* is shown so the subject was *standing, jumping, standing, falling, getting up* and finally, *standing*.

axes of every frame. Identifying which available information is relevant and useful was done after observing every signal extracted from the sensor for every frame and axis. The following sections explain the measurements and sensor information considered.

4.6.1 Approximation to Body Frame

XSens MTx IMU provides the measurements in the sensor frame (SF) and the necessary attitude information in order to rotate them to global frame (GF). However, acceleration and angular velocity measurements relative to the human body seems to be the most relevant information (and not relative to an Earth-fixed reference frame or the sensor itself as the sensor can be placed on the body in any position).

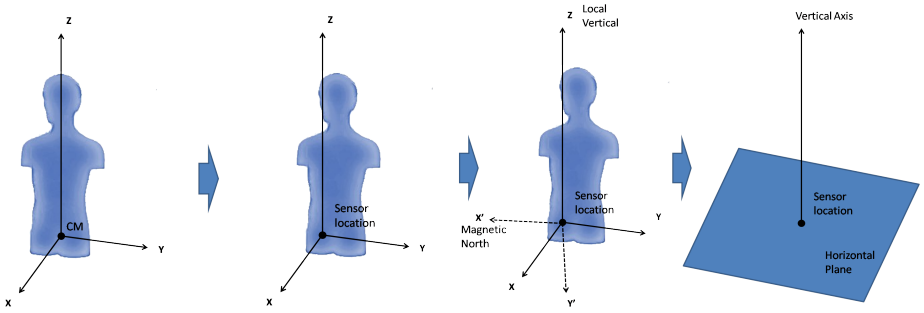
This reference frame called body frame (BF) will be similar to the reference frame shown in Figure 3.1 whose axes will be defined regarding the sensor placement, where, instead of the center of mass (CM) of the body as the origin of the BF, the three axes of the BF will cut at the sensor location.

Obtaining attitude information of the sensor regarding the BF is not simple. First of all, the person should be in a known position, for example, *standing* and not moving. A sensor fusion algorithm should be defined such as the algorithm shown in Figure 4.2 and Section 4.3.1.2. However, the human body heading cannot be estimated unless the human body is directed to Earth magnetic north so magnetometers are used.

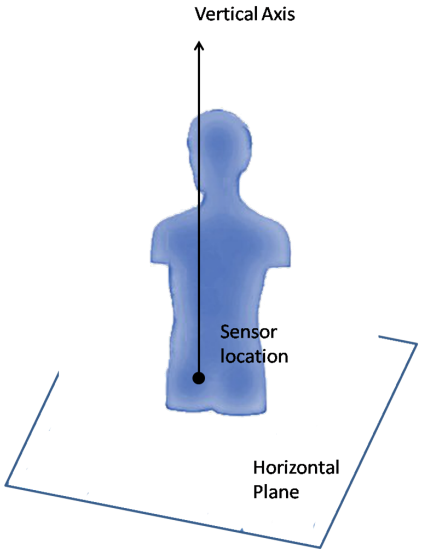
Fortunately, to obtain the sensor measurements regarding all axes of the BF or to define a new sensor fusion algorithm is not required (see Figure 4.9 for a graphical representation of the following explanation). In this work, the horizontal plane and the vertical axis of the BF were identified as the most useful sources of information. Even more, there is no need of designing a new sensor fusion algorithm as the 3D orientation provided by the sensor can be used for obtaining the BF measurements.

3D orientation provided by the sensor is given regarding the GF. If the person is *standing*, the z-axis of the BF or vertical axis in flat terrain can be approximated to the z-axis of the GF. With respect to x and y axes, the heading of the human body is needed. The heading in the GF is estimated using magnetometers and the magnetic north of the Earth, but the heading in the BF does not have any point of reference that could be measured or estimated using a sensor. A solution is that the subject faces the Earth magnetic north every time it runs the sensor so his heading can be computed. However, there is no need of obtaining the acceleration or the angular velocity for every axis of the BF. The recognition of the activities presented in this Master Thesis only require the information contained in the horizontal plane of the body and the vertical axis (see Figure 4.9 (c)).

Consequently, obtaining that information can be done easily from the 3D orientation information of the sensor. MTx sensor should remain quasi-static for 5 seconds after plugging it in order to compute its orientation [63]. During this phase, if the person is *standing*, the output *3D* orientation computed by the sensor corresponds to the attitude between the sensor and a BF of a virtual human facing the magnetic north, if the definition of BF is the one given in Figure 4.9 (a). This ambiguity of the real heading of the wearer of the sensor can be solved joining the information of both axes using the Euclidean norm (see Equation 4.2). Besides, if the sensor is attached to the



(a) (b)



(c)

Figure 4.9: The information required of an approximation to the body frame (a) The ideal body frame will have its origin in the center of mass of the human body in *standing* position or any other location that does not vary over time. However, the sensor location will define the origin of the body frame and location depends on where the user puts the sensor. (b) The heading of the human body cannot be estimated unless magnetometers are used and the wearer of the sensor is oriented facing the magnetic north. Relevant information is included in the norm, the vertical axis and the horizontal plane of the body frame that joins x and y axes measurements using Euclidean norm. (c) Final approximation to the body frame.

body in a way that it cannot move, the orientation between this SF and the BF does not vary and the measurements of the sensor can be rotated

obtaining the information regarding the human body. These are the reasons why the assumptions given in Section 4.1 are done.

Finally, some considerations about acceleration and angular velocity in this new frame are pointed out. Acceleration in the horizontal plane of the body consists of the Euclidean norm of the projection of the acceleration of x and y axes. Acceleration in vertical axis and horizontal plane also contains the information about the body attitude thanks to the gravitational field of the Earth (see Appendix D for details). With respect to the angular velocity, the angular velocity in the vertical axis is easy to interpret as a rotation over this axis. The sign of this rotation depends on the sense of the rotation, being positive if it is clockwise in the direction of the axis of rotation (also known as corkscrew rule or right hand rule) [62]. However, angular velocity in horizontal plane of the BF could not be easily understandable. In this case, the sense of the rotation or if it was done over x or y axis does not matter, as can be seen from the formula of the Euclidean norm of both axes (see Equation 4.2). Any rotation in any of these axes will contribute to the value of the angular velocity in the horizontal plane. This signal will contain information about any rotation done in any direction of the horizontal plane. This information is very useful for activities such as *falling*.

4.6.2 Global Frame

The GF reflects the mechanics of the Earth. The Earth gravity field is measured by the accelerometer and measurements of the acceleration regarding this reference frame could be also important sources of information and are considered in this Master Thesis. Concretely, the relevant information is contained in the vertical axis - that includes the measurement of the acceleration of the local gravity field -.

The angular velocity in GF is not studied as the information of this signal over this frame cannot be interpreted and related to human motion.

4.6.3 Information Signals

Summarizing, the information signals obtained and studied in this Master Thesis are:

- $|a|$, norm of the acceleration and $|w|$, norm of the angular velocity defined as

$$\begin{aligned} |a| &= \sqrt{a_x^2 + a_y^2 + a_z^2} \\ |w| &= \sqrt{w_x^2 + w_y^2 + w_z^2}, \end{aligned} \tag{4.1}$$

where a_i and w_i, $i \in \{\text{x,y,z}\}$ is the acceleration and angular velocity respectively measured in the i-axis in any frame [Unit: m/s^2].

- $|a_{horiz_{BF}}|$, horizontal acceleration in the BF and $|w_{horiz_{BF}}|$, angular velocity in the horizontal plane in the BF,

$$\begin{aligned} |a_{horiz_{BF}}| &= \sqrt{a_{x_{BF}}^2 + a_{y_{BF}}^2} \\ |w_{horiz_{BF}}| &= \sqrt{w_{x_{BF}}^2 + w_{y_{BF}}^2}, \end{aligned} \tag{4.2}$$

  where $a_{i_{BF}}$ and $w_{i_{BF}}$, $i \in \{\text{x,y}\}$ is the measured acceleration and angular velocity respectively in the i-axis of the BF [Unit: m/s^2].

- $a_{vert_{BF}}$, vertical acceleration in the BF and $w_{vert_{BF}}$, angular velocity in the vertical axis in the BF,

$$\begin{aligned} a_{vert_{BF}} &= a_{z_{BF}} \\ w_{vert_{BF}} &= w_{z_{BF}}, \end{aligned} \tag{4.3}$$

  where $a_{z_{BF}}$ and $w_{z_{BF}}$ are the measured acceleration and angular velocity respectively in the z-axis of the BF [Unit: m/s^2].

- $a_{vert_{GF}}$, vertical acceleration in the GF,

$$a_{vert_{GF}} = a_{z_{GF}}, \tag{4.4}$$

  where $a_{z_{GF}}$ is the measured acceleration in the z-axis of the GF [Unit: m/s^2].

- The first derivative of the signals above.

4.7 Feature Extraction

The classification problem described in Section 3.3.1 consists on classifying a target random variable (RV) through the observation of some features.

A feature is a statistical parameter or characteristic of the signal that could be significant for at least one activity or transition.

Most significant sources of information described in Section 4.6.3 for every activity should be observed and studied to recognize and extract meaningful features from them. These signals were observed in time and frequency domain (detailed information related to digital signal processing can be found in [66] and [67]). Figure 4.10 shows an example of the process of observation of the signals for the activity *walking*.

The following sections explain how features are computed and the process of selecting and reducing the number of features.

4.7.1 Feature Computation

The features computed in this work are:

- Temporal domain features:
  - Minimum and maximum and its subtraction.
  - Mean and standard deviation.
  - Median and mean absolute deviation.
  - Interquartile Range (IQR).
  - Root Mean Square (RMS).
  - Integrated value using trapezoidal approximation.
  - Mean crossings.
  - Pearson correlation coefficient between two signals.
- Frequency domain features:
  - Main frequency component.
  - Spectral entropy and relative spectral entropy.
  - Energy of the signal in some frequency bands of interest.

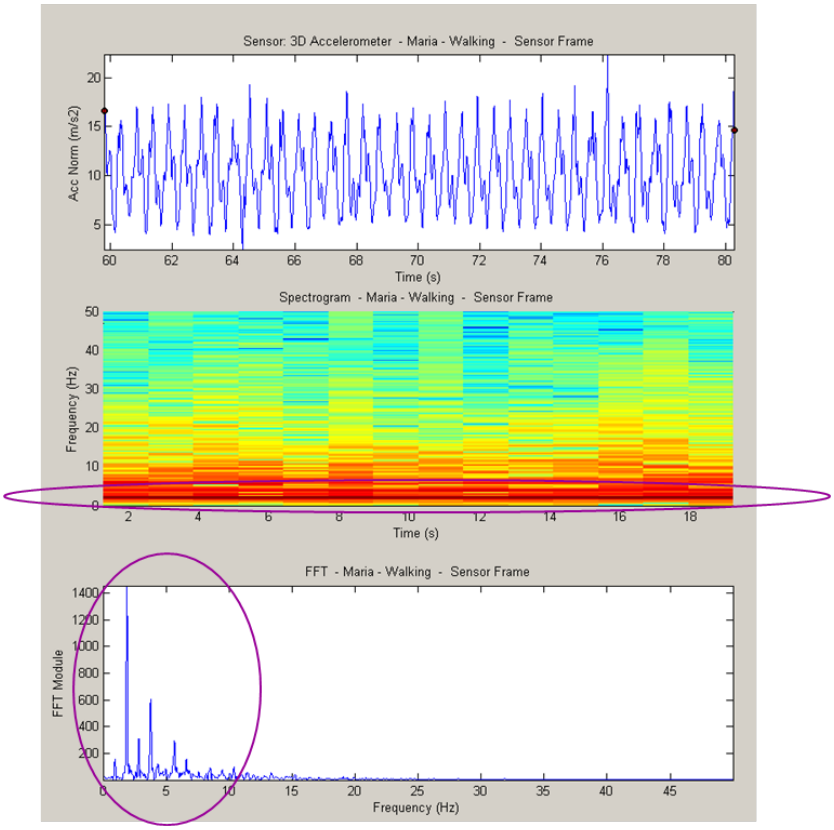


Figure 4.10: Log of 20 seconds of *walking* from one of the subjects. $|a|$ is plotted (above) as well as its spectrogram [66] (middle row) and absolute value of the discrete Fourier transform coefficients [67] (bottom row). The spectrogram shows how the main frequency component of the signal remains almost constant in time. The module of discrete Fourier transform coefficients reflects that the energy of the signal is mostly contained in its first seven harmonics. Besides, *walking* is located in a low frequency band, below 10 Hz.

Not all features are computed for all the information signals described in Section 4.6. Discrete-time signals are acquired from the sensor at a sample frequency of 100 Hz. As a first approach, features are computed every 25 new samples arrived, i.e. at 4 Hz. As activities have different time duration, these features are computed for five window lengths. See Table 4.5 for details.

Samples	Seconds	Overlap
32	0.32	21.87%
64	0.64	60.93%
128	1.28	80.46%
256	2.56	90.23%
512	5.12	95.11%

Table 4.5: Different window lengths used for feature computation. Window lengths are expressed in samples and in seconds for a sample frequency of 100 Hz. Window overlap is also computed for every window length taking into account that features are computed at 4 Hz.

4.7.2 Feature Reduction and Selection

Our objective is to get a final feature set where main physical phenomena and signal information observed during the feature extraction process for every activity are represented. Furthermore, every feature should give new information to the system. The questions to be answered for every feature are:

- **Does it make sense or is its performance in the system due to a limited data set?** Physical explanation of the features and a physical explanation of the signals obtained from the sensors are required. Also, the window length should be chosen carefully: a window length bigger than a couple of seconds will be useful for long-time activities. A short window length of less than one second will be useful as well but for short-time activities. As a compromise, to discriminate between short and long activities, a medium length for the window is usually taken (128 samples). Finally, this analysis will tell if the feature will be good under real world conditions and generalizable for every user of the system.

- **Does it give new information to the system?** Redundant information will increase the computational burden of the system. It is desirable that redundancy is reduced as much as possible.

- **Is it a good discriminator?** It is required to find good discriminators for the activities in order to get good results in the final classification.

- **What are the computational costs?** Real time computation is a requirement of our system, so high computational burden features should be avoided.

The observation in 2D of the result of the computation of the features for every activity has been done to reduce the number of features and help selecting the most relevant ones. Besides, features can be observed in time as well as shown in Figure 4.11 in order to observe their behavior during the performance of one activity.

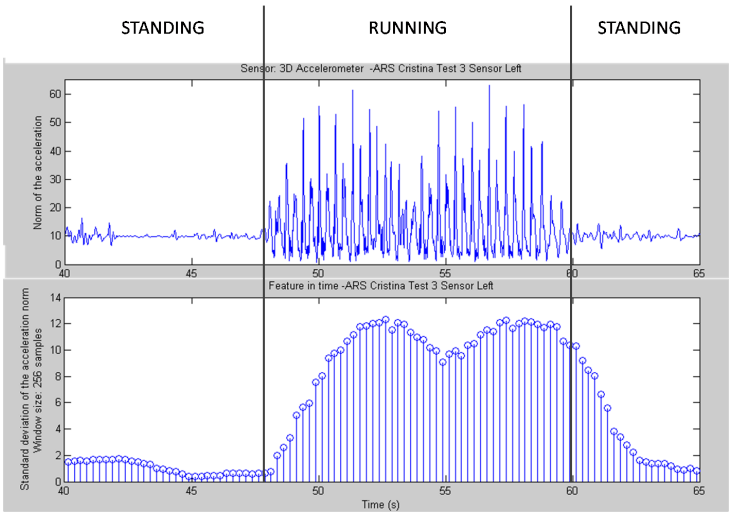


Figure 4.11: Example of the evolution of the features in time for the sequence *standing, running* and *standing*. $|a|$ is plotted for this sequence. Below, standard deviation of $|a|$ in a window of 256 samples, $\sigma_{|a|}$.

Finally, a final set of features need to be selected.

4.8 Final Features

The final feature set was set to the 19 most relevant features. On the one hand, if the objective is just distinguishing activities, the number of features of the final set can be reduced. On the other hand, if a complete characterization of human motion related activities is desired, more features are required in order to represent relevant information of the physics involved on the performance of every activity.

In this section, the final set of features is described and analyzed one by one with several examples related to the behavior of the features for all the activities considered will be shown. See Appendix D for more examples related to the classification of the activities by the following features and see Table 4.6 for a complete reference of the final features set. Appendix D shows the transitions *up* and *down* in contrast to the other activities as well. Some conclusions about the final set of features are given.

Feature No.	Definition	Window size
1	$MAX_{\lvert a_{horiz_{BF}} \rvert}$	128
2	$\overline{\lvert a_{horiz_{BF}} \rvert}$	128
3	$\sigma_{\lvert a_{horiz_{BF}} \rvert}$	128
4	$MAX_{a_{vert_{BF}}}$	128
5	$\overline{a_{vert_{BF}}}$	128
6	$\sigma_{a_{vert_{BF}}}$	128
7	$RMS_{a_{vert_{BF}}}$	128
8	$IQR_{\lvert w_{horiz_{BF}} \rvert}$	128
9	$\overline{a_{vert_{GF}}}$	32
10	$\overline{\lvert a \rvert}$	32
11	$\overline{\lvert a \rvert}$	512
12	$\sigma_{\lvert a \rvert}$	256
13	$IQR_{\lvert a \rvert}$	128
14	$MFC_{\lvert a \rvert}$	128
15	$\hat{E}(\lvert a \rvert_{LPF\ 2.85\ Hz})$	128
16	$\hat{E}(\lvert a \rvert_{BPF\ 1.6-4.5\ Hz})$	64
17	$\hat{E}(\lvert a \rvert_{BPF\ 1.6-4.5\ Hz})$	512
18	$\rho_{a_{vert_{BF}}, \lvert a \rvert}$	128
19	$att_{\lvert a_{horiz_{BF}} \rvert, a_{vert_{BF}}}$	64

Table 4.6: Final set of features. Most of the features are extracted from the body frame and $|a|$ and computed for a window length of 128 samples as it is meaningful for short and long-time activities. A brief description of the features is done in the following sections.

4.8.1 Horizontal Acceleration in the Body Frame

The horizontal acceleration in the body frame, $|a_{horiz_{BF}}|$, is another interesting source of information. The main reason is that body attitude information is contained in it.

4.8.1.1 Maximum Value

The next feature, Feature 1, is the maximum value of $|a_{horiz_{BF}}|$ in a window length of 128 samples:

$$MAX_{|a_{horiz_{BF}}|} = max(|a_{horiz_{BF}}|) \ , \tag{4.5}$$

where $|a_{horiz_{BF}}|$ is the norm of the horizontal acceleration described in Section 4.6.

As can be observed from Figure 4.12, *falling* implies a high value of $|a_{horiz_{BF}}|$ once the body hits the floor for the reaction force that the ground produces in the body. E.g. *running* and *jumping* have higher values than *walking*. For *static* activities, while you are *standing*, this feature is around zero. As soon as the attitude of your body changes to *sitting*, depending on the inclination of your body, this value will be higher being maximum for *lying*, as gravity value is felt in this plane for this activity.

4.8.1.2 Mean Value

Feature 2, $\overline{|a_{horiz_{BF}}|}$, is the mean value of $|a_{horiz_{BF}}|$ in a window of 128 samples, defined as:

$$\overline{|a_{horiz_{BF}}|} = \frac{1}{N}\sum_{k=1}^{N} |a_{horiz_{BF}}|\,[k] \text{ with } N = 32,\, 512, \tag{4.6}$$

where N is the number of samples, $|a_{horiz_{BF}}|$ is the horizontal acceleration of the body frame (BF) defined above and $|a_{horiz_{BF}}|\,[k]$ is $|a_{horiz_{BF}}|$ for the $k-th$ sample.

Window length of 128 samples provides information for all the activities, short-time or long-time activities.

This feature is more reliable than $MAX_{|a_{horiz_{BF}}|}$, but specially meaningful for *static* activities. They can be distinguished for the same reasons explained for $MAX_{|a_{horiz_{BF}}|}$. Figure 4.13 shows this feature in combination

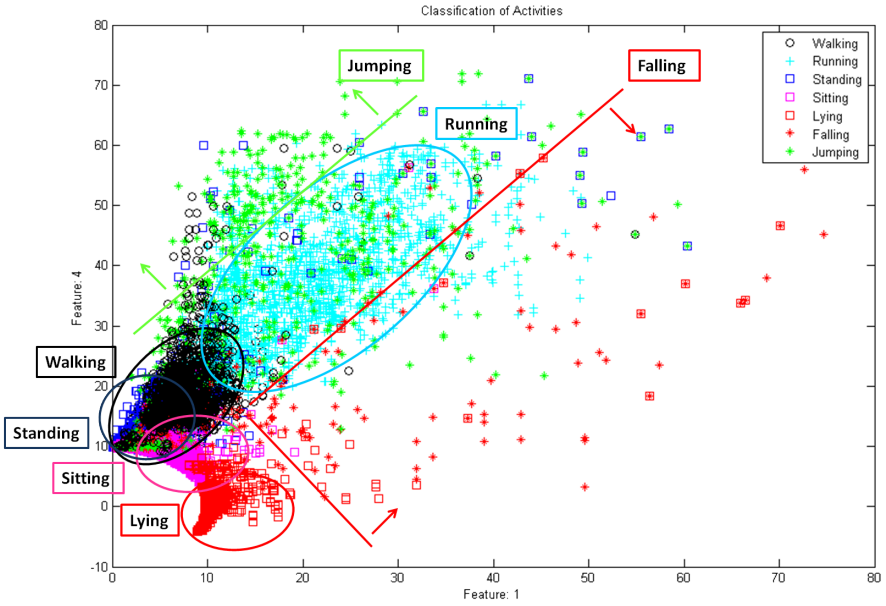


Figure 4.12: $MAX_{|a_{horiz_{BF}}|}$, Feature 1, and $MAX_{a_{vert_{BF}}}$, Feature 4, in a 128 samples window. The combination of both allows to distinguish *standing*, *sitting* and *lying* as body attitude information is contained in the body frame. The distinction of *falling* and *jumping* is also possible as both presents their maximum value in different signals once the body has hit the floor. Respect to *walking*, *running* and *jumping* are distinguished from it as well.

with $\overline{|a_{vert_{BF}}|}$. The combination of both are a good discriminator for *static* activities and for *falling* with respect to non-static activities as attitude information of the body is reflected on the acceleration measured.

4.8.1.3 Standard Deviation

Finally, Feature 3 is the standard deviation of $|a_{horiz_{BF}}|$ represented by $\sigma_{|a_{horiz_{BF}}|}$ and computed over a window of 128 samples (see Figure 4.14):

$$\sigma_{|a_{horiz_{BF}}|} = \sqrt{E[(X - \mu_{|a_{horiz_{BF}}|})^2]} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(|a_{horiz_{BF}}|\,[k] - \overline{|a_{horiz_{BF}}|})^2} \tag{4.7}$$

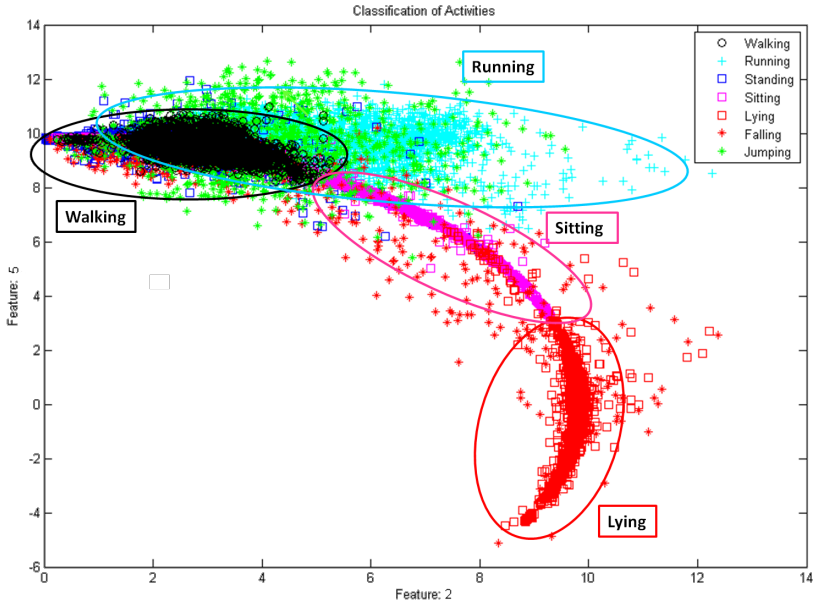


Figure 4.13: $\overline{|a_{horiz_{BF}}|}$, Feature 2, and $\overline{a_{vert_{BF}}}$, Feature 5, in a 128 samples window. Same conclusions as Figure 4.12 are extracted from the point of view of classification of activities even if feature information is different.

This feature measures the variation over $\overline{|a_{horiz_{BF}}|}$ for every activity. A window length of 128 samples is chosen in order to be a meaningful feature for all the activities considered. *Static* from non-static activities can be distinguished except for *walking*. The combination of this feature with $\sigma_{|a_{vert_{BF}}|}$ are useful for the distinction of *falling, running* and *jumping*.

4.8.2 Vertical Acceleration in the Body Frame

A total of four features were calculated from $a_{vert_{BF}}$. For a description of the information given by $a_{vert_{BF}}$, see Appendix B.

4.8.2.1 Maximum Value

Feature 4, $MAX_{a_{vert_{BF}}}$, is the maximum value of $a_{vert_{BF}}$ calculated in a window size of 128 samples:

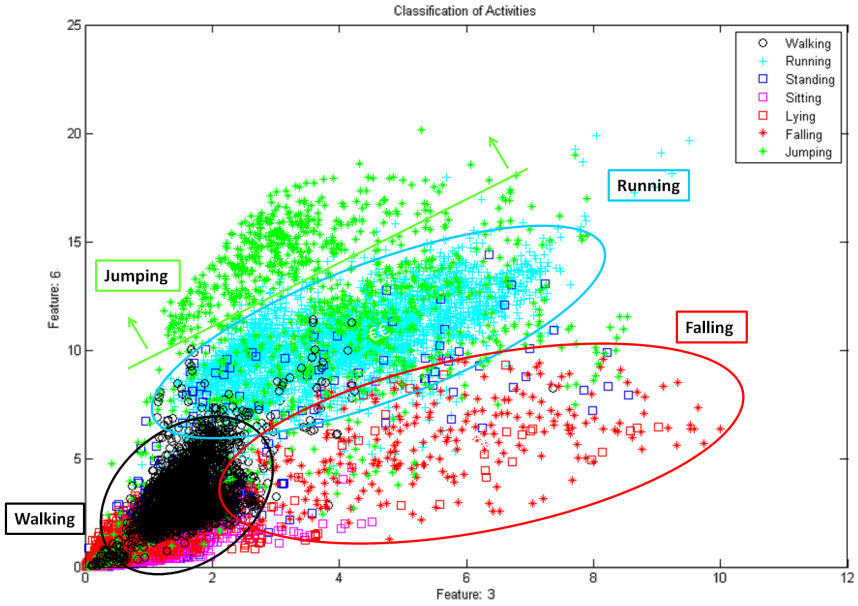


Figure 4.14: $\sigma_{|a_{horiz_{BF}}|}$ (Feature 3) and $\sigma_{|a_{vert_{BF}}|}$ (Feature 6) in a 128 samples window. Apart from their mean value, $\sigma_{|a_{horiz_{BF}}|}$ and $\sigma_{|a_{vert_{BF}}|}$ provide information about the variation of these signals over their mean value for every activity. The distinction of *static* from non-static activities (see Appendix D for more details) and *jumping* from *running* improves considering the standard deviation of these signals.

$$MAX_{a_{vert_{BF}}} = max(a_{vert_{BF}}) \tag{4.8}$$

Implying a high value of $|a_{horiz_{BF}}|$ does not imply a high value in $a_{vert_{BF}}$. In order to characterize the activities, $MAX_{a_{vert_{BF}}}$ was also taken into account. As it is used to distinguish between *jumping*, *falling* and also *walking*, a compromise window length of 128 samples was chosen (see Figure 4.12).

4.8.2.2 Mean Value

$\overline{a_{vert_{BF}}}$ is Feature 5. It consists of the mean value of $a_{vert_{BF}}$ over a window of 128 samples (see Figure 4.13 for details).

This feature is very useful for distinguishing *standing, sitting* and *lying* for the same reasons explained for $\overline{|a_{horiz_{BF}}|}$ and more reliable than $MAX_{a_{vert_{BF}}}$.

Besides, as soon as the wearer of the sensor hits the floor, a peak of the $a_{vert_{BF}}$ is measured. These peaks are higher for activities such as *jumping* and *running* than *walking* and are the cause that the mean value of this signal increases for these activities (see Appendix B for details).

4.8.2.3 Standard Deviation

Feature 6 consists of the standard deviation of $a_{vert_{BF}}$, $\sigma_{a_{vert_{BF}}}$ calculated in a window of 128 samples (see Figure 4.14). This feature in combination with $\sigma_{|a_{horiz_{BF}}|}$ is a good discriminator for activities such as *jumping, running, walking* and *falling*.

4.8.2.4 Root Mean Square (RMS)

Feature 7 is Root Mean Square (RMS) of $a_{vert_{BF}}$, $RMS_{a_{vert_{BF}}}$ for a window of 128 samples (see Figure 4.15). It is defined as:

$$RMS_{a_{vert_{BF}}} = \sqrt{\frac{\sum_{k=1}^{N}(a_{vert_{BF}}[k])^2}{N}}, \tag{4.9}$$

where N is the window length considered.

This feature is useful for distinguishing *lying* from *standing* and *sitting*.

4.8.3 Horizontal Angular Velocity in the Body Frame

Feature 8 consists of the IQR of $|w_{horiz_{BF}}|$ in a window of 128 samples (see Figure 4.16). It is denoted as $IQR_{|w_{horiz_{BF}}|}$ and it is significant and useful for *falling* as was explained in Section 4.6.1.

Gyroscope signals are usually not used for activity recognition. For activities like *walking* and *running*, accelerometers are good enough to characterize them. However, e.g. *falling* implies a fast rotation of the body and this fact is shown in Figure 4.16. This is the reason that this feature was selected. No other feature from angular velocity was chosen.

4.8.4 Vertical Acceleration in the Global Frame

Even if $\overline{|a|}$ is already used in a 32 samples window length, the mean value of the vertical acceleration, $\overline{a_{vert_{GF}}}$, for the same window length gives

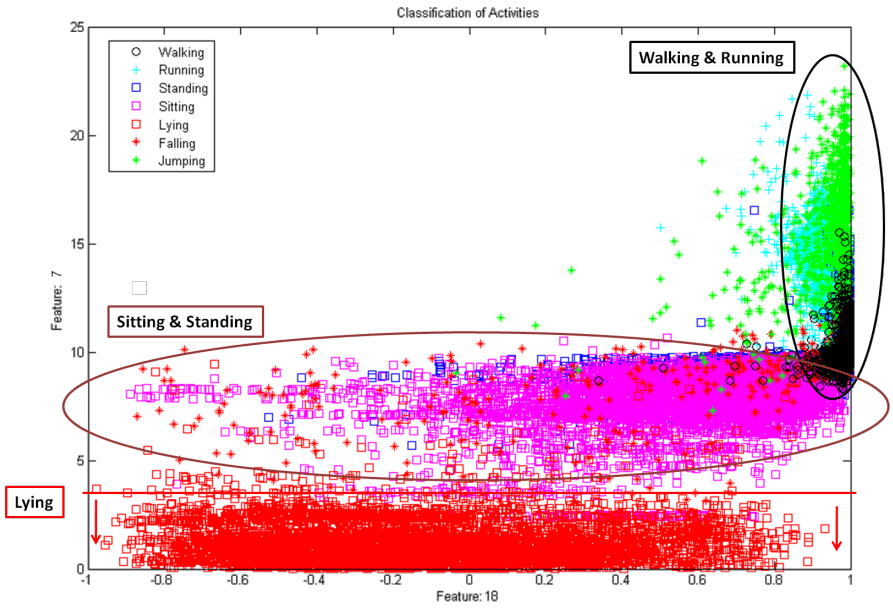


Figure 4.15: Feature 18, $\rho_{a_{vert_{BF}},|a|}$ and Feature 7, $RMS_{a_{vert_{BF}}}$ in a window of 128 samples. Feature 18 achieves high values if the current motion acceleration in 3D is mainly contained in vertical axis. Feature 7 is useful in the distinction of *lying* from other activities.

us different information for activities such as *jumping* and *running* where horizontal acceleration has a significant value (see Figure 4.17 for details).

Feature 9 is the only feature computed from $a_{vert_{GF}}$ and it was selected to reflect the *free fall* phenomenon (see Figure 4.16). If a accelerometer is free falling and measuring local vertical acceleration of *GF*, this acceleration is zero. As soon as the body is stopped, it will feel a reaction force of the surface that stopped its movement. This reaction force comes from the acceleration of the gravity field of the Earth. For the activity of *jumping*, values around zero are shown in this feature (see Appendix B) as one of the phases of *jumping* consists of a *free fall*.

The acceleration suffered starting going down after *taking impulse* and *flying up* however is stronger than gravity (see Appendix B). The reason for this behavior could be that the human body can be seen as a spring that compresses itself while *going up* and expands after producing a force while *going down* of a high value that could produce an acceleration during *going*

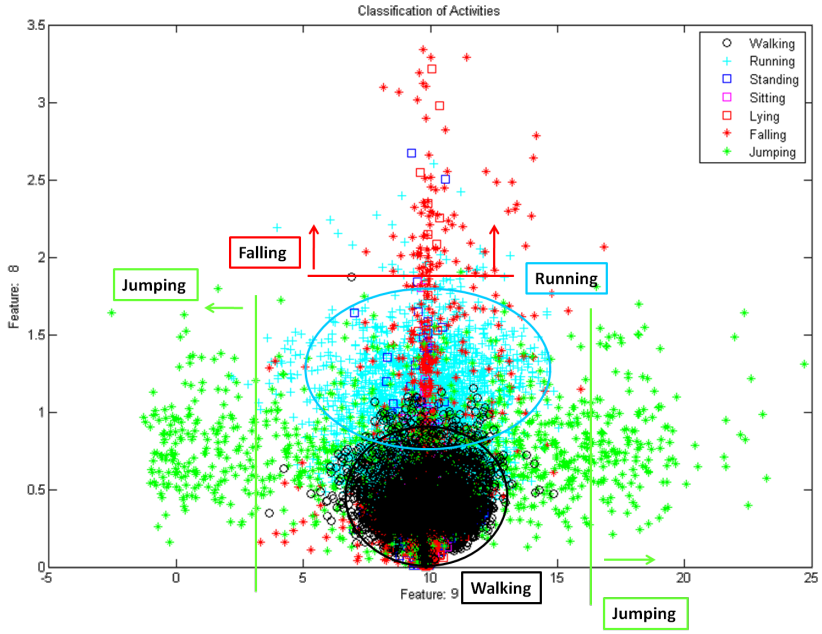


Figure 4.16: Feature 9, $\overline{a_{vert_{GF}}}$ computed over a window of 32 samples and Feature 8, $IQR_{|w_{horiz_{BF}}|}$ in a 128 samples window. Feature 9 is already shown in Figure 4.19. Feature 8 is specially useful for *falling* that implies a rotation of the body in its horizontal plane.

down lower than 0 m/s^2 that is the acceleration measured while *free falling*.

4.8.5 Norm of the Acceleration

Several features are calculated from $|a|$ measured by the IMU and described below.

4.8.5.1 Mean Value

The first feature selected from $|a|$ is $\overline{|a|}$. This feature was selected for a window length of 32 samples (Feature 10) and 512 samples (Feature 11).

In Figure 4.18 all the activities are plotted for both window lengths.

For a window length of 32 samples, the value of this feature could be compared with the instantaneous value of $|a|$. The oscillations in the value of $|a|$ that implies every activity can be observed. For example, *walking* implies

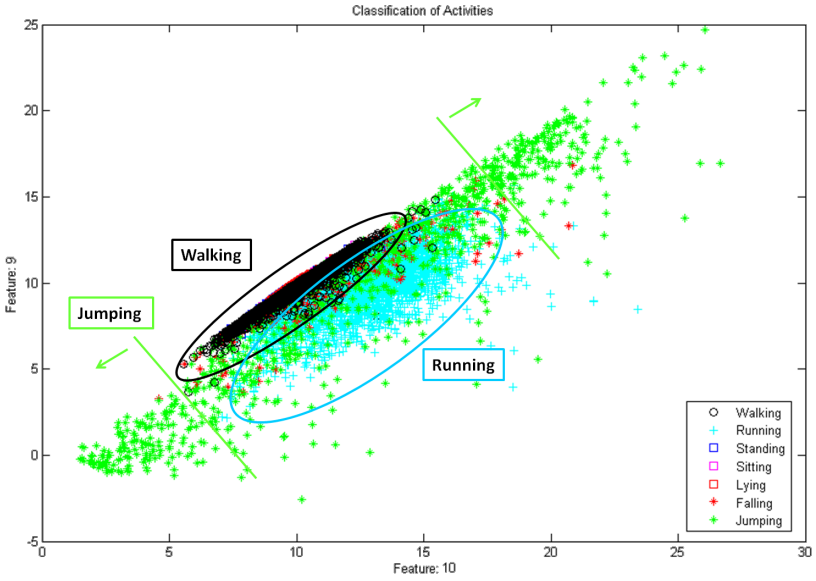


Figure 4.17: Feature 10, $\overline{|a|}$ and Feature 9 $\overline{a_{vert_{GF}}}$ both computed in a 32 samples window length. They both provide different information, specially for activities such as *jumping* and *running* where horizontal acceleration contributes significantly.

a nearly symmetrical oscillation of $|a|$ except the peaks of the acceleration measured once the floor is hit (as discussed in Appendix B).

The main information given by this feature to the system is the characterization and distinction of the activity *jumping* from the other activities. On one hand, during the rising phase, $|a|$ will decrease to increase once the impulse has been taken. But after increasing the acceleration of the body, the acceleration value decreases considerably (see Appendix B for details) during the phase of free falling. On the other hand, after falling and hitting the floor, a big peak in the acceleration due to the reaction forces measured by the sensor is detected. Both values are very characteristic for *jumping*.

The same feature calculated for a window length of 512 samples gives different information. Figure 4.12 shows how *running* could be easily distinguished from *walking*, once you get several periods of the signal (see Appendix B for more information). In fact, as described before (see Appendix B), peaks of the acceleration during *running* are much higher and increase $\overline{|a|}$ from gravity acceleration to more than 11 m/s^2. *Walking* can be distinguished from

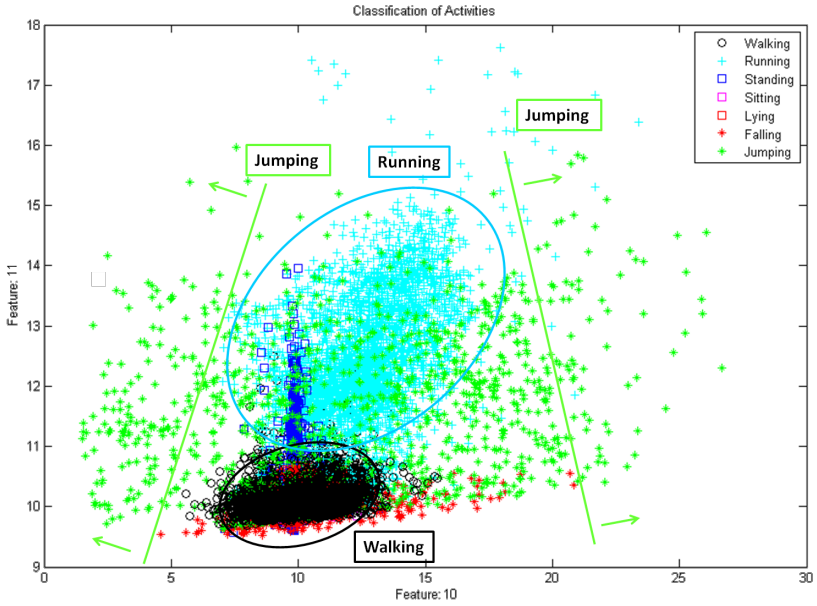


Figure 4.18: $\overline{|a|}$ for 32 samples (Feature 10) and 512 samples (Feature 11). Feature 10 is meaningful for short-time activities as *jumping* and *falling*. In contrast, Feature 11 can distinguish clearly between *walking* and *running*.

static activities like *sitting, standing* or *lying* as well (see Appendix D for details) due to the peaks of the acceleration measured once the floor is hit that increase lightly $\overline{|a|}$ over gravity value.

4.8.5.2 Standard Deviation

Feature 12 is the standard deviation of $|a|$ in a 256 samples window. It is used to distinguish between *static* and *dynamic* activities calculated in a window length of 256 samples. $\overline{|a|}$ is not robust enough to distinguish between *static* activities and *walking*. However, $\sigma_{|a|}$ is a good parameter to see the variation of the signal respect to the mean value, key for this distinction. Choosing 256 samples, enough information from *walking* and *running* could be obtained and it can assumed that a person will stand still or sit more than 2.56 seconds as well (see Figure 4.19).

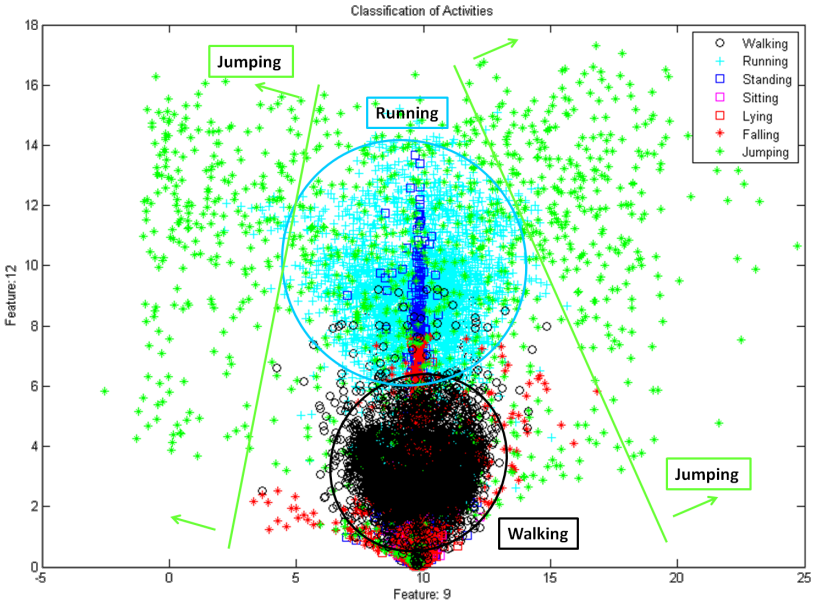


Figure 4.19: $\overline{a_{vert_{GF}}}$ in a 32 samples window (Feature 9) and $\sigma_{|a|}$ over a 256 samples window (Feature 12) plotted for all the activities. As commented before, Feature 9 is meaningful specially for *jumping* and for reflecting *free falling* phenomenon. Feature 12 gives information about *static* activities, *walking, running* and *jumping* repeatedly (see Appendix D for more examples).

4.8.5.3 Interquartile Range

Feature number 13 is calculated in a 128 samples window and gives the IQR of $|a|$:

$$IQR_{|a|} = Q_3 - Q_1 \ , \tag{4.10}$$

where Q_1 is the 25th-percentile and Q_3 is the 75th-percentile [Unit: m/s^2].

To obtain these values, the signal is sorted in ascendant order. Q_1 is defined as the value x that divides the ordered signal such that 25% of the samples are smaller than itself. Q_3 is defined as the value x that divides the ordered signal such that 75% of the samples are smaller than itself, so 25% are greater.

This feature (see Figure 4.20) can be used for distinguishing between *jumping* and *falling*: *falling* can be performed only once (after you have to

get up and fall again), but *jumping* can be done repeatedly. Consequently, very high values of $|a|$ because you hit the floor several times and low values produced after you push yourself up are contained in the same window (see Appendix B for further details). These are the reasons why this feature has higher values for *jumping* than for *falling*.

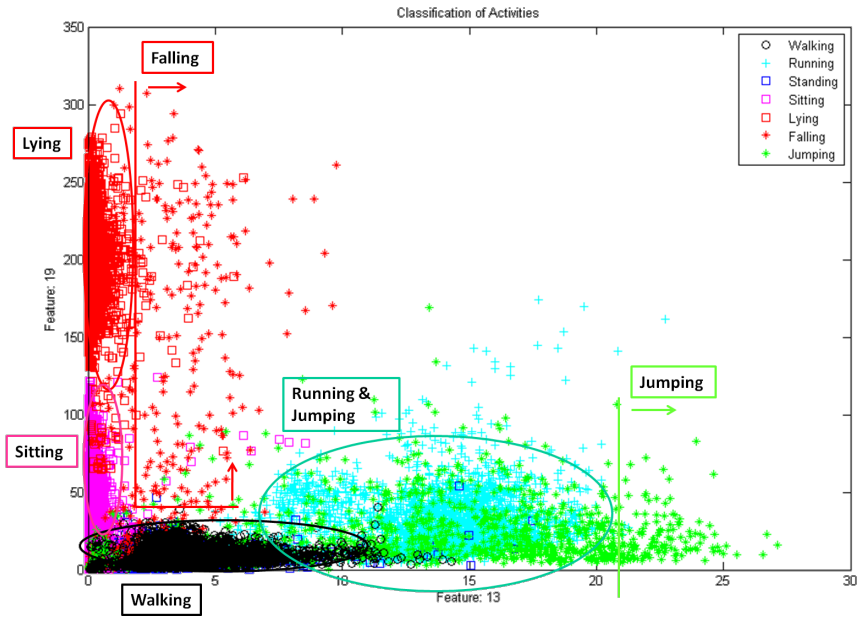


Figure 4.20: $IQR_{|a|}$ (Feature 13) in a 128 samples window and $att_{|a_{horiz_{BF}}|,a_{vert_{BF}}}$ (Feature 19) over a window of 64 samples are shown for all the activities. It can be observed that *lying* and *falling* appear in the upper part of the plot as they imply a radical change in the attitude of the sensor respect *standing*. *Sitting* can be also distinguished in the middle part of the plot if it implies a variation of the attitude of the sensor respect *standing* too. Besides, *static* activities present a low value of Feature 11 as expected and contrary to *jumping* and *running*.

4.8.5.4 Main Frequency Component

Feature number 14 is defined as the main frequency component of $|a|$, $MFC_{|a|}$, computed over a window size of 128 samples.

Fast Fourier Transform (FFT) is an efficient implementation of the Discrete Fourier Transform (DFT). DFT is used for numerical computation in

digital signal processing as it allows to represent a discrete time domain signal in frequency domain (see [68]).

The algorithm to compute this feature is described below:

- Compute the mean value of $|a|$ over the 128 samples window.

$$\overline{|a|_{128}} = \frac{1}{N}\sum_{k=1}^{N}|a|\,[k] \text{ with } N = 128 \tag{4.11}$$

- Extract $\overline{|a|}$ computed before.

$$|a|\,[k]_{AC} = |a|\,[k] - \overline{|a|_{128}} \tag{4.12}$$

- Multiply by a window of 128 samples. Each window function has its advantages and disadvantages [69]. In this case, the amplitude accuracy of a single frequency component is not more important than the exact location of the component in a given frequency bin [70]. According to this requirement, the possible options are *Hann* or *4-term Blackman-Harris* window [71]. In general, the Hann window is satisfactory in 95% of cases [70] as it has good frequency resolution and reduced spectral leakage [71].

  So, the window used is Hann window of length 128 defined as:

$$w[n] = 0.5(1 - cos(\frac{2\pi n}{N-1})) \text{ with } 0 \leq n \leq M-1, \tag{4.13}$$

  where N is the window length and M is $\frac{N}{2}$ for N even and $\frac{N+1}{2}$ for N odd.

  The second half of the symmetric Hann window is obtained by flipping the first half around the midpoint [1].

- Use zero padding to sample better the spectrum (see [68] for further details). The length of the final signal is four times the length of the original one.

[1] Extracted from http://www.mathworks.com/

- Apply Fast Fourier Transform (FFT) algorithm in order to get the coefficients of the DFT, defined as (see [68]):

$$X[k] = \sum_{k=0}^{N-1} x[n] e^{-j2\pi kn/N} \; k = 0, ..., N-1 \; , \tag{4.14}$$

  where

  $X[k]$ are the DFT coefficients
  $x[n]$ is the discrete time signal
  N is the length of $x[n]$

- Compute the absolute value of the coefficients. The output coefficients are symmetric and therefore only the first half has to be taken into account.

- Find the maximum value.

- Compute the continuous frequency:

$$f_c = f_s \frac{k}{L} \; , \tag{4.15}$$

  where

  f_c is the frequency [Unit: Hz]
  f_s is the sample frequency [Unit: Hz]
  k is the index of the DFT coefficient
  L is the total length of the FFT

During this work, interesting results were achieved, specially from activities as *walking* and *running*: the main frequency component of both activities, the fundamental frequency (as both are periodical activities) is within the frequency band from 1.6 to over 3 Hz. For the characterization of both activities, this feature should be computed at a high window length to get several periods of the signal.

However, this feature is mainly used to distinguish activities like *falling* and *jumping* from *running* (Figure 4.21). Due to this fact, a window length of 128 samples was chosen in order to get enough periods of running, but also a short window length in which we can rely for short activities like *jumping*

and *falling*. Choosing the shortest possible window, it can be mainly assured that the recognition of these activities will not be extremely dependent on our data set.

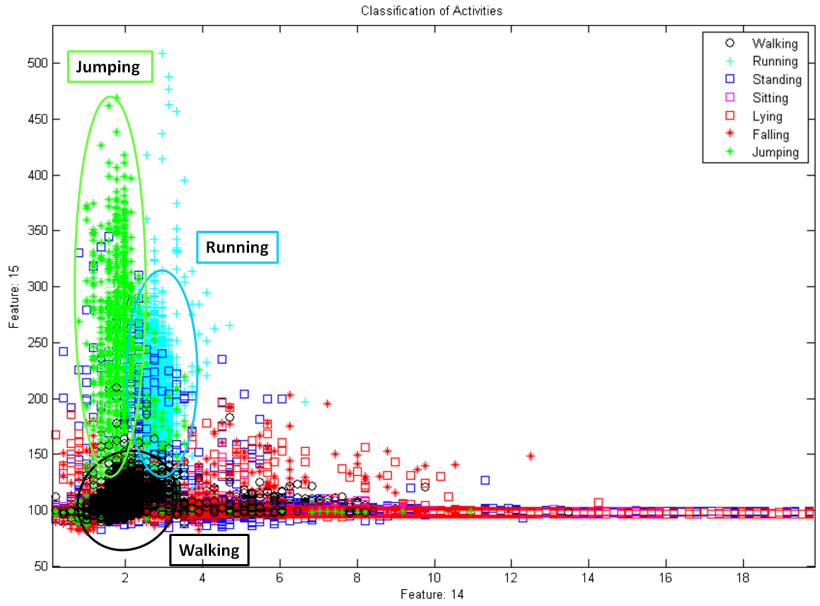


Figure 4.21: Main frequency component of $|a|$ abbreviated as $MFC_{|a|}$ (Feature 14) and energy of $|a|$ below 2.85 Hz, referenced as $\hat{E}(|a|_{LPF\ 2.85\ Hz})$ (Feature 15) for all the activities. Approximately, the main frequency component of $|a|$ during *walking* is around 2 Hz. However, the main frequency component of $|a|$ during *running* is around 3 Hz. This feature is very useful for distinguishing *jumping* from *running* as *jumping* repeatedly is done at other frequency than *running*. With respect to Feature 15, overlapping of the main frequency component of *walking* with *jumping* and *running* is avoided due to *jumping* and *running* energy in this frequency band is clearly bigger than *walking*.

4.8.5.5 Energy of the Signal in some Frequency Bands

Energy of the signal can be computed in frequency or time domain using Parseval's theorem [67]. In this Master Thesis, Finite Impulse Response (FIR) filters were used to compute the energy of the signal in some frequency bands. These filters are not ideal, so it cannot be claimed that the values of energy represented in the plots and the features are the energy value of

the signal in the frequency bands defined in this section because there is a transition band given by the frequency response of the filter that should be taken into account. As the objective of this Master Thesis is not related to biomechanics, but to obtain features that could characterize and distinguish activities in real-time, it is preferable to use low-order filters instead of very precise ones.

The algorithm and details of implementation of this feature are described below:

- Obtain the coefficients b_i of the Finite Impulse Response (FIR) filter using the Parks-McClellan algorithm [2]. As energy of the signal is computed in temporal domain, time alignment of the filtered signal is desired. Linear phase of FIR filters for the band pass is assured using this method so frequency distortion of the signal is not produced.

- Filter the signal using the difference equation:

$$y[n] = b_0 x[n] + b_1 x[n-1] + ... + b_N x[n-N] \ , \tag{4.16}$$

  where

  $x[n]$ is the input signal, in this case, $|a|$
  $y[n]$ is the output signal
  b_i are the coefficients of the filter calculated before
  N is the order of the filter

- Compute the energy normalized to the window length of the output signal:

$$\hat{E} = \frac{1}{N} \sum_{k=1}^{N} (|x[k]|^2) \tag{4.17}$$

As explained above, human motion related activities frequency content of measured acceleration is below 10 Hz (see [47]). The acceleration norm measured during the performance of the activity *walking* has most of its energy in its first seven harmonics. Taking into account a fundamental frequency around 1.6 to 2 Hz, the energy of the signal will remain below 10 Hz. For

[2] See http://www.mathworks.com/

running, the fundamental frequency is higher, around 2.5 Hz to over 3 Hz. FIR filters used for computing features 15, 16 and 17 are shown in Figures 4.22 and 4.23.

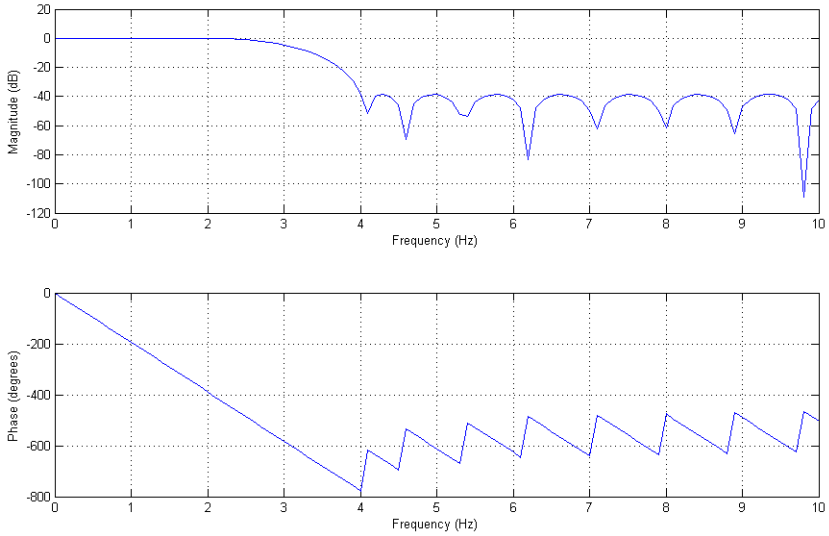


Figure 4.22: Lowpass filter used for obtaining the energy of the signal below 2.85 Hz. The order of the filter is 109. The finite impulse response filter response is plotted. As can be observed, an attenuation of 3 dB is presented at 2.85 Hz while the attenuation value at 4.05 Hz is 50 dB. Phase response is linear with frequency in the bandpass so no frequency distortion is produced in the signal.

Feature 15 is the energy of the filtered $|a|$ by FIR filter whose characteristics are represented in Figure 4.22, $\hat{E}(|a|_{LPF\ 2.85\ Hz})$, for a window length of 128 samples. It will be referenced in this Master Thesis as the energy of $|a|$ for frequencies lower than 2.85 Hz. This feature is able to distinguish *walking* from *jumping* and *running* as the energy of this activity is lower than the others (see Figure 4.21 for details).

Features 16 and 17 corresponds to the energy of $|a|$ for a frequency band from around 1.6 to 4.5 Hz, $\hat{E}(|a|_{BPF\ 1.6-4.5\ Hz})$, and filter used is shown in Figure 4.23. Window length considered for these features are 64 and 512 respectively. On one hand, as can be seen in Figure 4.24, the energy of this signal for a window of 64 samples is able to give information in order to distinguish *jumping* from *walking* and *running*. On the other hand, considering

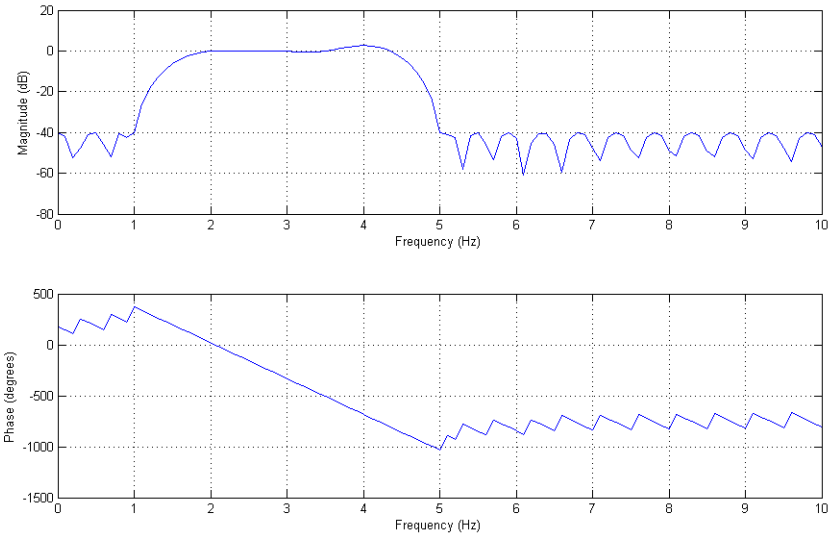


Figure 4.23: Bandpass filter used for obtaining the energy of the signal between 1.6 and 4.5 Hz. The order of the filter is 196. The finite impulse response filter response is represented. An attenuation of 3 dB is presented at 1.6 and 4.5 Hz while the attenuation value at 1 and 5 Hz is 50 dB. Phase response is linear with frequency in the bandpass so no frequency distortion is produced in the signal.

a window of 512 samples, long-time activities such as *walking* from *running* and *jumping* repeatedly can be distinguished.

4.8.6 Correlation between Signals

This feature is the correlation coefficient between $a_{vert_{BF}}$ and $|a|$ in 128 samples window size:

$$\rho_{a_{vert_{BF}},|a|} = \frac{\overline{a_{vert_{BF}} \cdot |a|} - \overline{a_{vert_{BF}}}\ \overline{|a|}}{\sigma_{a_{vert_{BF}}} \sigma_{|a|}} \tag{4.18}$$

It is observed that $|a|$ for activities such as *walking* are mainly influenced by the acceleration produced in the vertical axis of body frame (BF). This feature helps in the characterization of human motion related activities pointing out if the subject's motion or acceleration is mainly contained in the vertical axis of BF.

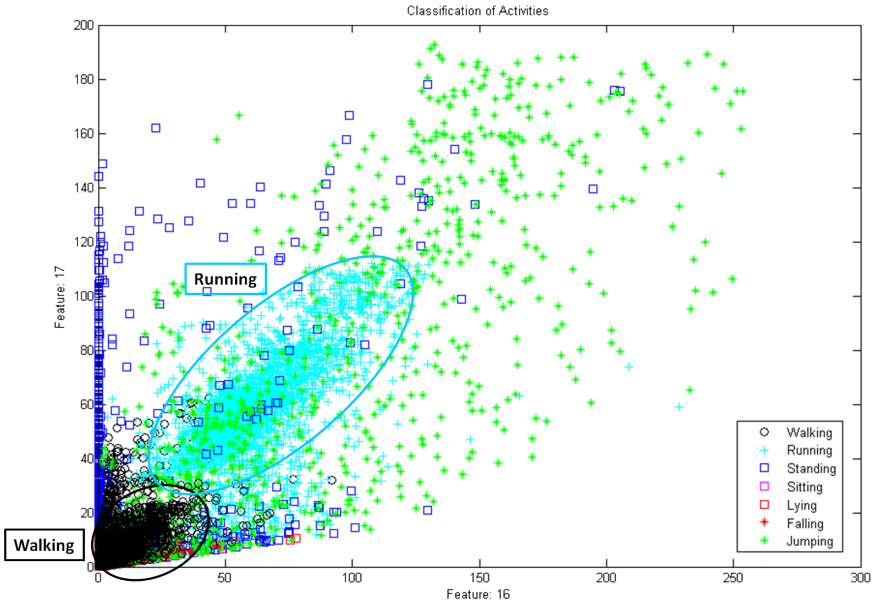


Figure 4.24: Energy of $|a|$ in the frequency band from around 1.6 to over 4.5 Hz for 64 (Feature 16) and 512 (Feature 17) window length. In this frequency band, the main frequency component of $|a|$ during *walking, jumping* and *running* is contained. A window size of 64 samples is chosen in order to get very low frequency information about short time activities (*jumping* and *falling*). It can be observed that Feature 16 is not enough to distinguish *walking* and *running* so 512 samples window length is used for long-term activities.

While *walking* and *running* lead to high values (the absolute value of the correlation coefficient is bigger than 0.9) at a 128 samples window size, *falling* and *static* activities do not behave in the same way: it can take any value (as can be seen in Figure 4.15).

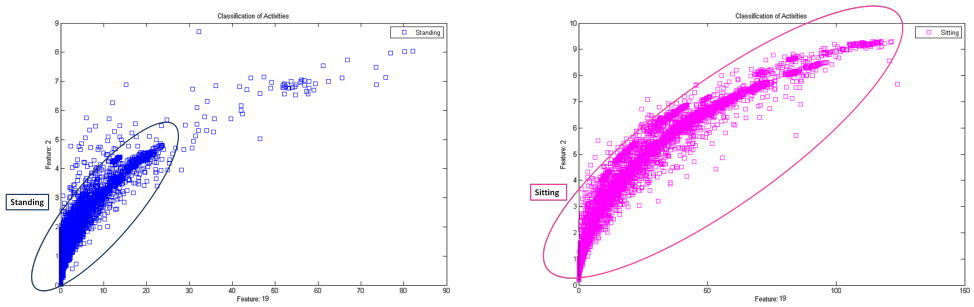
4.8.7 Attitude of the Sensor

Every feature computed until now from $|a_{horiz_{BF}}|$ and $a_{vert_{BF}}$ gives information about body attitude during the performance of the activity as they belong to the BF. However, this information can be summarized in one feature that will use the interaction of the gravity field of the Earth in the horizontal plane and the vertical axis to fuse this information.

Feature 18 (see Figure 4.20), $att_{|a_{horiz_{BF}}|,a_{vert_{BF}}}$, is defined in a 64 samples window size and calculated using the variation of the $\overline{|a_{horiz_{BF}}|}$ and $\overline{|a_{vert_{BF}}|}$ between the current activity and the initial activity *standing*:

$$att_{|a_{horiz_{BF}}|,|a_{vert_{BF}}|} = (\Delta\overline{|a_{horiz_{BF}}|})^2 + (\Delta\overline{|a_{vert_{BF}}|})^2 \tag{4.19}$$

One of the most difficult distinctions between our activities are *standing* from *sitting* (see Figure 4.25). Thanks to assumption number 2 (see Section 4.1), the position of the sensor during *standing* is known and sensor measurements can be rotated to BF. However, a person could sit in such a position that the sensor attitude is the same for *standing*. In such situations, not considering transitions and just checking sensor attitude, it is not possible to distinguish them.



(a) (b)

Figure 4.25: This figure shows the classification problem between (a) *standing* and (b) *sitting* plotting Feature 19, $att_{|a_{horiz_{BF}}|,a_{vert_{BF}}}$ in a window of 64 samples and Feature 2, $\overline{a_{horiz_{BF}}}$ over a window of 128 samples. As soon as the wearer of the sensor is sitting in such a position that the body attitude where the sensor is located is similar to *standing*, both activities are confused. Studying the transitions between both is required in order to improve the inference of both activities.

4.8.8 Final Set Observations

The distinction of the activities is not trivial. The main effort during this feature identification process was to try to characterize the activities with features that are meaningful in real world and physics, as well as to include

the features that will distinguish them. See Table 4.7 for an evaluation of the most useful features for every activity.

Static activities such as *standing*, *sitting* and *lying* can be distinguished from non-static activities like *walking, running, falling* or *jumping*. *Lying* implies a complete change in the attitude of the sensor, so it could be also distinguished from *standing* or *sitting*. The main challenge is to be able to distinguish *standing* from *sitting*.

The only feature that can be used for both activities is related to the attitude of the sensor. This attitude depends, for example, on the type of chair you use or how the person sits usually. If the person sits in chair with a straight back and the sensor is on the belt, far away from the join between your leg and the hip, this situation will not change the attitude of the sensor and *sitting* may be confused with *standing*.

For activities like *walking* and *running*, even if human motion is quite complex and dependent on the person, they can be characterized easily thanks to the acceleration that your body feels in every plane of the human body near the center of gravity. The main frequency component of $|a|$ and the energy of the signal around this frequency component are good features to differentiate *running* from *walking, jumping* or *falling*.

Activities such as *jumping* and *falling* are more difficult to characterize. They do not follow always a similar pattern and their feature values depend on many variables like, for example, the person, the clothing, the context or the mood. There are different phases as well in the performance of these activities whose values are completely different for most of the features.

In order to distinguish *falling* from *jumping*, mainly the attitude during and after landing on the ground is the clue.

Different window lengths are required to get robust features for the different activities. If it would not, the classification and recognition of the activities would be highly dependent on your data set. Even if one feature in a 512 samples window seems good for activities such as *falling*, it is not reliable as *falling* is a quick activity that takes around one second.

4.9 Recognition Algorithm

The following sections explain and justify the design of the final recognition algorithms including the results obtained and problems faced.

The aim of this section is to design the recognition algorithm and classifier

Features			Activities						
No.	Description	Window size	Standing	Sitting	Lying	Walking	Running	Jumping	Falling
1	$MAX_{\lvert a_{horiz_{BF}} \rvert}$	128	×	✓	✓	×	✓	✓	✓
2	$\overline{\lvert a_{horiz_{BF}} \rvert}$	128	✓	✓	✓	×	×	×	×
3	$\sigma_{\lvert a_{horiz_{BF}} \rvert}$	128	✓	✓	✓	×	×	×	×
4	$MAX_{a_{vert_{BF}}}$	128	×	✓	✓	×	✓	✓	×
5	$\overline{a_{vert_{BF}}}$	128	✓	✓	✓	×	×	×	×
6	$\sigma_{a_{vert_{BF}}}$	128	✓	✓	✓	✓	✓	✓	×
7	$RMS_{a_{vert_{BF}}}$	128	×	×	✓	×	×	×	×
8	$IQR_{\lvert w_{horiz_{BF}} \rvert}$	128	×	×	×	×	×	×	✓
9	$\overline{a_{vert_{GF}}}$	32	×	×	×	×	×	✓	×
10	$\overline{\lvert a \rvert}$	32	✓	✓	✓	×	×	✓	✓
11	$\overline{\lvert a \rvert}$	512	✓	✓	✓	✓	✓	×	×
12	$\sigma_{\lvert a \rvert}$	256	✓	✓	✓	✓	✓	×	×
13	$IQR_{\lvert a \rvert}$	128	✓	✓	✓	✓	✓	✓	×
14	$MFC_{\lvert a \rvert}$	128	×	×	×	✓	✓	✓	×
15	$\hat{E}(\lvert a \rvert_{LPF\ 2.85\ Hz})$	128	✓	✓	✓	✓	✓	✓	×
16	$\hat{E}(\lvert a \rvert_{BPF\ 1.6\text{–}4.5\ Hz})$	64	✓	✓	✓	×	×	✓	×
17	$\hat{E}(\lvert a \rvert_{BPF\ 1.6\text{–}4.5\ Hz})$	512	✓	✓	✓	✓	✓	✓	×
18	$\rho_{a_{vert_{BF}},\lvert a \rvert}$	128	×	×	×	✓	✓	×	×
19	$att_{\lvert a_{horiz_{BF}} \rvert, a_{vert_{BF}}}$	64	✓	✓	✓	×	×	×	×

Table 4.7: Evaluation of the features in function of the activities. The most significant features for every activity are specified.

that will provide the current activity given the current features. However, the activity that a subject is doing depends on the activity that the subject performed before. This information is wasted if the recognition algorithm does not include it.

4.9.1 Selection of Classifier

In order to select a classifier, Chapter 2 gives a review of well-known classifiers that could be used for this project.

The first step is analyzing the nature of the data that will be given to the classifier. As can be seen in Section 4.8, it can be considered that there are *noise* and *outliers* in the collected data due to two main reasons:

- The nature of the processing where the feature extraction is performed through a window that contains signal samples of past activities, not only the current one as the signal is continuously being processed. This window could have a maximum of 512 samples, enough to contain information from different past activities.

- Labeling of the data has been done manually. There is not an accurate method that provides you the exact sample where the subject started to fall.

Activity recognition should be performed in real-time, so the speed of the classification is an important characteristic to be considered. The main goal is to be able to integrate this system in a single chip that does not require a high computational power or memory, so the storage and the inference complexity should be minimum. Also, feature reduction and the selection of the final features is performed by expert knowledge and could be desirable that the classifier is robust to irrelevant or redundant features. Finally, the amount of data of every activity is not equal as there are more instances of activities such as *walking* than *falling*.

K-Nearest Neighbor (kNN), for example, is usually considered intolerant to noise, has a high storage, not suitable for real-time applications as the speed of the classification is low and is sensitive to irrelevant features [51].

Support Vector Machines (SVMs) and Artificial Neural networks (ANNs) have similar disadvantages for this application. Both have to deal with danger of overfitting the data, so the classifier will be sensitive to *outliers* or *noise*

unless the training is not well adjusted. In SVMs, as the data set is not linearly separable, a kernel function should be used and choosing this function and the parameters of the model is not trivial. For ANNs, the structure of the network should be predefined and to define these parameters is not easy as well. Besides, the interpretation of SVMs and ANNs results is notoriously difficult [51]. Finally, the tolerance to noise is not very high.

To sum up, an analysis of principal components is desired previously in order to avoid or quit irrelevant features even if ANNs could do it through the adjusting of the synaptic weights of the network.

Decision Trees have the problem of overfitting the data, so control on the training should be done in order to keep it robust to noise and outliers.

Finally, statistical learning algorithms are preferred for this application such as Naïve Bayes or Bayesian networks as they present several advantages that make them appropriate for activity recognition:

1. Discrete Bayesian networks requires little storage space as it only needs to the conditional probability table (CPT) of every node.
2. Bayesian networks gives good results in many applications.
3. They may need a relatively small dataset [51].
4. They are very transparent and it is easier for human reasoning to interpret the results. So the link to physical world is not lost.
5. It is also suitable for real-time applications as the speed of classification is high under the constraint that there is evidence in all nodes.
6. They are very tolerant to noise as they use the probability distribution of the data.
7. No problem of overfitting has been observed in these algorithms.

Finally, discrete Bayesian networks will be chosen.

4.9.2 Feature Discretization

The feature discretization process tries to identify meaningful states of these features. To identify these states, histograms and plots in 2D of the features have been used. The algorithm followed for the feature discretization consists of:

1. Identifying the activities the feature is appropriate for.

2. Noticing *meaningful* changes in the histograms comparing to other of these activities.

3. Defining the final intervals to discretize the features. Zooming in and observing histograms at different resolutions is required.

Figure 4.26 shows an example of the process of discretization. The discretized feature is the main frequency component of $|a|$. First, the activities for which this feature is meaningful and adequate are *walking, running* and *jumping*. Histograms and plot in 2D of the feature provides four states and intervals of these states are done using the histograms and the plots at different resolutions.

To evaluate how important the discretization process is for inference, another algorithm for the feature discretization has been applied and results will be evaluated. Also, different discretizations for every algorithm described in this section have been designed.

A process of discretization of the features in 2D joining features of $|a_{horiz_{BF}}|$ and $a_{vert_{BF}}$ or other 2D combinations that provides good clustering for the activities could be useful. So, a second algorithm of the process of discretization is defined. The steps to follow are:

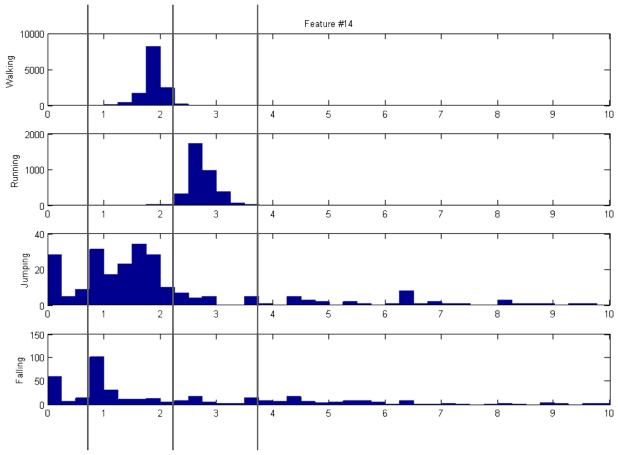
1. Identifying the activities the feature is appropriate for.

2. Noticing *meaningful* areas in the plots comparing to other of these activities (see Figure 4.27 (b) for a graphical example of this process).

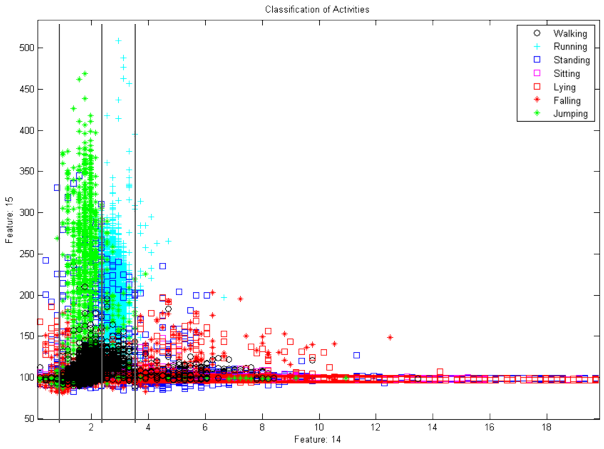
3. Defining the final intervals that will discretize the features. Zooming in and observing histograms and 2D plots at different resolutions is required.

Both algorithms give different results as shown in Figure 4.27. A total of four feature discretizations will be done:

- Two discretizations of the features following the first algorithm. More relevant states will be chosen. A maximum of 9 and a minimum of 4 states were identified. The will be referred in this report as *1Da* and *1Db*.



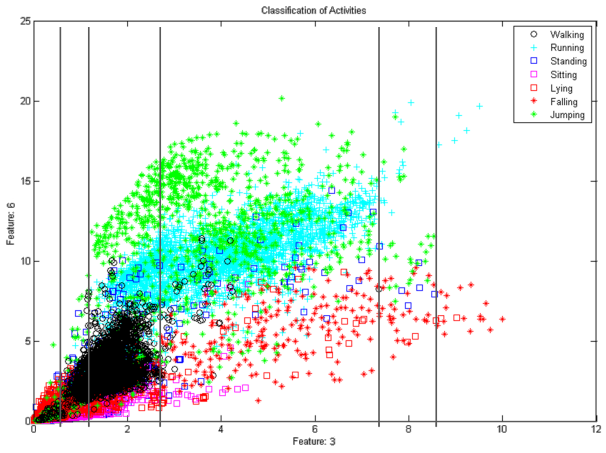
(a)



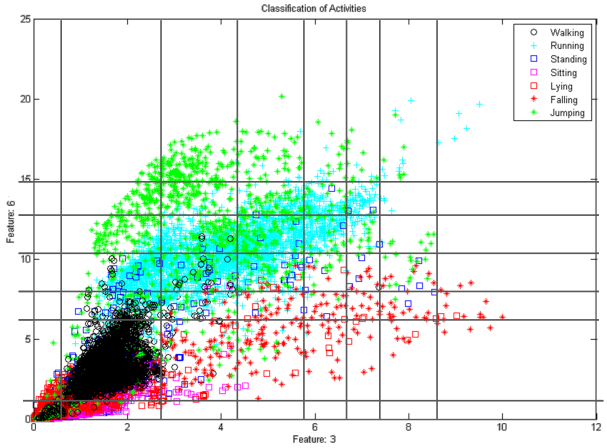
(b)

Figure 4.26: Feature discretization example. The discretized feature is the main frequency component of $|a|$. The activities for which this feature is appropriate are *walking, running* and *jumping*. Four states are identified. Intervals of these states are done zooming into the histograms (a) and the plots of a pair of features (b).

- A third discretization of the features using the second algorithm. A maximum of 14 states and a minimum of 4 states were identified. The identifier *2Da* will be assigned to this discretization.
- A fourth discretization limiting the number of states using the second



(a)



(b)

Figure 4.27: Comparison of possible results of the algorithms to discretize the features. Feature 9, $\sigma_{|a_{horiz_{BF}}|}$ states are identified using both algorithms. (a) Individual discretization of feature 9. (b) Discretization in 2D of features 9 and 10.

algorithm. Again, more relevant states should be selected. A maximum of 10 states are defined. This discretization will be called *2Db*.

These discretization of the features should be evaluated.

4.9.3 Naïve Bayes for Activity Recognition

Naïve Bayes is a statistical algorithm whose structure for activity recognition is shown in Figure 4.28. It assumes that all the features computed are independent (see Section 3.3.2.5). CPTs of all the nodes of this Bayesian network can be learnt from the data set.

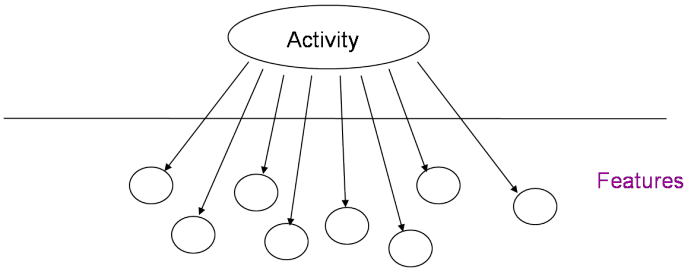


Figure 4.28: Naïve Bayes approach for activity recognition. The activity that the user is performing is the cause of the observation of the features.

The information required from Naïve Bayes is the most probable activity i given the current set of features $f_1, f_2, ...f_M$:

$$arg_i max(p(act^i|f_1, f_2...f_M)) \ , \tag{4.20}$$

where act^i is the activity (*jumping, falling, walking...*), f_i is the feature number i with $1 \leq i \leq M$ and M is the total number of features considered.

Using Bayesian theorem, this probability can be written as follows:

$$\begin{aligned} p(act^i|f_1, f_2...f_M) &= \frac{p(act^i, f_1, f_2...f_M)}{p(f_1, f_2...f_M)} \\ &= \frac{p(f_1, f_2...f_M|act^i)p(act^i)}{p(f_1, f_2...f_M)} \ , \end{aligned} \tag{4.21}$$

As $p(f_1, f_2...f_M)$ is constant for all the activities,

$$\begin{aligned} arg_i max(p(act^i|f_1, f_2...f_M)) &= arg_i max(p(act^i, f_1, f_2...f_M)) \\ &= arg_i max(p(f_1, f_2...f_M|act^i)p(act^i)) \ , \end{aligned} \tag{4.22}$$

So, the activity that maximizes $p(act^i|f_1, f_2...f_M)$ is the activity that maximizes $p(f_1, f_2...f_M|act^i)p(act^i)$ where $p(f_1, f_2...f_M|act^i)$ is learnt using the data set as data set contains the value of the features for every activity

and $p(act^i)$ is a prior probability set by expert knowledge. Using the chain rule as explained in Section 3.4.1 for Naïve Bayes network where *activity* CPT contains prior probabilities, $p(f_1, f_2...f_M|act^i)$ is computed as:

$$p(f_1, f_2...f_M|act^i) = \prod_{j=1}^{M} p(f_j|act^i) \ , \tag{4.23}$$

And joint probability distribution in this causal network is computed as:

$$p(act^i, f_1, f_2...f_M) = \prod_{j=1}^{M} p(f_j|act^i)p(act^i) \ , \tag{4.24}$$

Consequently, taking into consideration that all features are observed, there is no need of propagation of evidences and inference algorithm (as was briefly explained in Section 3.4.3.1) consists in the following:

- Set priors probabilities of *activity* according to expertise knowledge of real world human motion. The prior distribution of the node *activity* was set to the values given in Table 4.8.
- Compute joint probability distribution $p(act, f_1, f_2...f_M)$ for all the activities considered.
- Find the activity that maximizes $p(act, f_1, f_2...f_M)$.

Priors probabilities of the activities for the static approach							
Sitting	Standing	Walking	Running	Jumping	Falling	Lying	Up & Down
0.195	0.2435	0.409	0.001	0.001	0.0005	0.14	0.01

Table 4.8: Priors probabilities of the node *activity*. They are set according to expertise knowledge of real world human motion.

4.9.4 Unrestricted Bayesian Network for Activity Recognition

Naïve Bayes works under the hard assumption that features computed are independent. However, this assumption is not always fulfilled. The following subsections will explain the hypothesis and characteristics of training and using an unrestricted Bayesian network for activity recognition.

4.9.4.1 Hypothesis

One hypothesis for this approach is shown in Figure 4.29. The activity that the wearer of the sensor is doing provokes an acceleration, turn rate and attitude in different parts of the body that are interconnected. The sensor is located in the hip measuring a turn rate produced by the arm that moves the trunk or a turn rate produced by the foot that moves the hip and the trunk. These are hidden nodes of the causal network learnt for activity recognition. These random variables and nodes will not be learnt or measured by any data set. So, the consideration of the dependencies of the features is required.

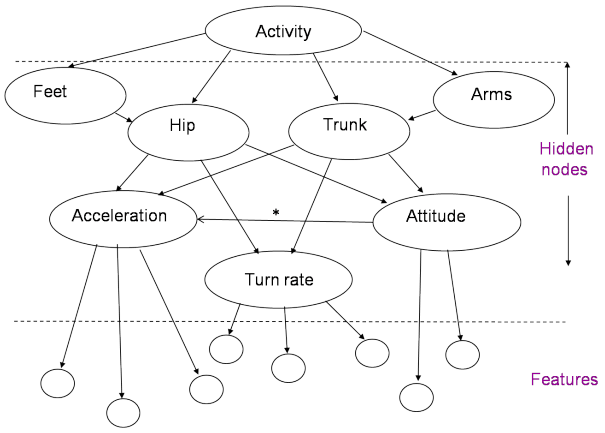


Figure 4.29: Hypothesis for unrestricted Bayesian network approach. Consideration of the dependencies of the features is required because hidden nodes in the network due to the location of the sensor and the origin of the features exist.

4.9.4.2 Structure, Markov Blanket and *D*-Separation

The structure of this network is learnt using a Greedy Hill Climber with Random Restarts based on the Cooper and Herskovits Log score for fully observed data sets and Dirichlet distributions of the conditional probability tables implemented (see references [54] and [55] for more information). This learning engine was modified to impose causality in the network as *activity* is always a parent of features and to limit the number of parents of every node (see Section 3.4.1 for more details) during the realization of this Master Thesis.

On the one hand, the Naïve Bayes structure is imposed by definition of Naïve Bayes. On the other hand, the structure of this unrestricted Bayesian network is learnt using the training algorithm described above. Learning time for all the unrestricted Bayesian networks that will be presented in this work was set to a maximum of 5 days and the maximum number of parents limited to 5. The starting network for the training algorithm has been set to the Naïve Bayes structure, but the final structure of the network could take out some of the features of the Markov Blanket (MB) of *activity* as explained in Figure 4.30.

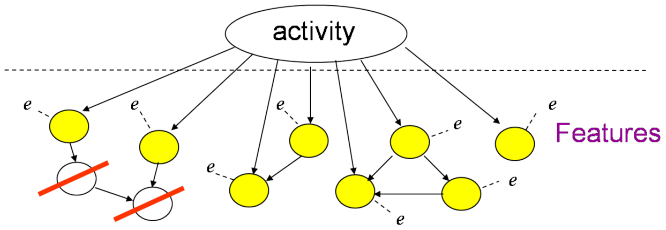


Figure 4.30: Unrestricted Bayesian network structure. The starting network structure has been set to Naïve Bayes structure. Causality of activity as the parent of all nodes has been imposed. Typically 10^8 - 10^9 network structures searched (1 - 5 days). Maximum number of parents for every node have been limited. The Markov blanket of the node *activity* is shown in yellow. As all features have got evidence, inference is very simple: computation of joint probability distribution for every activity without the propagation of evidences is required. Unnecessary, redundant or bad features have been taken out of the Markov blanket of *activity*.

4.9.4.3 Inference Algorithm

Using the chain rule as explained in Section 3.4.1 for Bayesian networks where *activity* CPT contains prior probabilities, $p(f_1, f_2...f_M|act^i)$ is computed as:

$$p(f_1, ...f_M|act^i) = \prod_{j=1}^{M} p(f_j|Parents(f_j)) \ , \tag{4.25}$$

And joint probability distribution in this causal network is calculated as:

$$p(act^i, f_1, f_2...f_M) = \prod_{j=1}^{M} p(f_j|Parents(f_j))p(act^i) \ , \tag{4.26}$$

Inference algorithm is the same explained in Section 3.4.3.1 as all features are observed.

4.9.5 Including dynamic information

In most cases, the last activity you performed influences the current activity you are doing. For instance, if you are *lying*, the most probable activity you will perform after it is *getting up* or still *lying*, but not *falling*. This knowledge can provide valuable input for activity recognition.

Figure 4.31 shows a Hidden Markov Model (HMM) scheme that models this process. Feature values are observed during the current user's activity. Future feature values will be observed depending on the future user's activity. The only influence from a time instant t to a time instant $t+1$ is the current activity the user is performing. Periodic evidence by features disclose the probabilities of every activity through Bayesian network inference, whereas these probabilities are modified also depending on the probabilities of the last activities performed.

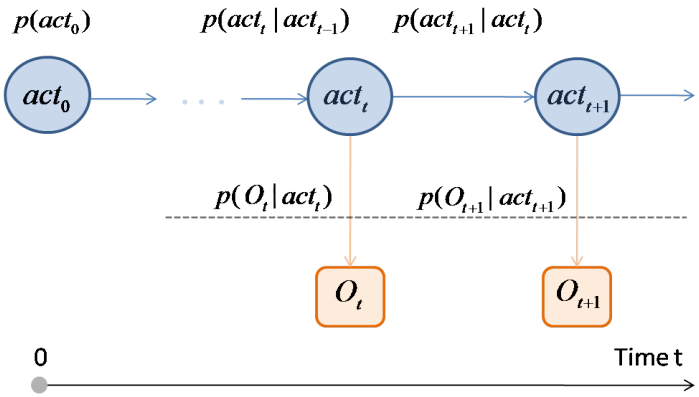


Figure 4.31: Hidden Markov Model scheme. Transition model, $p(act_t|act_{t-1})$, were manually configured by expert knowledge and $p(O_t|act_t^i, \lambda)$, were learned from the recorded data sets and provided by the Bayesian network. It is used by update equation of Grid-based filter to compute $p(act_t|O_{1:t}, \lambda)$.

This first-order HMM for the activity recognition can be characterized by the following:

- N, the number of states of the hidden variable which correspond to the different activities considered in the approach. The individual states at

time t are $act_t = act_t^1, act_t^2, ..., act_t^N$. Activities are considered as hidden because they cannot be observed directly.

- The physical or calculated output which can be observed is called the observation symbols [61]. An observation symbol for a single point in time is given by a vector with values for all features $f_1, f_2, ..., f_M$ computed from the raw sensor data. The vector of features is common for all the hidden states and can be denoted $O_t = (f_{1,t}, f_{2,t}, ..., f_{M,t})$ where $f_{i,t}$, $1 \leq i \leq M$ is the value of the feature i at time t.

- The state transition probability distribution or transition model $A = \{a_{ij} | a_{ij} = p(act_{t+1}^j | act_t^i), 1 \leq i, j \leq N\}$. A will be represented as a matrix and is given in Table 4.9.

- The observation symbol probability distribution in state j, $B = \{b_j | b_j = p(O_t | act_t^j), 1 \leq j \leq N\}$ defined by the measurement model in Bayesian algorithms. This probability is given by the Bayesian network designed before.

- The initial state distribution $\pi = \pi_i$ where $\pi_i = p(act_0^i)$, $1 \leq i \leq N$ is the prior probability. In our evaluations we assumed a startup with the activity *standing*. Therefore the prior assigns a 100% probability to this state and zero to the other activities.

To sum up, the first-order HMM denoted as λ is characterized by:

$$\lambda \sim (A, B, \pi) \tag{4.27}$$

Once the first-order HMM is defined and denoted as λ, the objective is to estimate the most probable hidden state at time t given the past and current observations $O_{1:t} = O_1, O_2, ..., O_t$ as well as the model λ. So, the objective is to estimate $arg_i max(p(act_t^i | O_1, O_2, ..., O_t, \lambda))$.

The problem of estimating the most probable state of a first-order HMM is covered by Bayesian State Estimation algorithms described in Section 3.5. As in our system the hidden state space has a finite number of states (i.e. activities), grid-based methods can be applied providing an optimal estimation of the posterior probability density function $p(act_t | O_{1:t}, \lambda)$. The so called Grid-based filter (see Section 3.5.2 for details), like any recursive Bayesian filter, consists of two successive stages: prediction and update. Prediction and update equations for this approach are:

act_t act_{t+1}	Sitting	Standing	Walking	Running	Jumping	Falling	Lying	Up & Down
Sitting	0.990902	0	0	0	0.000045	0.000045	0	0.009008
Standing	0	0.566024	0.424518	0	0.004717	0.000024	0	0.004717
Walking	0.033217	0.332171	0.498256	0.11626	0.003322	0.000166	0	0.016609
Running	0	0	0.39984	0.439824	0.079968	0.0004	0	0.079968
Jumping	0	0.36842	0.157894	0.263157	0.157894	0.000003	0	0.052631
Falling	0.4	0	0	0	0	0.1	0.5	0
Lying	0	0	0	0	0	0	0.818182	0.181818
Up & Down	0.666445	0.302929	0	0	0	0.000333	0	0.030293

Table 4.9: State transition probability matrix A. Each cell defines the transition probability $p(act_{t+1}|act_t)$.

- Prediction:

$$p(act_t|O_{1:t-1},\lambda) = \sum_{i=1}^{N} w_{t|t-1}^{i}\delta(act_t - act_t^i) \tag{4.28}$$

- Update:

$$p(act_t|O_{1:t},\lambda) = \sum_{i=1}^{N} w_{t|t}^{i}\delta(act_t - act_t^i) \tag{4.29}$$

where

$$\begin{aligned} w_{t|t-1}^{i} &\triangleq \sum_{j=1}^{N} w_{t-1|t-1}^{j} p(act_t^i|act_{t-1}^j) \\ w_{t|t}^{i} &\triangleq \frac{w_{t|t-1}^{i} p(O_t|act_t^i,\lambda)}{\sum_{j=1}^{N} w_{t|t-1}^{j} p(O_t|act_t^j,\lambda)} \end{aligned} \tag{4.30}$$

Once the posterior probability is estimated, the most probable activity is given by the state with the maximum probability.

Chapter 5

Implementation

The software developed for activity recognition is distributed in several packages:

- The main components and the abstract classes of the Java software for activity recognition are included in the package *de.dlr.kn.ARS*. The rest of the classes and Java files are included in different folders:
  - The sensors classes are in package *de.dlr.kn.ARS.sensors*.
  - The signals classes are in package *de.dlr.kn.ARS.signals*.
  - The features computed are in package *de.dlr.kn.ARS.features*.
  - The package *de.dlr.kn.ARS.humanMotionActivitiesHiddenMarkovModel* contains the HMM, the Grid-based filter and the recognition algorithm related classes.
  - The package *de.dlr.kn.ARS.interfaces* includes all the interfaces defined in the system.
- The GUI used in included in package *de.dlr.kn.panels*.
- The off-line training and the classifier evaluation related classes are packaged in *de.dlr.kn.classifiers*.
- The classes required to collect and label the data for the data set are in package *de.dlr.ARS.dataSetReceiver*.
- Other classes and utilities are in package *de.dlr.kn.ARS.utils*.

This chapter provides an overview over the main aspects of the software developed for activity recognition and future work. Appendix G provides the format of the input and output data files required, whose referenced names are given in this chapter.

5.1 Java Software for Activity Recognition

The software was designed taking into account these requirements:

- Efficient, as real-time computation of the features and inference is required.
- Modular and extensible, so new features, signals, sensors and activity recognition systems can be added easily.

5.1.1 Class Descriptions

The main components of the Java framework and the system are:

- Receiver of sensor data who listens to a socket waiting for new messages.
- Reader of sensor data responsible for extracting the measurements and relevant information from the messages and updating the correspondent sensor. Sensors should be registered in advance (see Figure 5.1 for details).
- The administrator of the system, that updates the registered activity recognition systems when the reader of sensor data notifies that new measurements of the sensor have arrived (see Figure 5.1 for more information).
- The activity recognition systems implemented in the framework (e.g.: human motion related activities recognition system).

The human motion related activities recognition system has five main components:

- Sensors used. Sensors can be a unique sensor or a platform of sensors such as an IMU (see Figure 5.2 for an example).

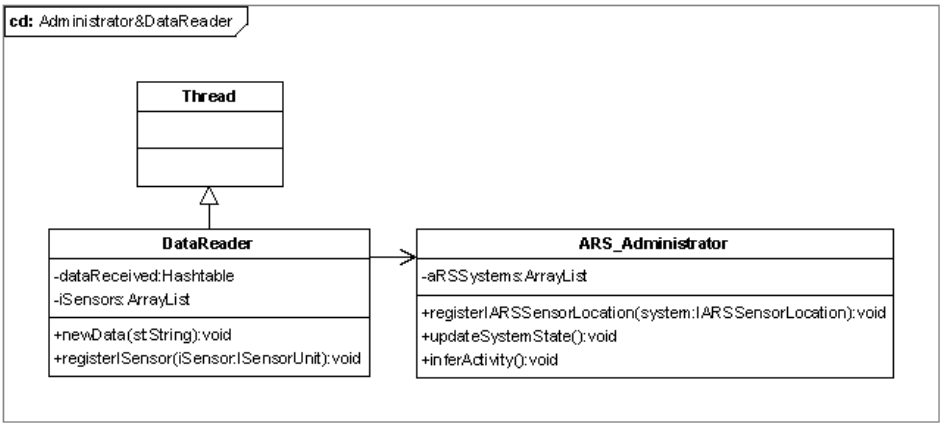


Figure 5.1: UML class diagram for data acquisition and notification.

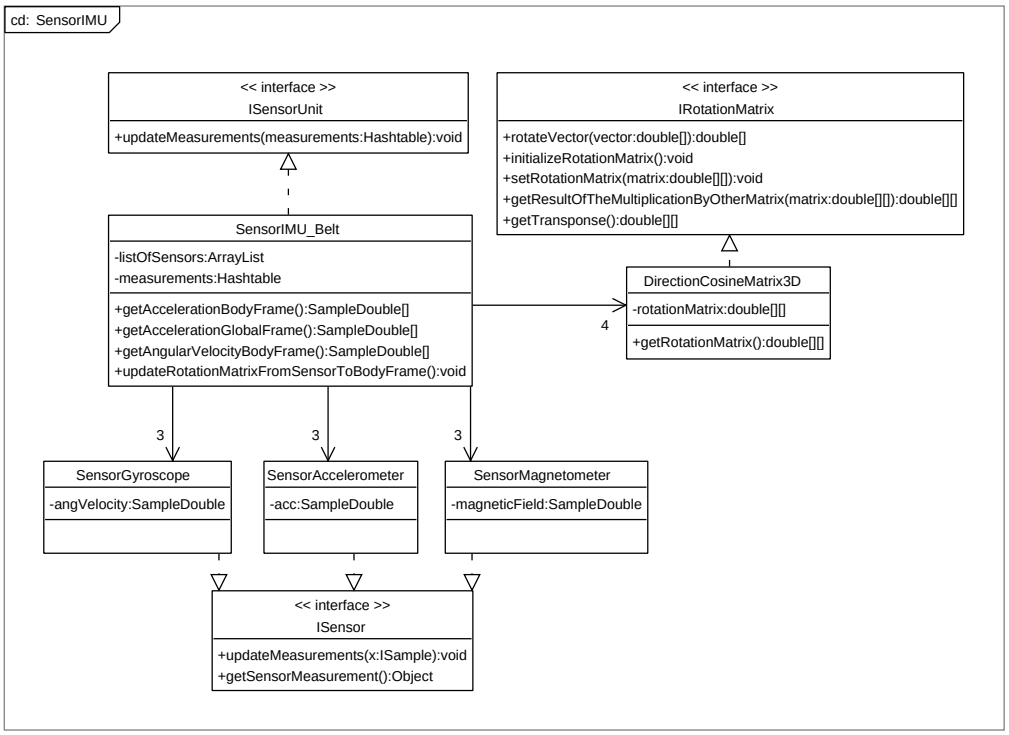


Figure 5.2: UML class diagram of sensors involved in the system.

- Signals that consist of a collection of values or samples. The class diagram is shown in 5.3. Four abstract classes are defined:
  - Measured signals from sensors.
  - IMU signals obtained by the rotation of measured signals or the norm of them.
  - Processed signals after processing of a measured signal or an IMU signal. E.g.: signal of DFT coefficients of $|a|$.
  - Filtered signals extracted through the filtering of a raw signal such as $|a|$.

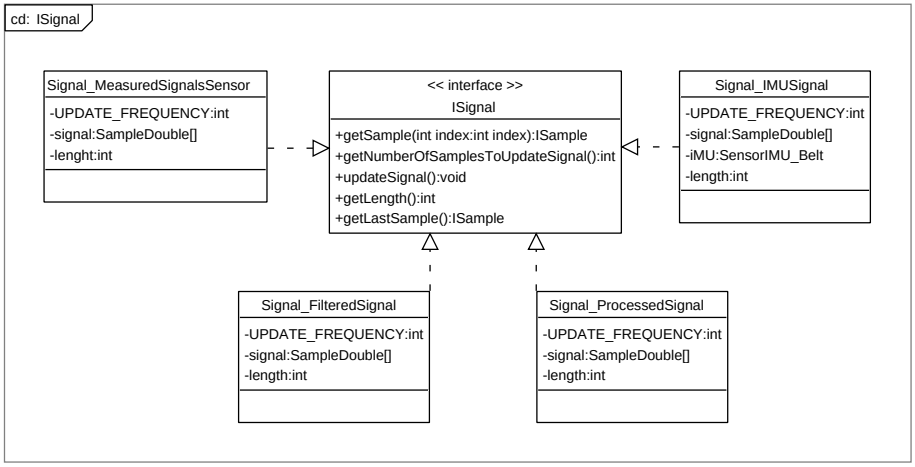


Figure 5.3: UML class diagram of signals involved in the system. Interface *ISignal* is implemented by four abstract classes.

- Features computed (see Figure 5.4 for details). Three abstract classes are defined:
  - Simple features extracted from a signal such as its mean value.
  - Compound features that are computed from a signal and the value of a simple feature like the standard deviation of a signal.
  - Compound features of second order that require previous values of at least one compound feature. It could require as well a signal or the value of a simple feature. An example is the correlation

coefficient between two signals where mean value and standard deviation are required.

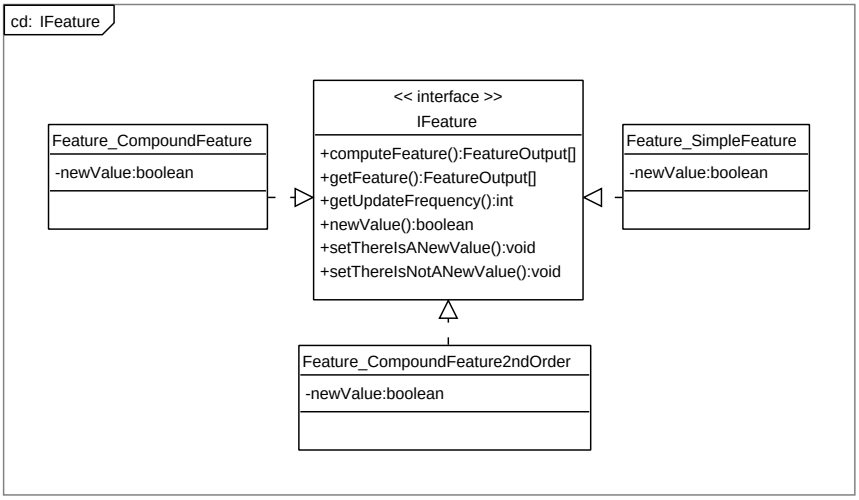


Figure 5.4: UML class diagram of features involved in the system. Interface *IFeature* is implemented by three abstract classes.

- Final set of features. Not all the features computed are included in this final set as some of them are previous values required for the computation of other features. This final set of features is discretized and is used as evidence for the recognition algorithm (see Figure 5.5 for details).
- The recognition algorithms (see Figure 5.6 to observe the class diagram):
  - Dynamic classifiers, based on a HMM that uses BNs and Grid-based filter for the inference. These approaches are defined using Grid-based filter to include dynamic information to the system.
  - Static approaches that use the information extracted from the BNs. They use the observation symbol probabilities of the HMMs combined with the priors probabilities of *activity* given in Chapter 4, so the information extracted from the BNs do not have to be computed twice in future work, for efficiency.

See Figure 5.7 for an example of several features and signals involved.

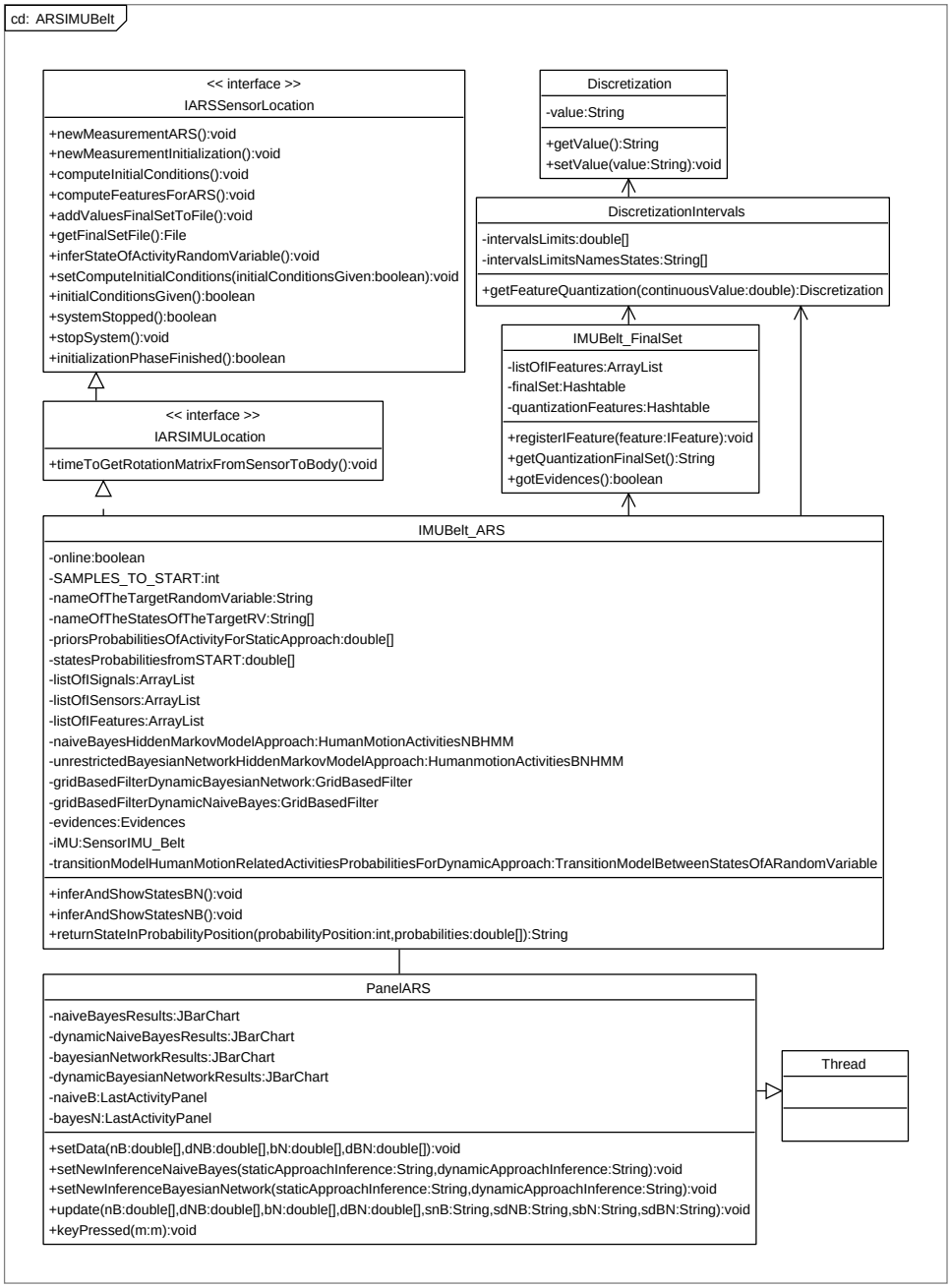


Figure 5.5: UML class diagram of the human motion related activities recognition system. The sensor used is an IMU and is located on the belt.

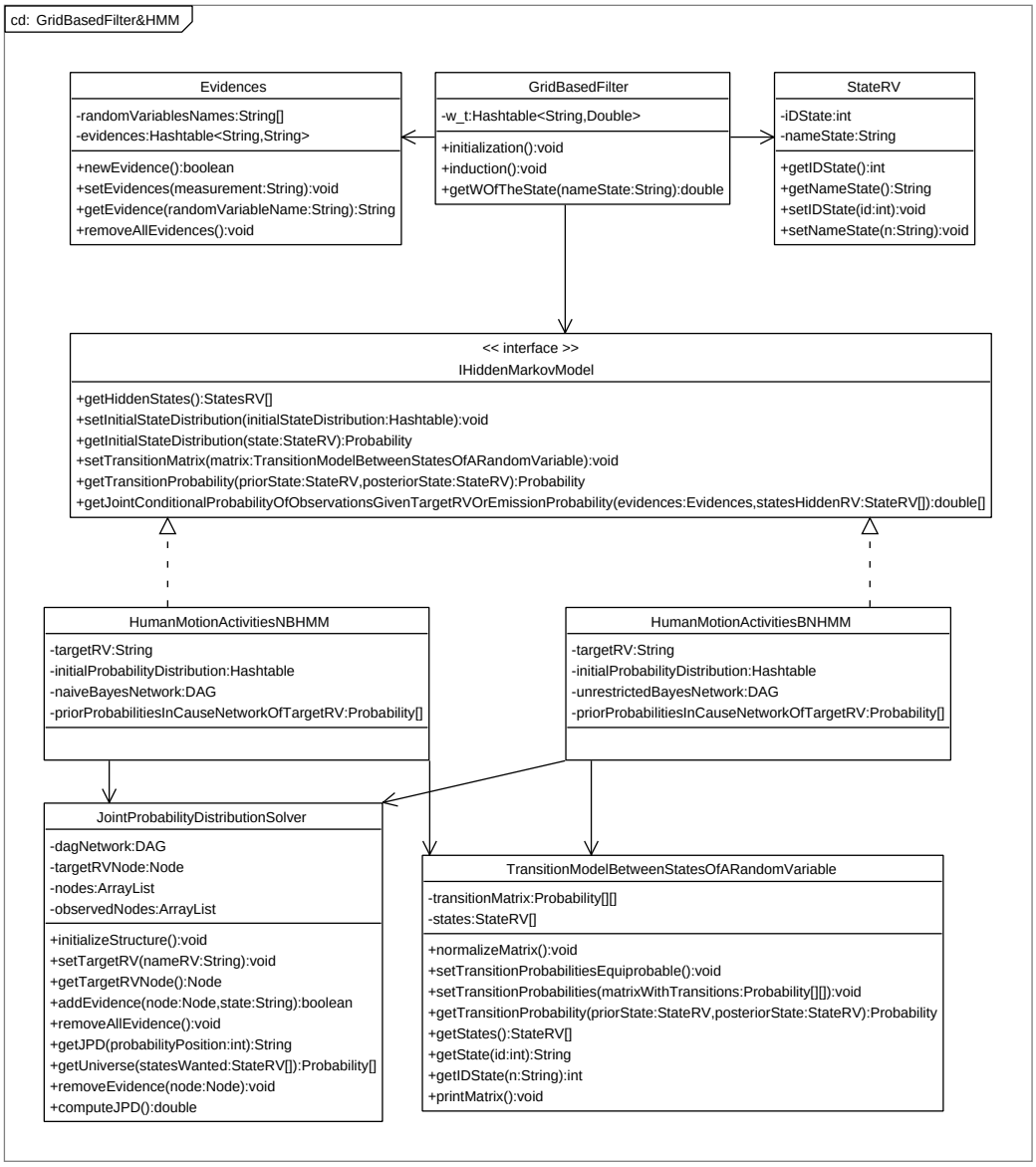


Figure 5.6: UML class diagram of inference algorithm. Unrestricted bayesian network and Naïve Bayes hidden Markov model are shown and used for the Grid-based filter.

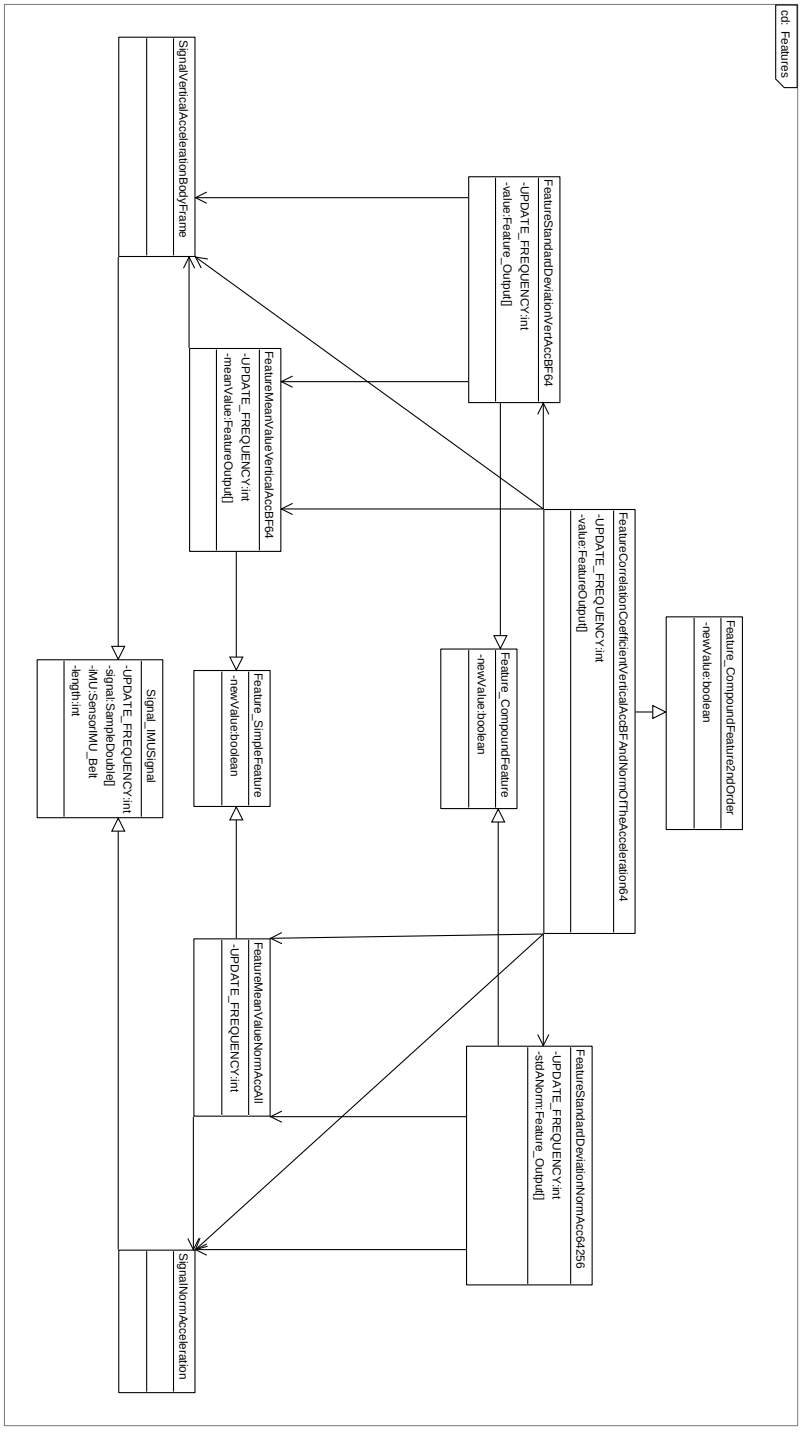


Figure 5.7: Example of UML class diagram of several features and signals involved in the system.

5.1.2 Data Reader and Administrator Operation

Figure 5.8 explains how data is acquired. As soon as a new measurement arrives, the administrator is notified and gives the order of updating all the registered activity recognition systems. The flowchart describing this process is given in Figure 5.9.

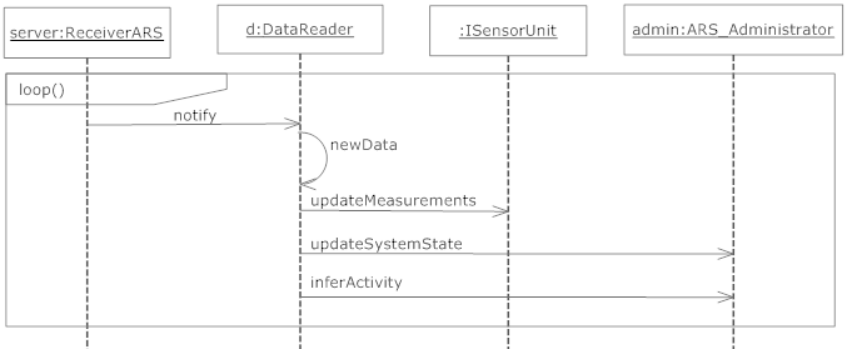


Figure 5.8: UML sequence diagram explaining data acquisition and notification to the system through the administrator and the data reader.

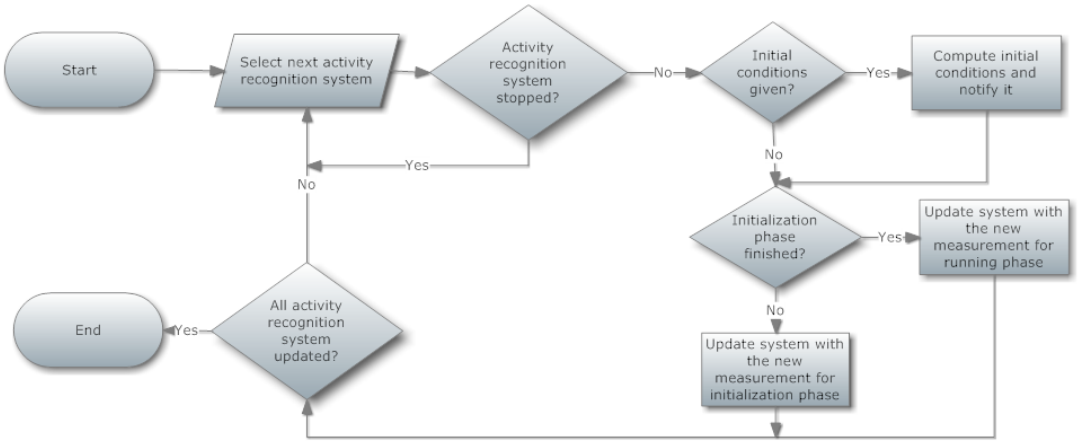


Figure 5.9: Flowchart of system update process. Initialization phase is finished when almost 512 measurements (length of the longest window) have arrived. Inference will not start until the user has been standing for almost 5.12 seconds.

When all the systems are updated, the administrator sends the order of inferring the activity (see Figure 5.10).

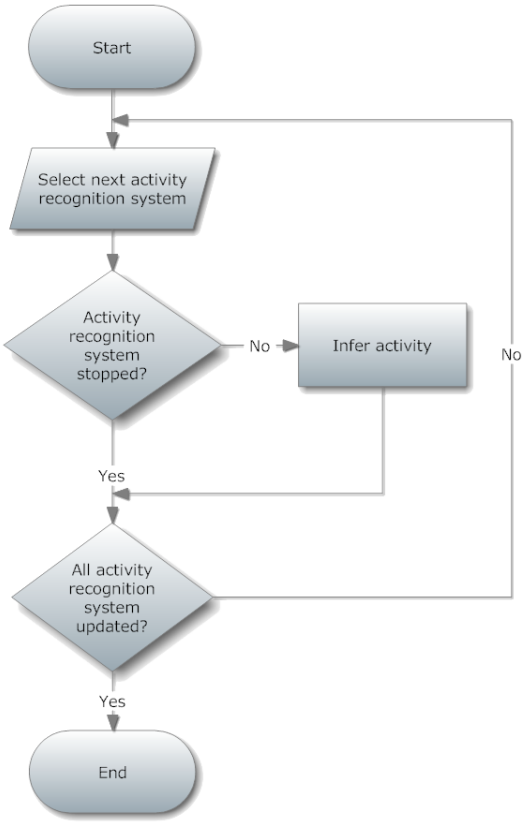


Figure 5.10: Flowchart of system inference process.

5.1.3 Human Motion Related Activity Recognition System Operation

The following sections will describe several aspects of our human motion related activity recognition system. The aspects to explain in this section are:

- States of the system and working modes that a researcher can use to evaluate and to use the system.
- The updating and inference process that occurs when the administrator sends the order to update and to infer the activity.

- The user interaction of the system as a guide of *how to use it*.

- Instructions about possible future modifications of the system and several configuration parameters.

All these comments can be seen as well in the source code with more details.

5.1.3.1 States and Working Modes

The activity recognition system has three different states and two working modes (see Figure 5.11 for more information).

The three states are:

- *Bootstrapping*: at least 512 samples are acquired and the user has been standing for at least 5.12 seconds.

- *Inferring*: the activity inference starts.

- *Stopped*: the inference is stopped.

The working modes are *on-line* or *off-line* activity recognition:

- *On-line*: the sensor is connected and the user should inform the system when inference must start. A description of the user interaction with the system will be given in Section 5.1.3.3.

- *Off-line*: the user data can be stored in a file[1] and used later for the evaluation of the system[2]. A data server will extract the messages from the file where sensor measurements are contained. The message where the user was standing for at least 5.12 seconds should be provided.

[1]These files are referenced as *measurementFiles* or *benchmarkFiles* in Appendix G.
[2]These files are referenced as *resultsInferenceFiles* in Appendix G.

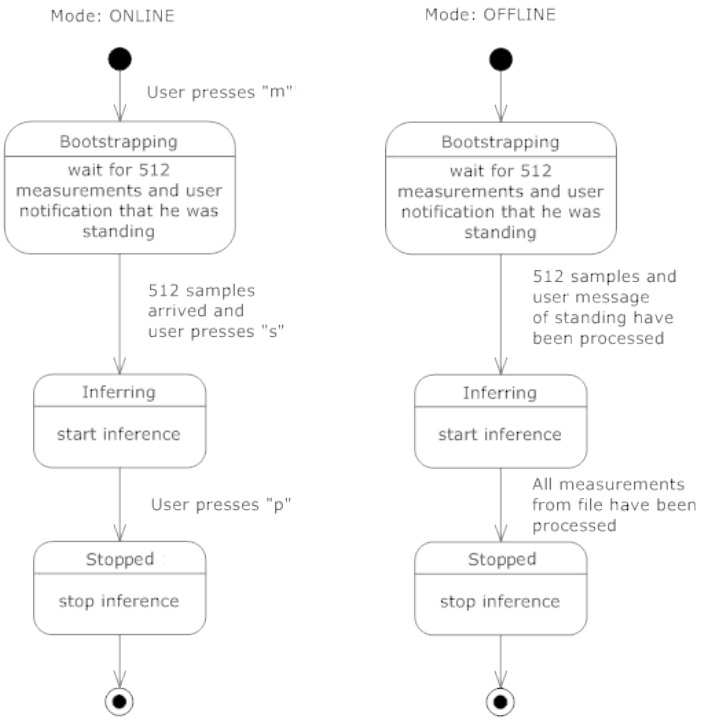


Figure 5.11: UML state diagram of human motion related activities recognition system for both working modes: *on-line* and *off-line*.

5.1.3.2 Updating and Inferring

Once the administrator sends the order to update, the activity recognition system updates its signals and features through its sensor information as follows:

- First, signals of the system are updated. Each signal has its own update frequency. The order in which they are updated is:
  1. Measured signals from sensors.
  2. IMU signals.
  3. Processed signals.
  4. Filtered signals.
- If the inference has started, each feature should check its update fre-

quency to know if it has to update. Features are updated in the following order:

1. Simple features.
2. Compound features.
3. Compound features second order.

The updating process (once the initialization phase has finished) is described in Figure 5.12. Figure 5.13 describes how every feature and signal is updated.

The process of inferring the current user's activity is described in Figure 5.14.

5.1.3.3 User Interaction with the System

The wearer of the sensor has to provide the following information to the system:

- The point of time where the sensor is already mounted to the belt.
- The initial time when the user has been *standing* for at least five seconds. If the user is *standing* for at least five seconds, initial evidence computed belong to *standing* activity being coherent with the assumptions given in 4.1 and the initial probability distribution of the Grid-based filter.

A Unified Modeling Language (UML) sequence diagram describing the user's interaction with the system is given in Figure 5.15.

5.1.3.4 Adding a New Signal

A new signal to the human motion related activities recognition system is added as follows:

- Declare and create it in class *IMUBelt_ARS*.
- Register it adding it to the list of *ISignal*.

If the new signal requires the information of another signal or sensor, they should be already registered in the system, so the signal can take it from the list of registered *ISignal* and *ISensorUnit*.

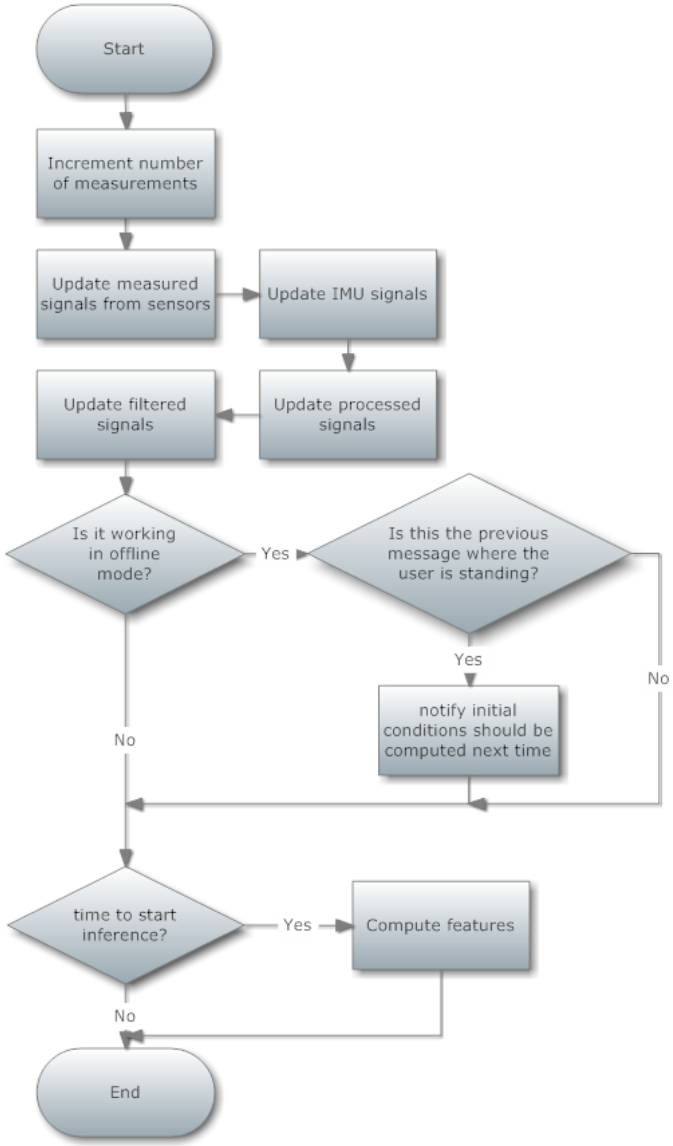


Figure 5.12: Flowchart describing the updating algorithm of this activity recognition system once initialization phase has finished (512 samples are acquired).

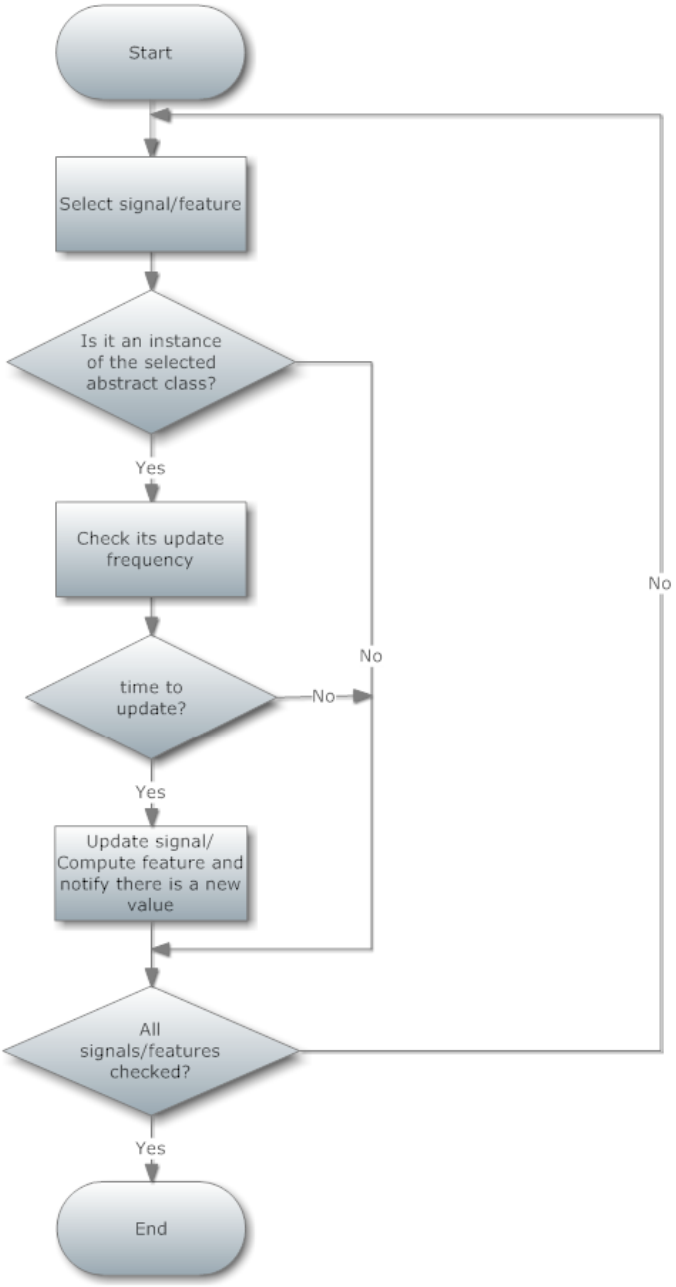


Figure 5.13: Flowchart describing the signal or feature updating process. Every signal and sensor should check through their update frequency when they should update and get a new value.

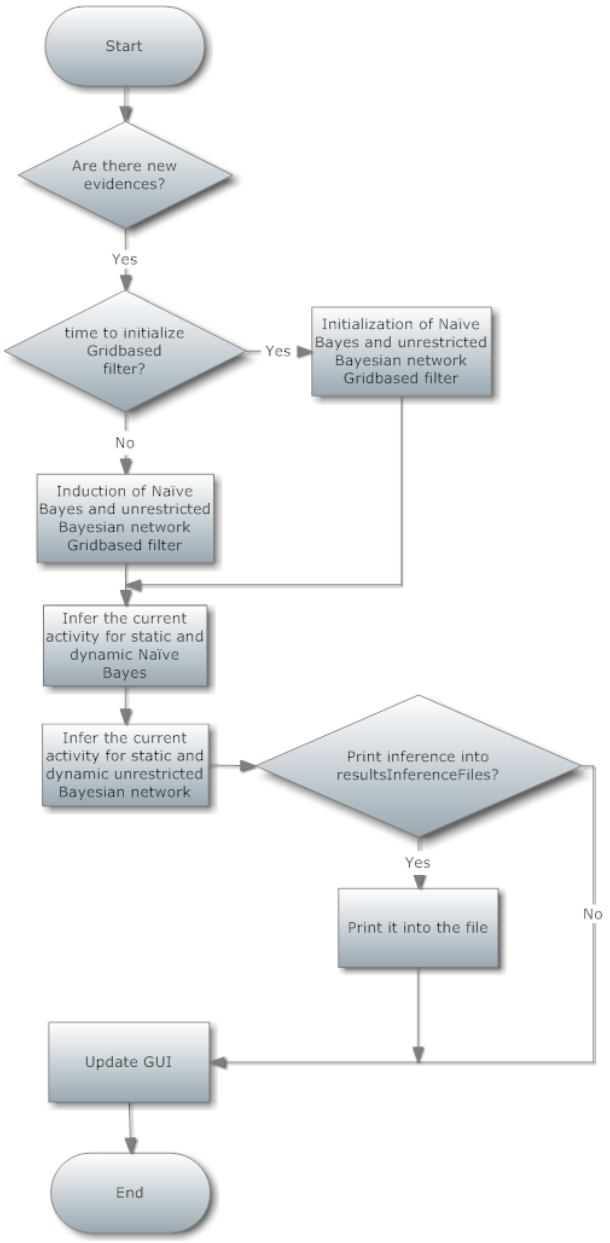


Figure 5.14: Flowchart describing the inference algorithm of this activity recognition system.

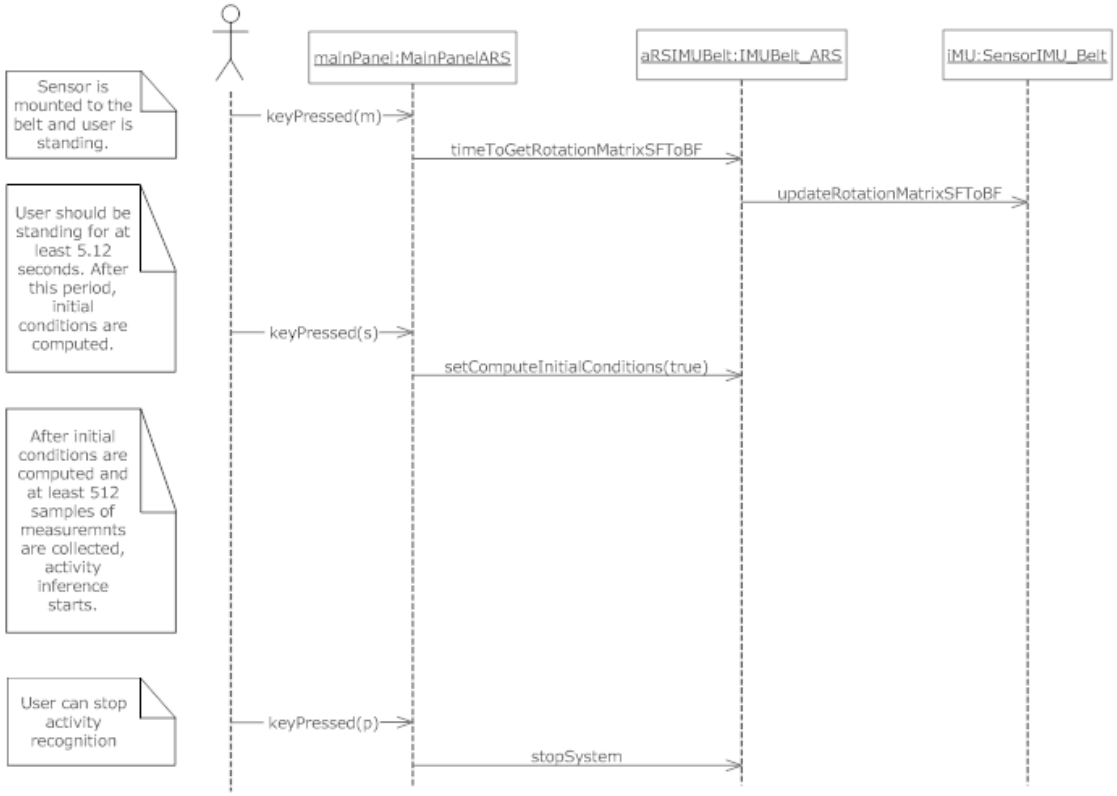


Figure 5.15: UML sequence diagram describing the user interaction with the system.

5.1.3.5 Adding a New Feature

A new feature is added as follows:

- Declare and create it in class *IMUBelt_ARS*.
- Register it adding it to the list of *IFeature*.
- If its feature value will be used as evidence, so it belongs to the final set of features used for inference, it should be registered and its *FeatureOutput* to the final set of *IMUBelt_ARS* giving its discretization intervals in *IMUBelt_ARS*. The final set size has to be incremented.

If the new feature requires the information of another signal or feature, they should be already registered in the system, so the feature can take it from the list of registered *ISignal* and *IFeature*.

5.1.3.6 Changing the Bayesian Networks

Bayesian networks used for inference can be changed. To use another Bayesian network, the following should be done:

- The class *HumanMotionActivitiesBNHMM* declares a variable *filenameNetworkSBN* where the trained unrestricted Bayesian network object file is contained. This network is obtained from training with the classes explained in Section 5.2.
- The class *HumanMotionActivitiesNBHMM* declares a variable *filenameNetworkNB* where the trained Naïve Bayes network object file is contained. This network is obtained from training with the classes explained in Section 5.2.

If these networks use a different quantization from the ones set before, these quantization intervals should be provided in the class *IMUBelt_ARS*.

5.1.3.7 ARS Settings and Possible Options

This section will describe the functionality of several configuration variables that are already commented in the code. They are defined in class *IMUBelt_ARS*.

- Boolean variable *online* that allows to use the system in *on-line* mode if *true* or *off-line* mode if *false*.
- Boolean variable *putFeatureInFile* that will store the value of the final set of features in the file *fileWithTheNumericalValuesOfTheFinalSetOfFeatures*[3].
- Boolean variable *putInferenceInAFile* that will store the computed probabilities for each activity for every recognition algorithm. The results are written in files *filenameNB,filenameDNB,filenameBN,filenameDBN*[4].
- `int` constant *SAMPLES_TO_START* that defines when the initialization phase has finished.

[3]This file is referenced as *featuresFinalSetValuesFiles* in Appendix G.
[4]These files are referenced as *resultsInferenceFiles* in Appendix G.

- `int` static variable *standingMessage* used for *off-line* mode. It defines when the user has been standing at least 5.12 seconds.
- String static variable *nameOfTheTargetRandomVariable* set to *activity_AOOG* that indicates the target RV that should be inferred.
- Array of strings, *nameOfTheStatesOfTheTargetRV* giving the states of the target RV.
- Priors probabilities for every activity, initial state probability distribution, transition probabilities and quantization intervals should be set as well (see Figure 5.5).

If updating the rotation matrix from sensor frame (SF) to body frame (BF) is required due to the free movement of the sensor, method *timeToGetRotationMatrixFromSensorToBody* should be executed (see Figure 5.5 for a reference to this method).

5.2 Off-line Classifier Evaluation

The evaluation of BNs as classifiers can be done off-line. The UML class diagram of the code allowing this evaluation is given in Figure 5.16. Using these classes, the user is allowed to:

- set the training[5] and test[6] set for the classifier.
- set training parameters of the Greedy Hill Climber algorithm such as learning minutes, maximum number of parents, Naïve Bayes approach or seed network[7]. The learning engine was modified and the selection of a seed network and and an initial configuration of learning algorithm is shown in the flowchart given by Figure 5.17.
- perform the required classification test. The confusion matrix and evaluation parameters like precision and recall are computed and written into a file.

[5]These files are referenced as *groundtruthFiles* in Appendix G.
[6]These files are referenced as *evidenceFiles* in Appendix G.
[7]Seed network structure file is referenced as *networkDAG* in Appendix G.

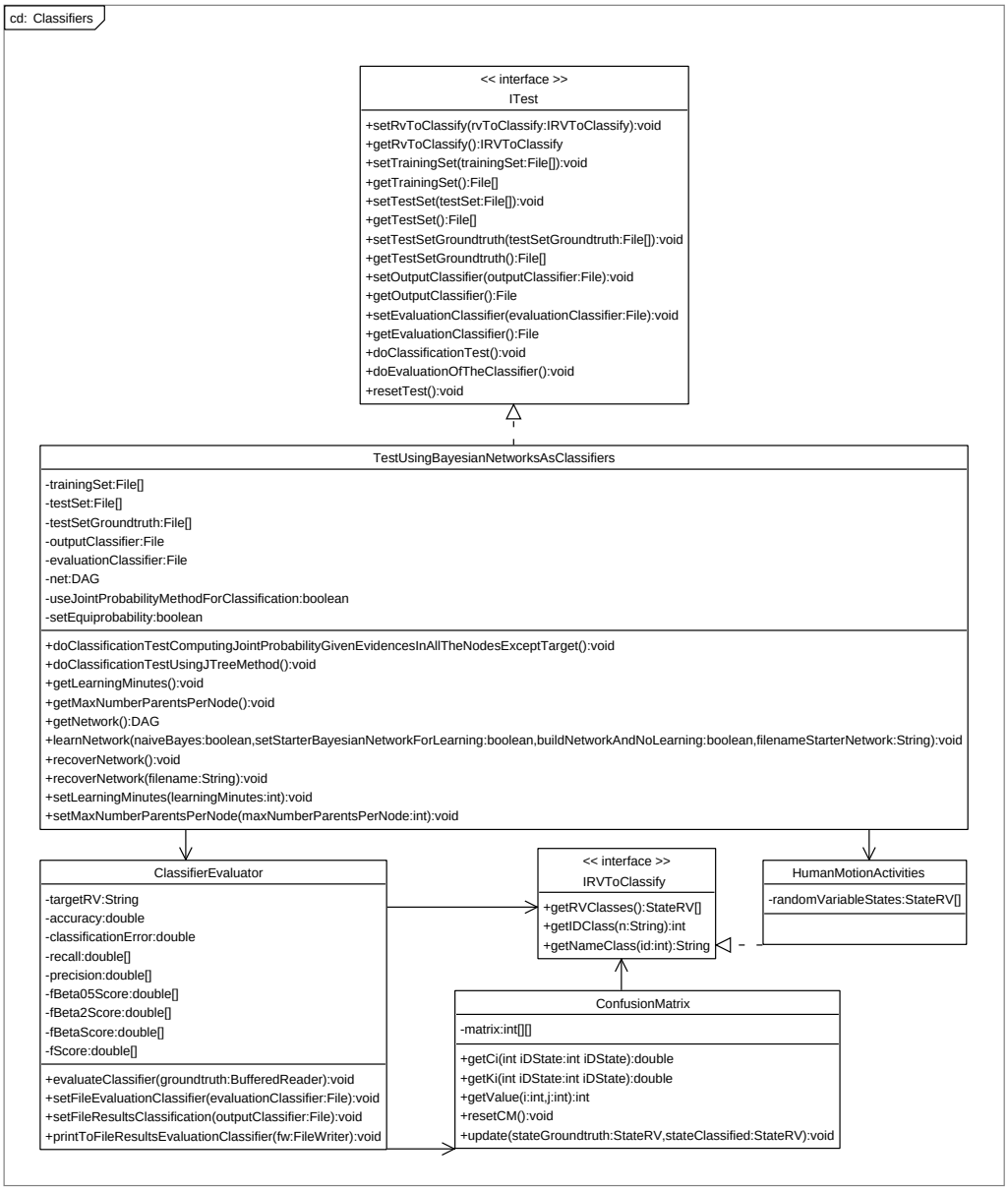


Figure 5.16: UML class diagram of package *classifiers*. Bayesian networks are trained using the training set and used to infer the random variable *human motion related activities* of the instances given by the test set. An evaluator of the classifier will build the resulting confusion matrix and compute parameters such as *precision* and *recall* for every activity.

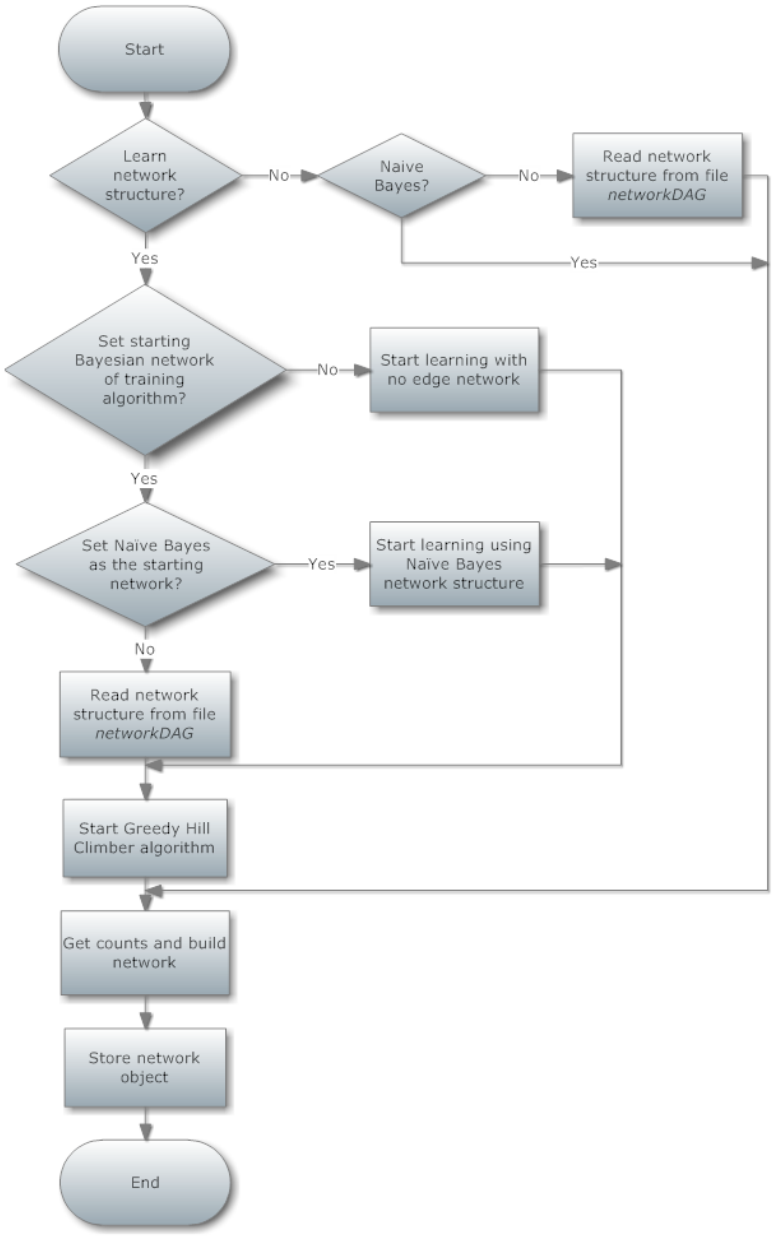


Figure 5.17: Flowchart explaining the possible configuration of the training algorithm. Structure can be set to Naïve Bayes, empty network or network written into a file.

Chapter 6

Evaluation

The four approaches for classifying activities to compare are:

- Naïve Bayes approach that does not include an activity transition model, called *static Naïve Bayes* or *Naïve Bayes* (NB).
- *Dynamic Naïve Bayes* using a Grid-based filter (DNB).
- *Static* unrestricted *Bayesian network* (BN).
- Unrestricted Bayesian network including dynamic information, *dynamic* unrestricted *Bayesian network* (DBN).

Each of them were designed for four different discretizations (see Section 4.9.2 for details).

Part of this work was the reduction of the number of features and the design the discretization of them. Consequently, one aspect to evaluate is the sensitivity of the classifier to both processes. Besides, update frequency of the features were set to 4 Hz. This chapter will evaluate if the update frequency of the features influences the inference quality.

With respect to the recognition algorithms based on the unrestricted Bayesian network, the structure of the network was learnt with evidence samples computed at 4 Hz from the data set that was collected in this work (see Section 4.4 for more details). Also, the feature discretization process was performed studying the histograms of the features when evidence was computed at 4 Hz. However, the conditional probability tables are learnt using the training set defined for every test. The reason for not using the training

data to learn the network structure is that it is assumed that the structure of the unrestricted Bayesian network will be learnt using the training data and this structure does not change significantly depending on how often evidence is computed. Besides, providing evidence at 4 Hz seems to be a sensible rate for obtaining meaningful and sufficient data (even if the noise for the nature of the processing of the signal increases as well). The main reason is that, on the one hand, evidence cannot be provided at a extremely high rate because the memory required for the training algorithm increases as well as the computational burden. On the other hand, the minimum duration of one activity is around one second, and obtaining four evidence per second assures to get the feature values of the final set when the most significant phases of the activity are performed.

The evaluation of the approaches will be done in terms of precision and recall for every activity (see Section 3.3.3 for more information). Classifier accuracy and classification error are not reliable, because the number of instances for every activity are not the same. Finally, real-time activity inference will be evaluated as well.

The process of the evaluation of the system will consist of the following:

- Naïve Bayes and unrestricted Bayesian network evaluation will be compared and evaluated without including the dynamic information. The objective is to analyze the effects of the feature discretization algorithm and the feature reduction process as well as the update frequency of the features. To perform this evaluation, the network structure and three tests will be studied (see Section 3.3.3 for details):

  - Resubstitution.
  - Four-fold cross-validation.
  - Evaluation of the inference by the data of two subjects, that have not been part of the training set.

  Resubstitution and four-fold cross-validation will be performed for different update frequency of the features. These tests will assume that the prior probabilities of the states of the node *activity* are equal, so the activities are equiprobable.

- Once the most adequate feature discretization is chosen as well as the update frequency of the features, static and dynamic approaches will

be compared using the data of two subjects, that have not been part of the training set.

- Finally, the system is evaluated in real-time.

Appendix I shows the confusion matrices obtained for the most relevant tests performed and explained in this chapter.

6.1 Feature Reduction Process

On the one hand, the Naïve Bayes structure is given by definition. If bad, redundant or unnecessary features are included, its inference quality could be affected by them improving or getting worse. On the other hand, the structure of the unrestricted Bayesian networks is learnt using the Greedy Hill Climber algorithm (see Section 3.4.2 for more information).

In this work, interesting results were obtained with respect to the structure of the trained Bayesian networks that can be seen in Figures 6.1,6.2,6.3 and 6.4:

- All the features of the final set are part of the Markov Blanket of the node *activity* for the discretizations 1Da and 1Db.

- For the discretization 2Da, features number 7 and 17 (see Table 4.6 for details) were identified as redundant or unnecessary and were not included in the Markov Blanket of *activity*.

- For the discretization 2Db, features number 7, 10 and 17 (see Table 4.6 for more information) were not included in the Markov Blanket of *activity*. Appendix

Therefore, the following can be observed:

- The discretization of the features is decisive to identify a feature as useful or not by the training algorithm. For example, if the number of states of the features is high, the training algorithm will receive enough information to characterize and infer the target node (through the process of fitting the data set), and, consequently it can prune the less useful features which are not in the Markov Blanket.

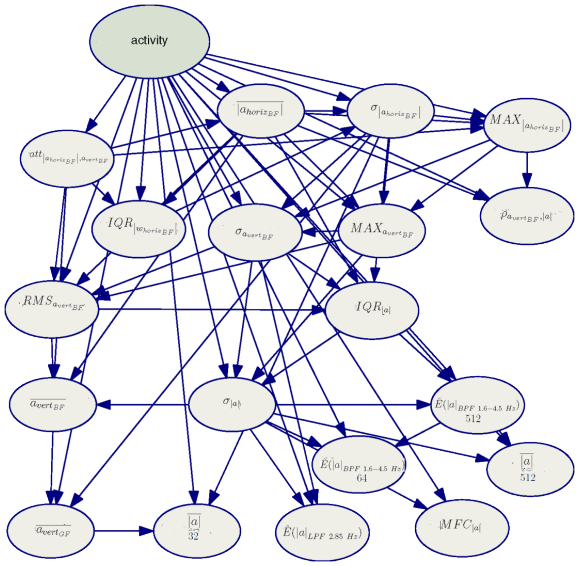


Figure 6.1: Structure of the trained Bayesian network for the discretization 1Da. All the features are inside of the Markov Blanket of the node *activity*.

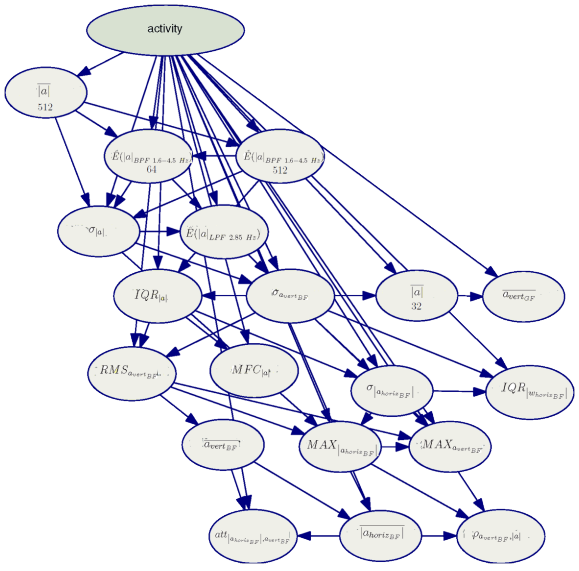


Figure 6.2: Structure of the trained Bayesian network for the discretization 1Db. All the features are inside of the Markov Blanket of the node *activity*.

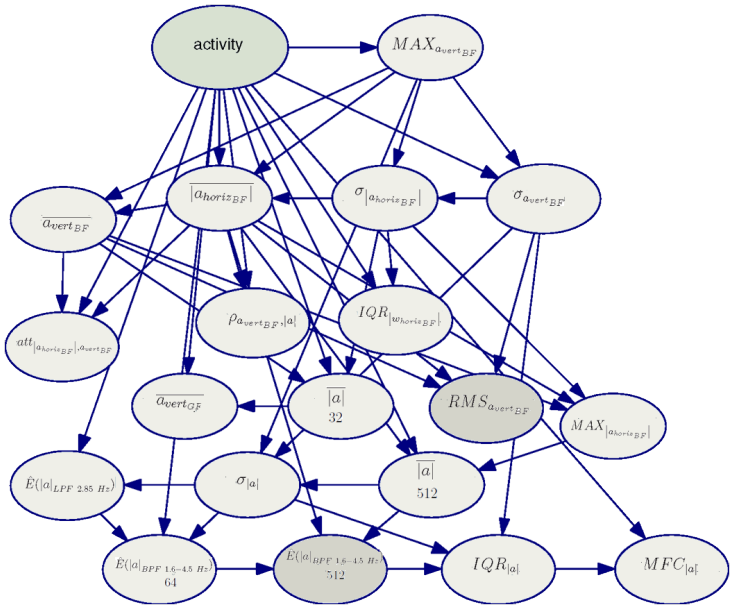


Figure 6.3: Structure of the trained Bayesian network for the discretization 2Da. The features that are outside of the Markov Blanket of the node *activity* are pointed out with a darker color.

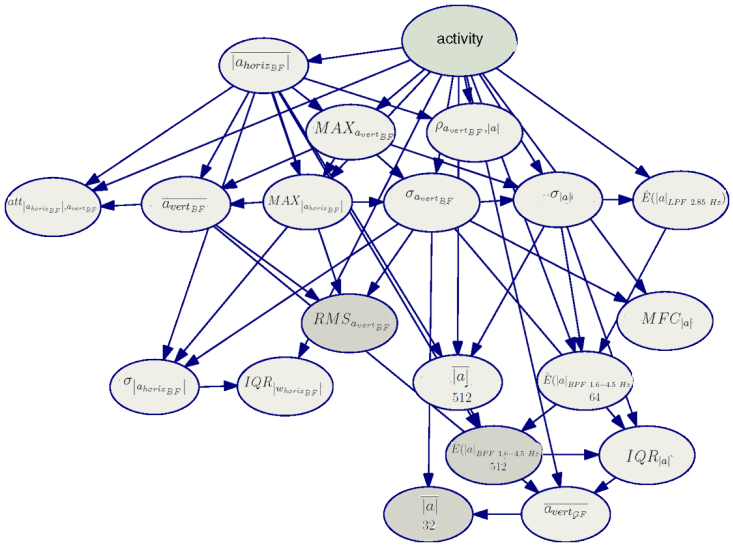


Figure 6.4: Structure of the trained Bayesian network for the discretization 2Db. The features that are outside of the Markov Blanket of the node *activity* are pointed out with a darker color.

- Unrestricted Bayesian networks will be more robust to the process of selection of the features because increasing the complexity of the network structure by adding a new arrow is penalized by the training algorithm. Consequently, it is added if the fitness computed supposes a significant improvement in the fitness of the network to the data set during the training process. If the information provided by the feature is redundant or unnecessary, the final structure of the Bayesian network will not include it in the Markov Blanket of *activity* as it happened with the features 7 and 17 for the discretizations in 2D. The same conclusion and analysis can be done for bad features or features that are not related to the node *activity*. In this case, the training algorithm will probably not find any improvement by considering the relation of this node to the Markov Blanket of *activity*.

6.2 Feature Discretization Algorithm and Update Frequency of the Features

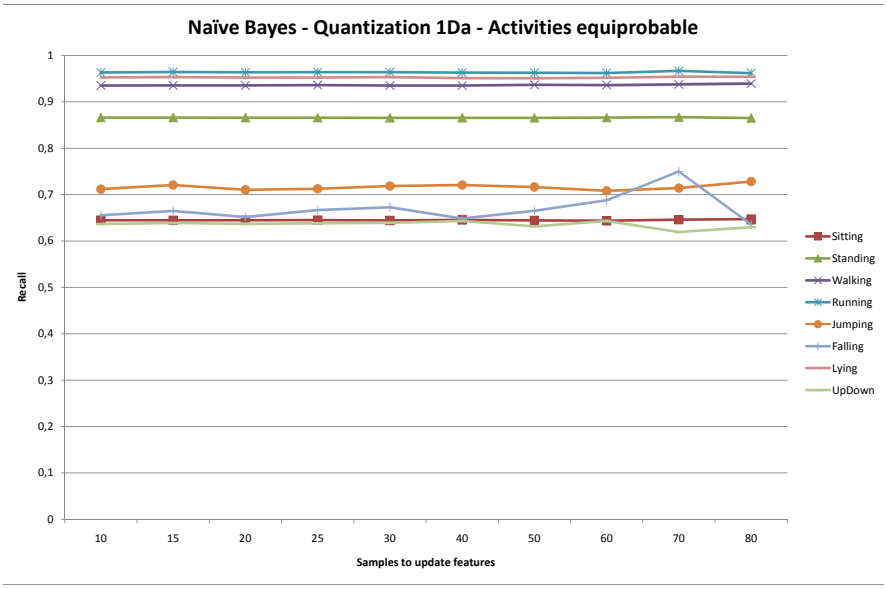
Apart from the four discretizations described in Section 4.9.2, new evidence could be computed at different rates. This section will consider the following update frequencies of the feature vector that are shown in Table 6.1 for some of the tests presented.

Number of samples	10	15	20	25	30	40	50	60	70	80
Time (s)	0.10	0.15	0.20	0.25	0.30	0.40	0.50	0.60	0.70	0.80
Frequency (Hz)	10	6.7	5	4	3.3	2.5	2	1.67	1.42	1.25

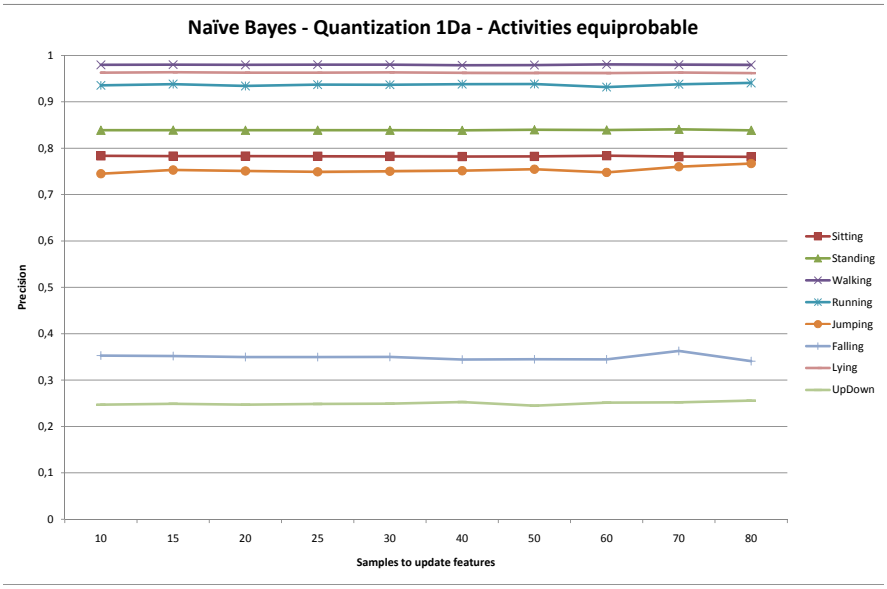
Table 6.1: Number of samples to compute the features. Time in seconds and frequency update of the feature vector are shown. Each time features are computed, a new evidence is given to the system.

Resubstitution test selects the training set as the test set (see Section 3.3.3 for details) and was done for the update frequencies of the features given in Table 6.1. Figures 6.5, 6.6,6.7 and 6.8 shows the precision and recall for every activity and transition for two different discretizations.

The next test performed was the four-fold cross-validation test for the update frequencies given in Table 6.1. Figures 6.9, 6.10,6.11, 6.12,6.13 and

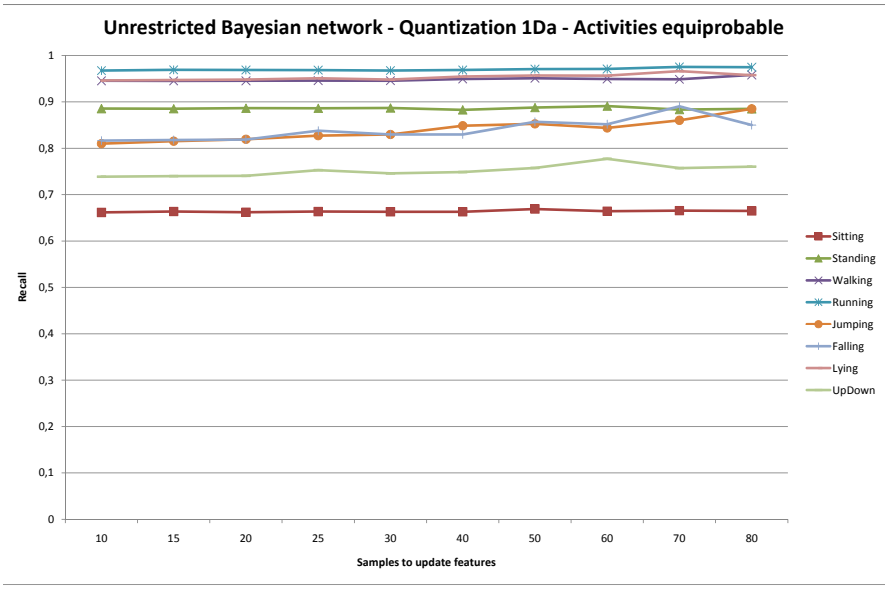


(a)

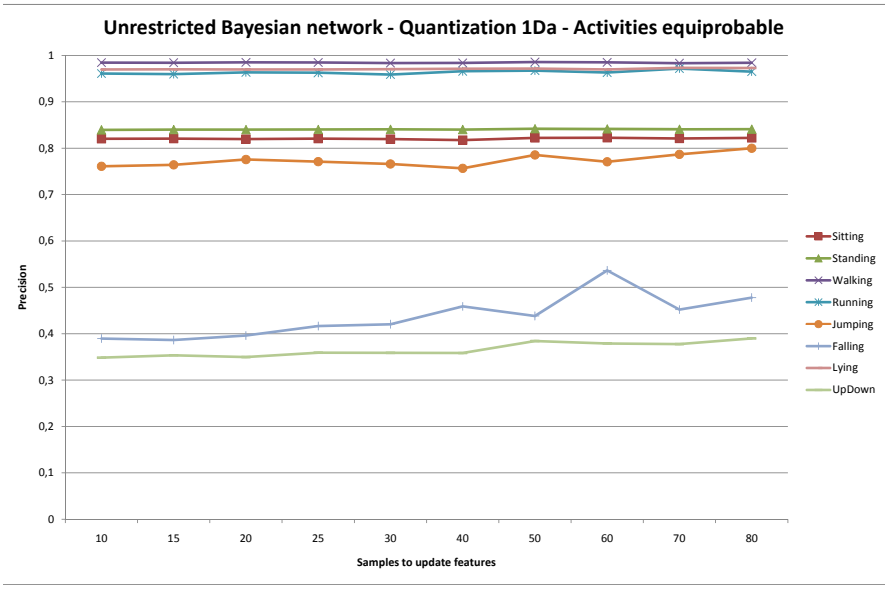


(b)

Figure 6.5: Recall and precision of the resubstitution test for every activity considering the Naïve Bayes approach for the discretization 1Da.

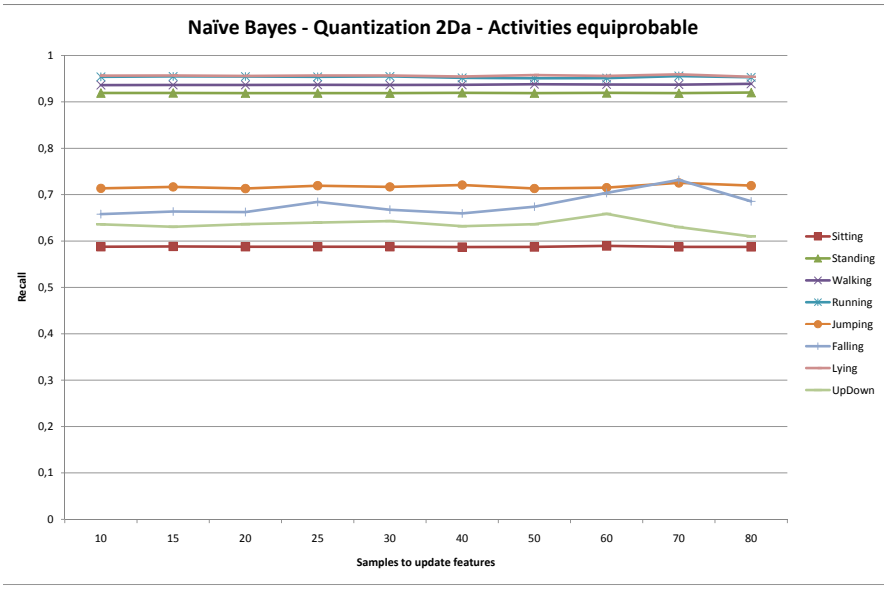


(a)

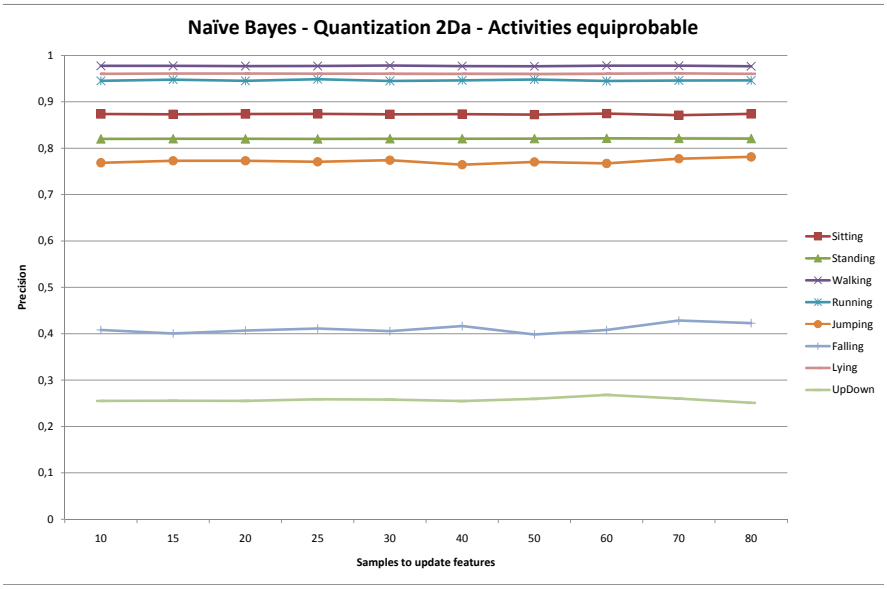


(b)

Figure 6.6: Recall and precision of the resubstitution test for every activity considering the unrestricted Bayesian network approach for the discretization 1Da.

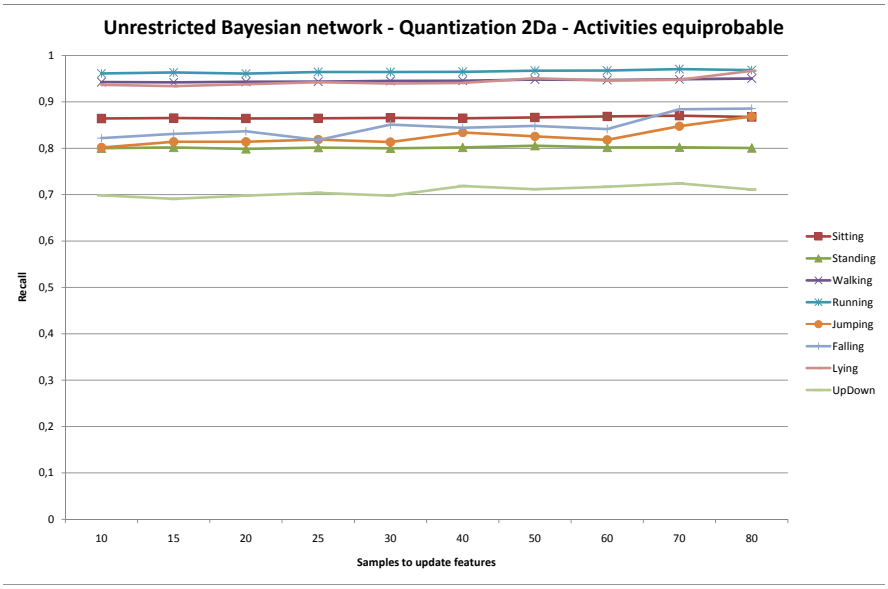


(a)

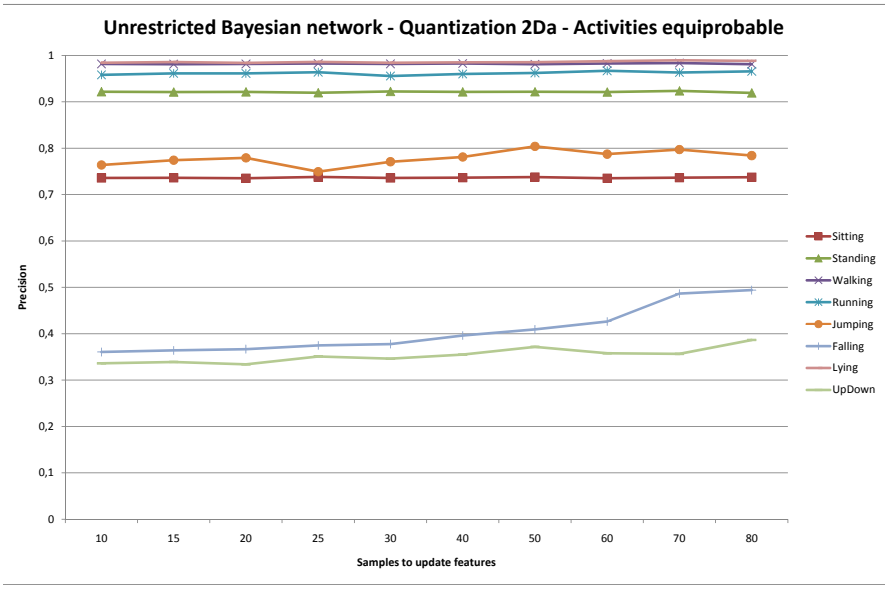


(b)

Figure 6.7: Recall and precision of the resubstitution test for every activity considering the Naïve Bayes approach for the discretization 2Da.



(a)



(b)

Figure 6.8: Recall and precision of the resubstitution test for every activity considering the unrestricted Bayesian network approach for the discretization 2Da.

6.14 show the precision and recall for every activity and transition for two different discretizations.

Finally, the frequency update of the features was set to 4 Hz for the following test. The test set is obtained from two subjects that are unknown to the system, i.e. not included in the training set. The training set of the Bayesian network corresponds to the whole data set, except those two subjects. Figures 6.15, 6.16,6.17 and 6.18 show the results of the inference in time.

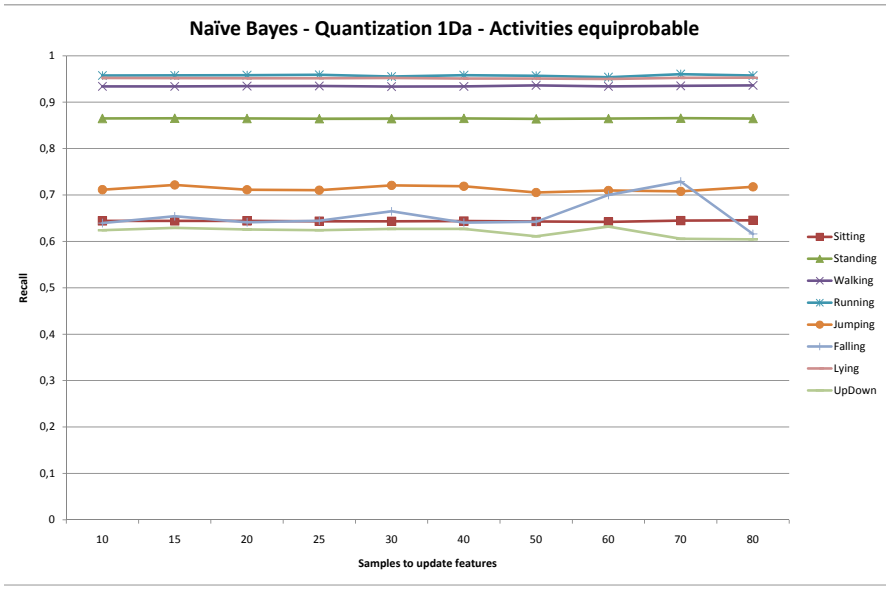
The conclusions extracted from these tests are given below:

- Unrestricted Bayesian networks improve the results of the resubstitution and four-fold cross-validation tests in comparison with Naïve Bayes because the training algorithm tries to fit evidence samples extracted from the data set. Also, there are activities such as *jumping* and *falling* or transitions such as *getting up* and *getting down* that are difficult to characterize and better recognized using the unrestricted Bayesian network approach.

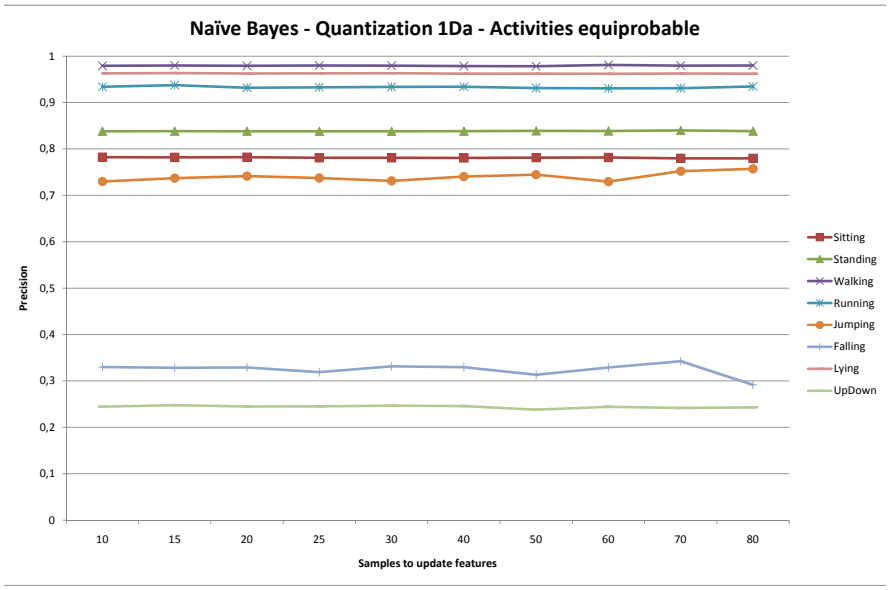
- The feature discretization in 2D has, in general, more states per feature, improving the results of the resubstitution test because the algorithm has more freedom and information to get the Bayesian network that better fits the data set. Four-fold cross-validation results for the 2D discretizations are also better. However, increasing the number of states of the features is unnecessary as can be observed through the comparison of the inference results of the test done with the data from two subjects that are unknown to the system. The discretizations 1Da and 1Db show similar or even better results than the discretizations 2Da and 2Db. Even more, the discretizations in 2D fit the data set that includes the evidence and the noise obtained by the processing of the sliding window, and the undesired overfitting of the data could be achieved.

- Even if the feature discretization has been designed for the extracted data at 4 Hz, the learnt network structure is equally effective and valid for the rest of the update frequencies as can be observed in the results of the resubstitution test.

- Changing the update frequency of the features does not change the inference results, except for *falling* and the transitions *getting up* or

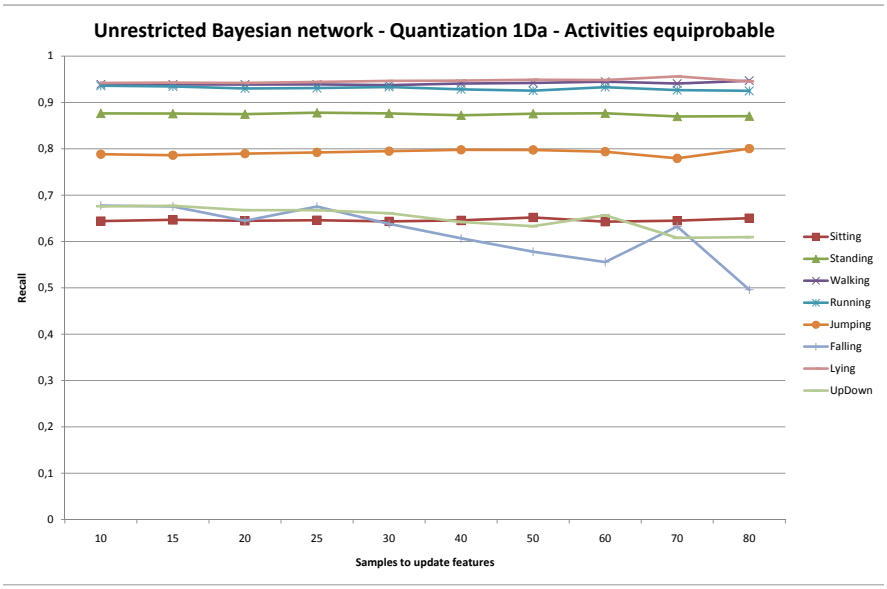


(a)

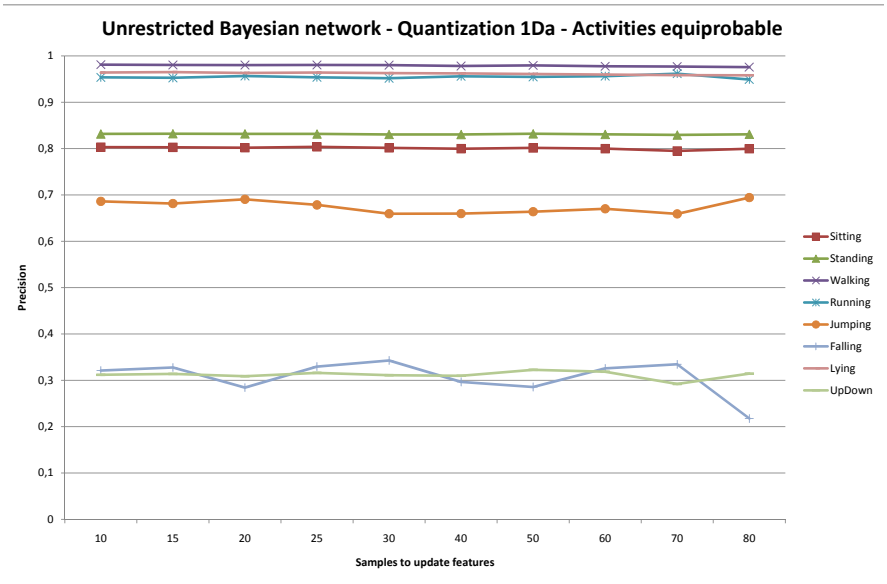


(b)

Figure 6.9: Recall and precision for the four-fold cross-validation test for every activity considering the Naïve Bayes approach for the discretization 1Da.

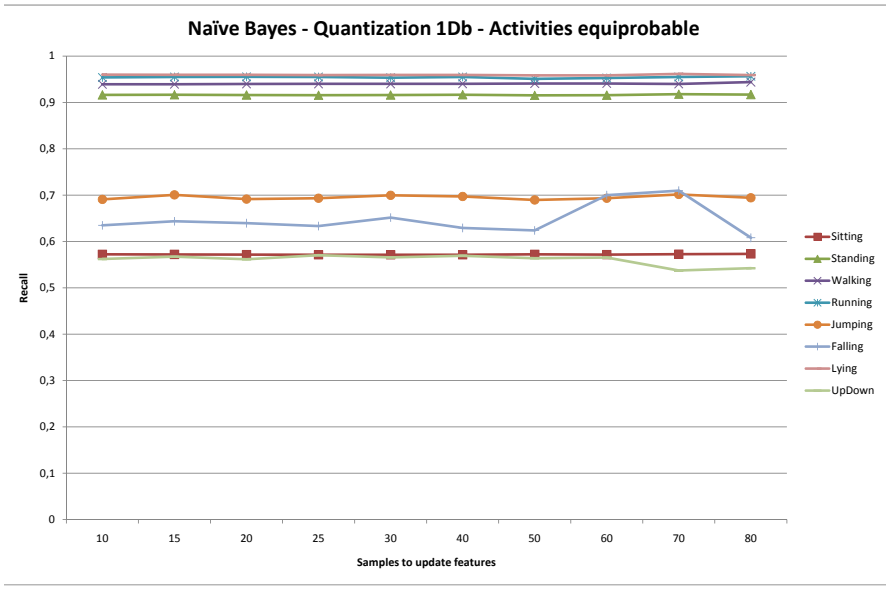


(a)

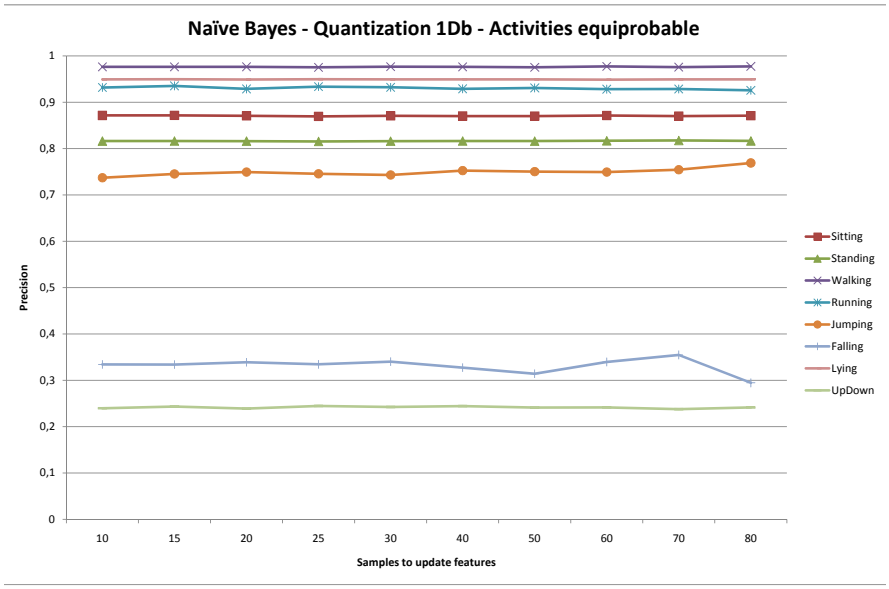


(b)

Figure 6.10: Recall and precision for the four-fold cross-validation test for every activity considering the unrestricted Bayesian network approach for the discretization 1Da.

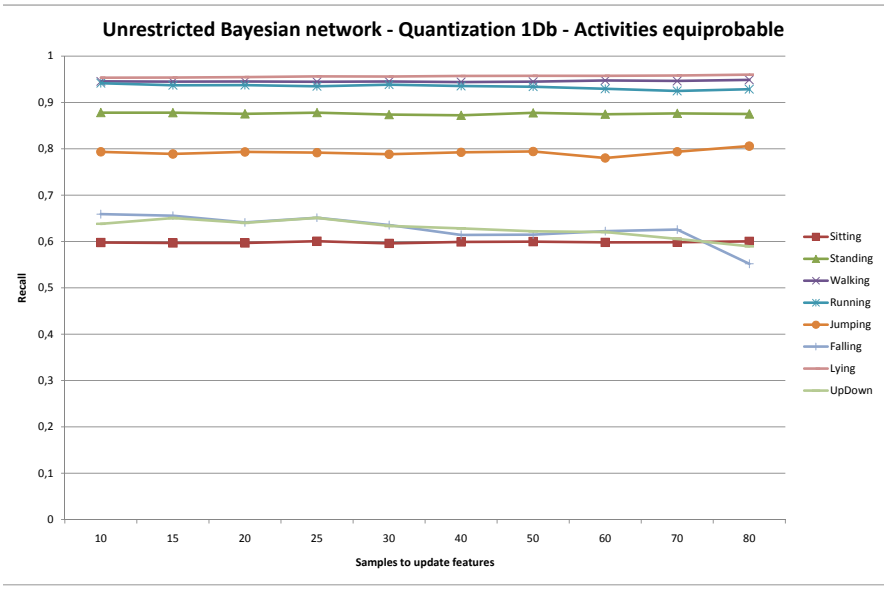


(a)

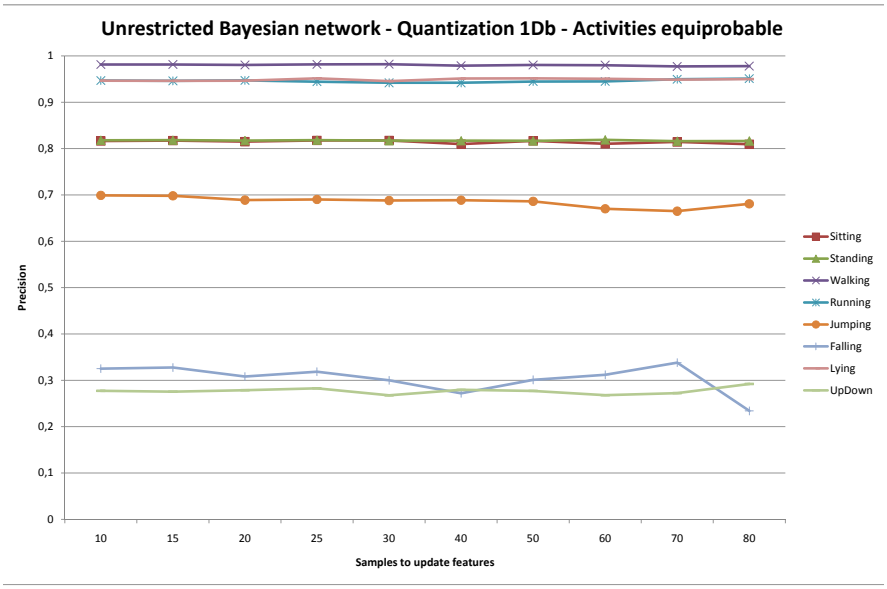


(b)

Figure 6.11: Recall and precision for the four-fold cross-validation test for every activity considering Naïve Bayes approach for the discretization 1Db.

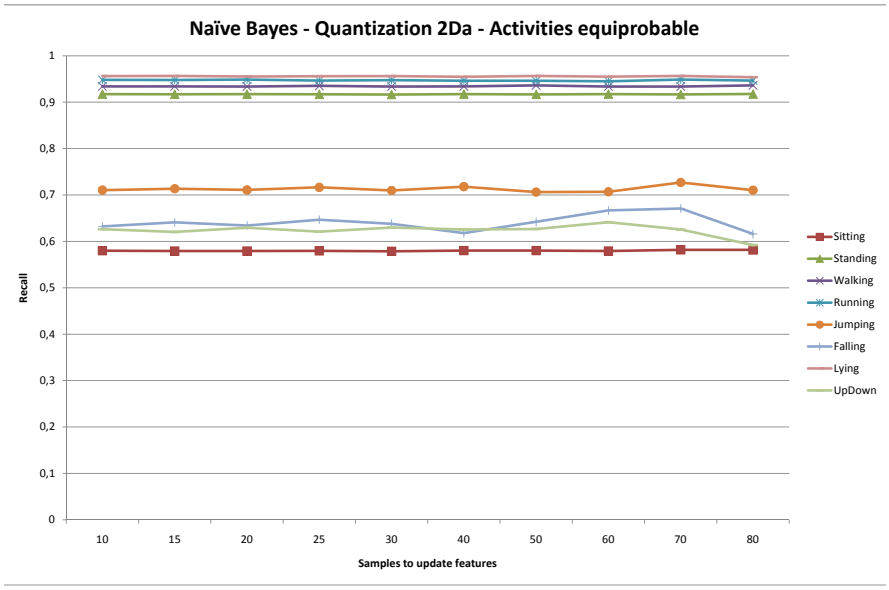


(a)

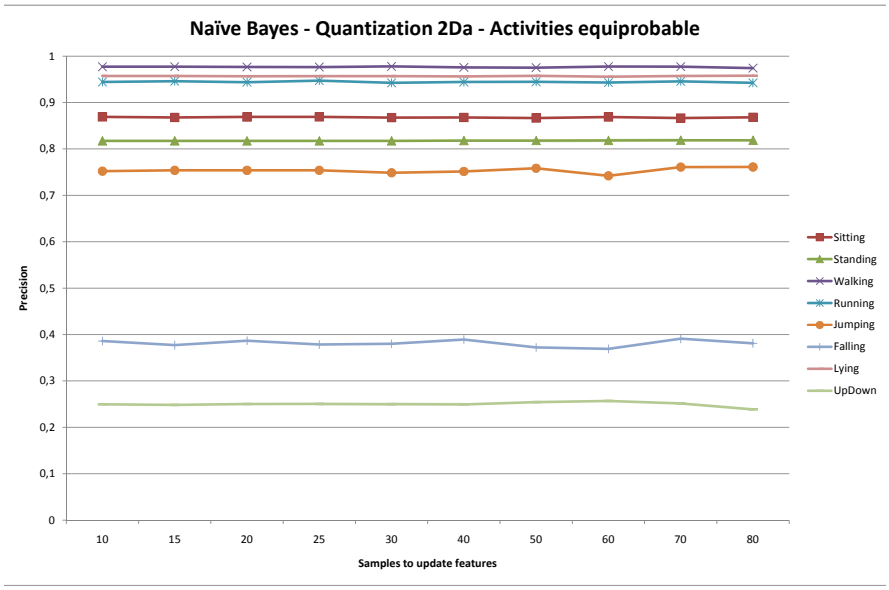


(b)

Figure 6.12: Recall and precision for the four-fold cross-validation test for every activity considering the unrestricted Bayesian network approach for the discretization 1Db.

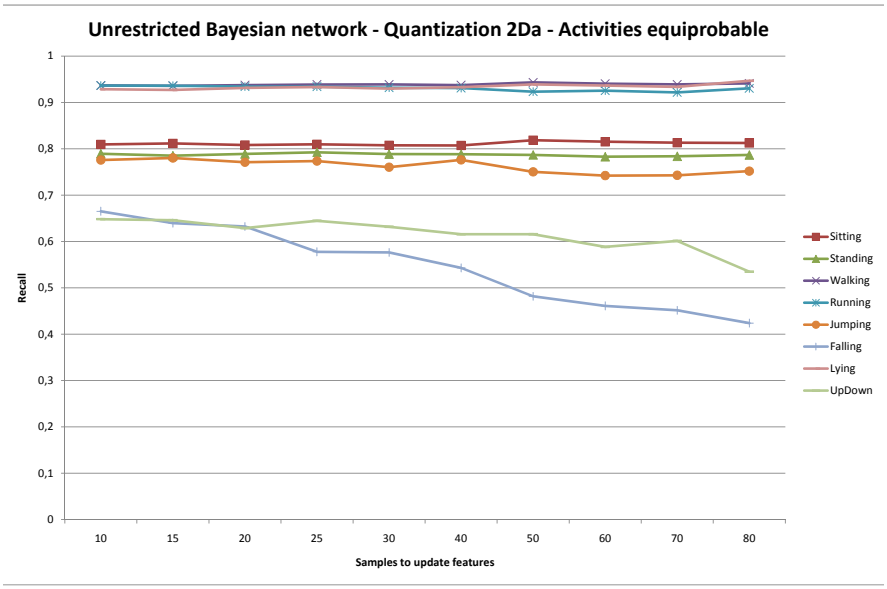


(a)

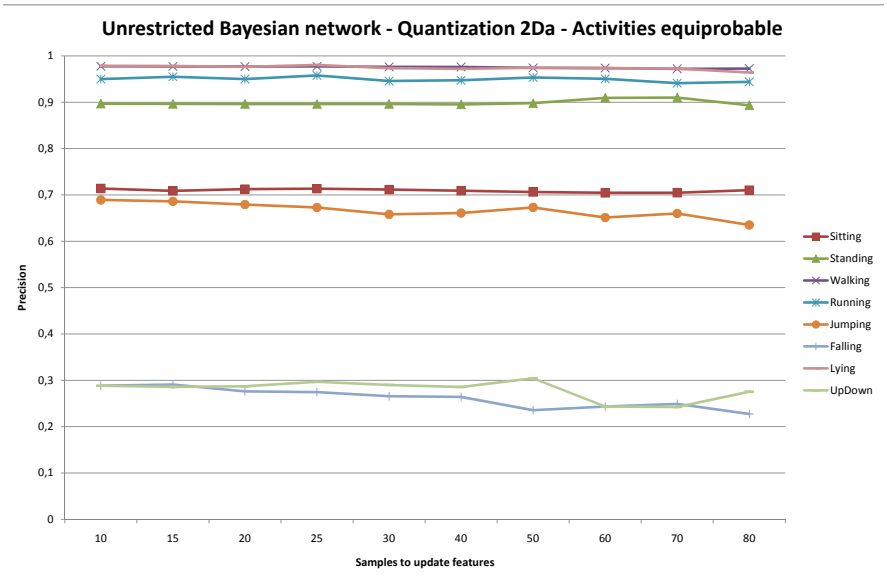


(b)

Figure 6.13: Recall and precision for the four-fold cross-validation test for every activity considering the Naïve Bayes approach for the discretization 2Da.

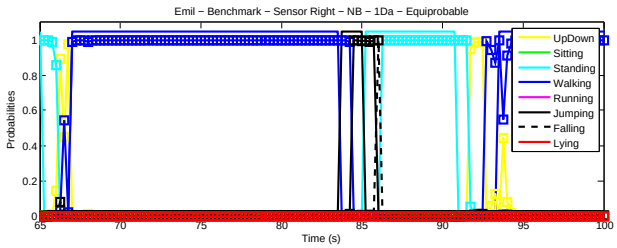


(a)

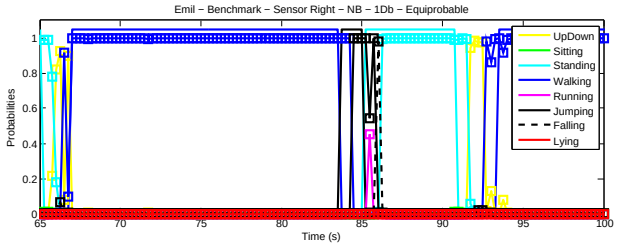


(b)

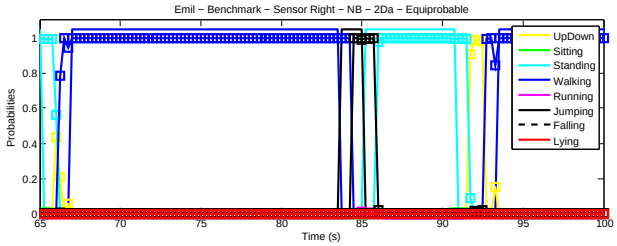
Figure 6.14: Recall and precision for the four-fold cross-validation test for every activity considering the unrestricted Bayesian network approach for the discretization 2Da.



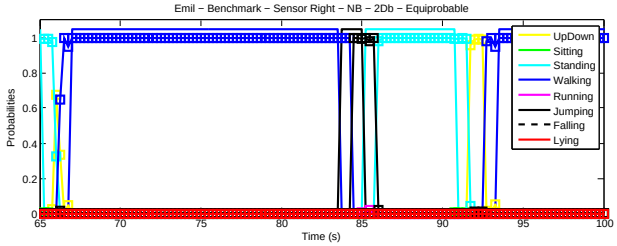
(a)



(b)

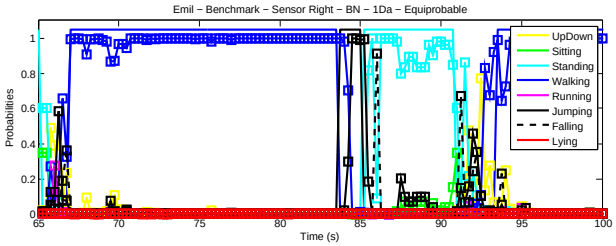


(c)

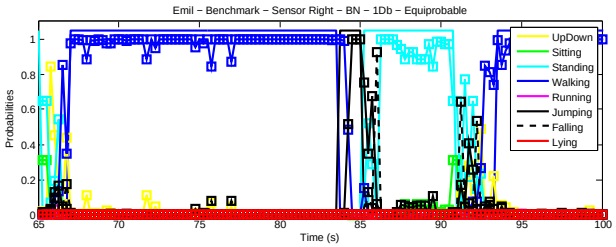


(d)

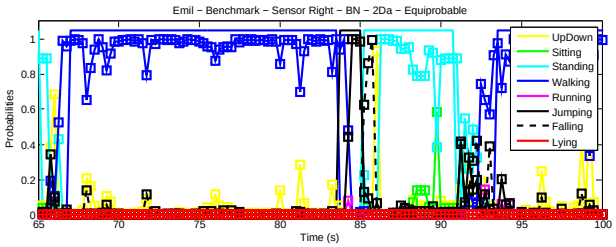
Figure 6.15: Naïve Bayes inference of the sequence *walking-jumping-standing* for all the discretizations of the features. The prior probabilities of *activity* are set to equiprobability. The top line represents ground truth, the curves below the estimated probabilities.



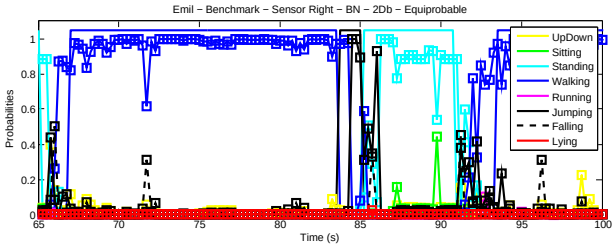
(a)



(b)

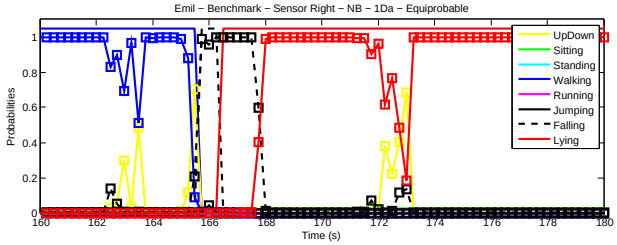


(c)

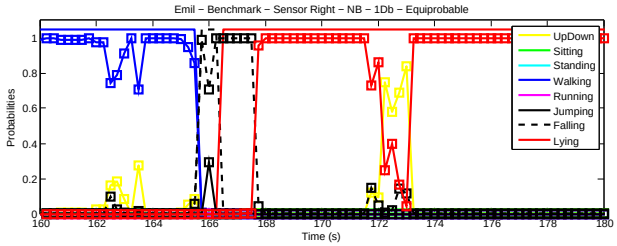


(d)

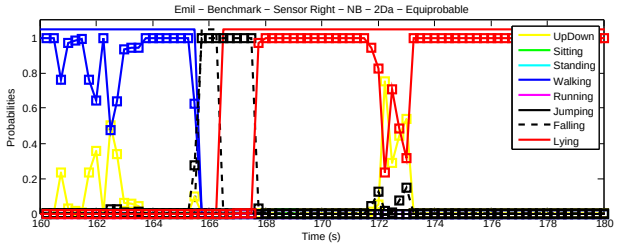
Figure 6.16: Unrestricted Bayesian network inference of the sequence *walking-jumping-standing* for all the discretizations of the features. The prior probabilities of *activity* are set to equiprobability. The top line represents ground truth, the curves below the estimated probabilities.



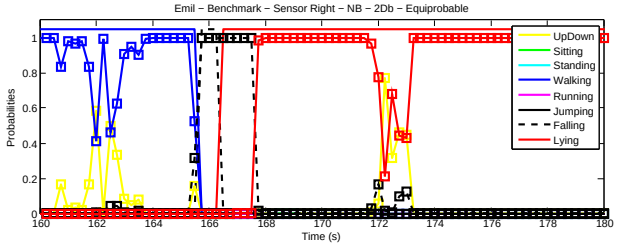
(a)



(b)

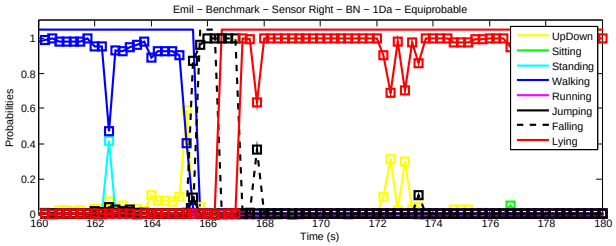


(c)

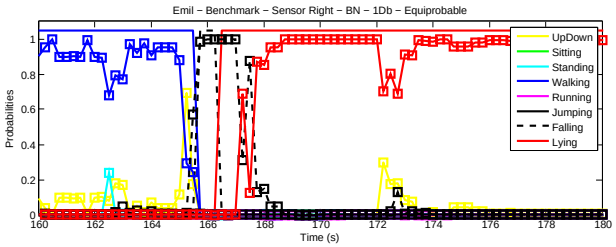


(d)

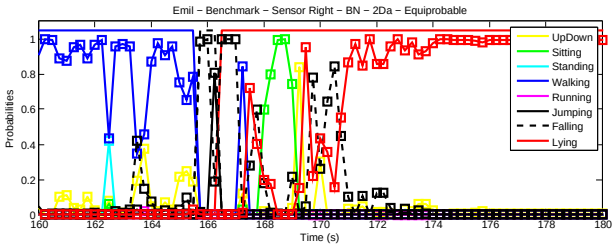
Figure 6.17: Naïve Bayes inference of the sequence *walking-falling-lying* for all the discretizations of the features. The prior probabilities of *activity* are set to equiprobability. The top line represents ground truth, the curves below the estimated probabilities.



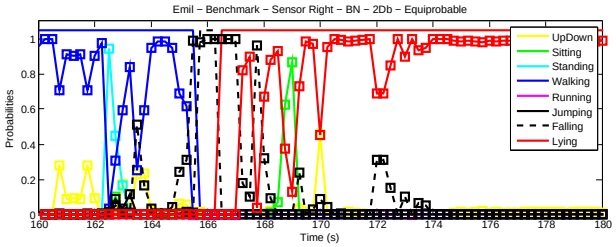
(a)



(b)



(c)



(d)

Figure 6.18: Unrestricted Bayesian network inference of the sequence *walking-falling-lying* for all the discretizations of the features. The prior probabilities of *activity* are set to equiprobability. The top line represents ground truth, the curves below the estimated probabilities.

down. The resubstitution test shows that computing evidence samples at a lower frequency improves the inference results. A possible reason for the improvement of the results for *falling* and the transition, is the reduction of the noise introduced in the system when the sliding window is getting into the activity but mostly contains samples of the past activity. However, obtaining evidence each 0.8 seconds has the risk of not getting enough evidence of the short-time activities that will help the inference of the dynamic approach with the Grid-based filter. With respect of the results of the four-fold cross-validation test, precision and recall of the unrestricted Bayesian network approach for *falling* and transition are worse as the update frequency of the features increases. Concretely, the discretizations in 2D achieves the worst results. The main causes for this behavior are:

- The collection of data for the data set consisted of the performance of a sequence of activities defined in advance. It means that for short-time activities, the past activity was *standing* in most of the cases. Therefore, when the sliding window is getting into the short-time activity, it take samples of the past activity *standing*. Besides, as the short-time activities are difficult to characterize, the features of the final set can take different values during two different *fallings* or *jumpings*. Consequently, the most significant state in these activities, as it is common in most of them, is the beginning, where the sliding window is getting into these activities and processing samples of the past activity. The final set feature values obtained in this processing is learnt by the training algorithm as a meaningful state and could produce false positives (see Chapter 3 for more details), it means, could infer that the user's current activity is a short-time activity while it is, most probable, a static activity. This effect can be observed in Figures 6.15 and 6.16, where *falling* was inferred while *standing* was performed. Also, after *falling*, the activity that was done in the data set was *lying*. It means that the evidence taken where the ground truth was *lying* have been computed with samples of *falling*. As soon as more states of the required are defined for the features, the unrestricted Bayesian network will try to fit the data set, and the overfitting effect could be more severe as can be observed in Figures 6.18 (c) and 6.18 (d). Same observations can be done for

the transition.

- The risk of overfitting the data is not only related to the noise introduced by the sliding window. As can be seen in Table 4.7, not all the features considered are really representative for all the activities. Consequently, during the performance of one activity, some of the features could take any value of the defined states. Besides, some activities consist of the execution of different phases or movements where the features could take very different values (such as *falling* or *jumping*). If the network fits the data set, it will not extract the relevant information of the features behavior and the activities. Consequently, the discretizations in 2D are more sensitive to the problem of overfitting the data, not extracting the relevant information of the features behavior and the activities and achieving *unstability* in the inference that is not observed in Naïve Bayes approaches for the same discretization.
- Finally, the noise introduced through the processing of the sliding window is more critical for short-time activities and trasition where the quantity of data of this activity or transition is very low and the noise produced by the sliding window could be important (for activities such as *standing* or *walking*, enough data is recorded and the relevant data to noise ratio is much higher). If new evidence is computed at a lower frequency, the training data from the short-time activities and the transition decreases. The lack of relevant training data produces a bad inference of these activities.

These three causes could produce an incomplete learning process of the behavior of the activity and transition in terms of probability that is more severe in the discretizations in 2D.

6.3 Static and Dynamic Inference Comparison

This section will provide the inference of the activities of the static and dynamic approach for the discretization 1Da that was identified as the most adequate due to the analysis of the results of the tests done above. The data used to evaluate both approaches is the data collected from the two unknown

subjects for the system, Emil and Sinja, allowing them individual freedom in the performance of the activities.

The results of the data recorded from Emil are shown in Figures 6.19, 6.20, 6.21, 6.22 and 6.23. The results of the data recorded from Sinja for the discretization 1Da are shown in Figures 6.24, 6.25, 6.26, 6.27 and 6.28.

The evaluation of the activities *standing, sitting* and *standing* for both subjects are shown in Figures 6.19 and 6.24. The distinction between *standing* and *sitting* is based on the attitude of the sensor when the subject is involved in them. However, the attitude of the sensor during *sitting* could be the same than during *standing*, depending on the way the subject sits. This confusion is alleviated including the dynamic information in the system.

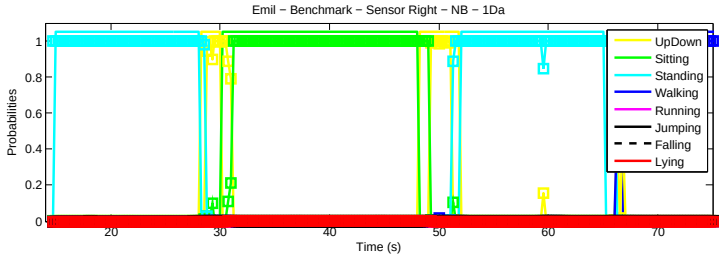
The activities *walking, jumping, standing* and *walking* for both subjects are evaluated in Figures 6.20 and 6.25. As expected, based on the results of four-fold cross-validation and resubstitution, the unrestricted Bayesian network approach improves the results in the inference of the activity *jumping*, while *walking* and *standing* are identified by the four recognition algorithms. Moreover, the dynamic approach improves the results of the static approach.

The evaluation of the sequence *walking, running, jumping* and *standing* for both subjects is shown in Figures 6.21 and 6.26. On the one hand, the distinction of *running* and *jumping* is improved by the approach based on unrestricted Bayesian network. On the other hand, *walking* and *running* are not confused by Naïve Bayes, but the unrestricted Bayesian network approach provides good results as well as soon as the dynamic information is included.

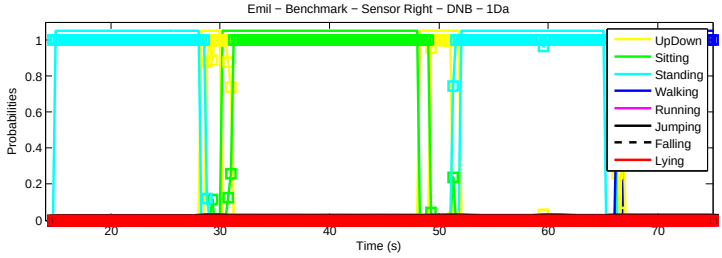
The sequence *walking, falling* and *lying* for both subjects is evaluated in Figures 6.22 and 6.27. Even if the results of four-fold cross-validation and the apparent error point out that *falling* could not be well detected by the system, inference results shows how the system is able to detect it. Details of the evaluation of the activity *falling* are given in Figures 6.23 and 6.28. The inference results for *lying* are also promising.

Even if it has been proven that the system is able to recognize the activities nearly 100% of the time, misclassification of the activities is done at the begining of them, obtaining not that promising results in terms of precision and recall as it can be observed in Figure 6.29. The reasons for this observation are:

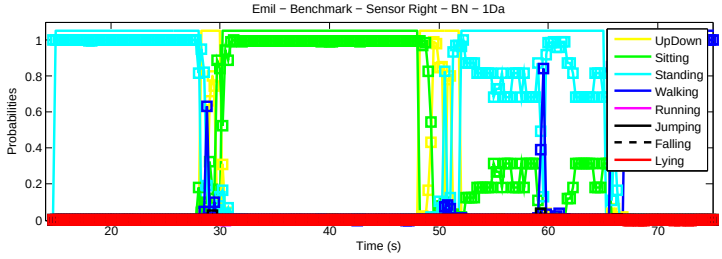
- The sliding window takes samples of the previous activity. In activities such as *falling* where the duration takes from 3 o 5 evidence samples,



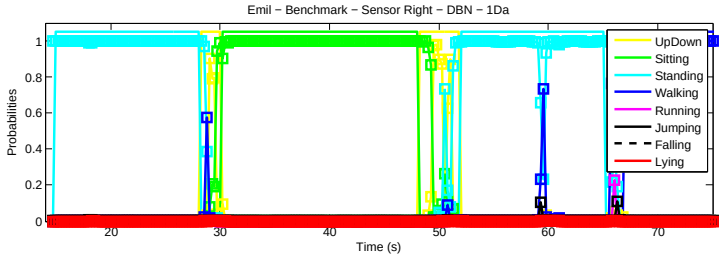
(a)



(b)

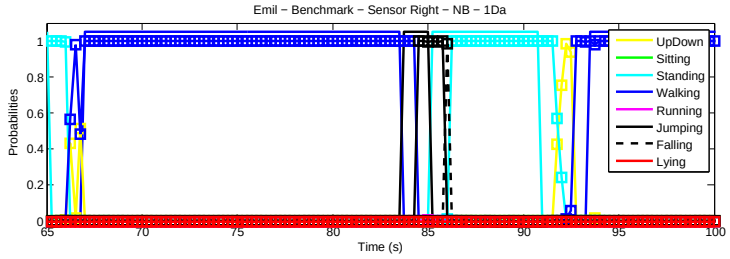


(c)

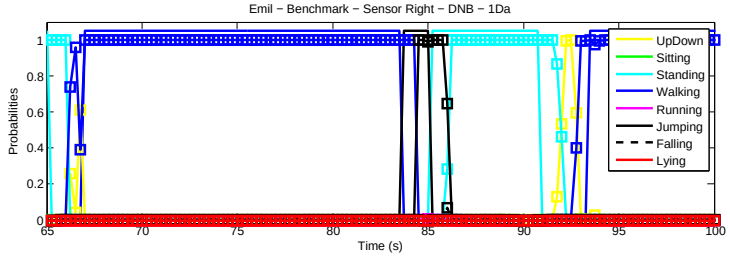


(d)

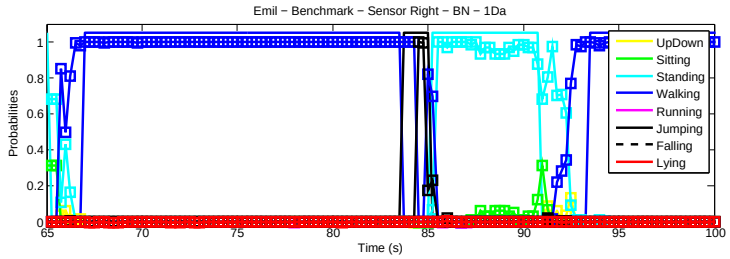
Figure 6.19: Inference results of the sequence *standing, sitting* and *standing* for the male subject, Emil. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.



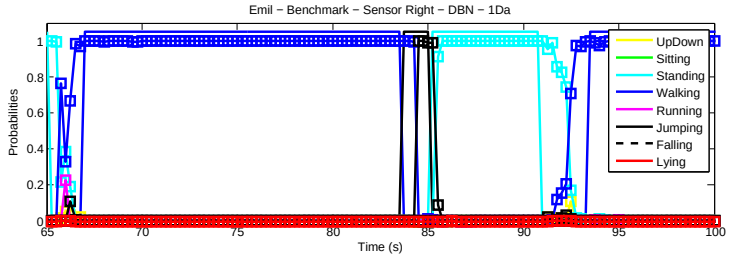
(a)



(b)

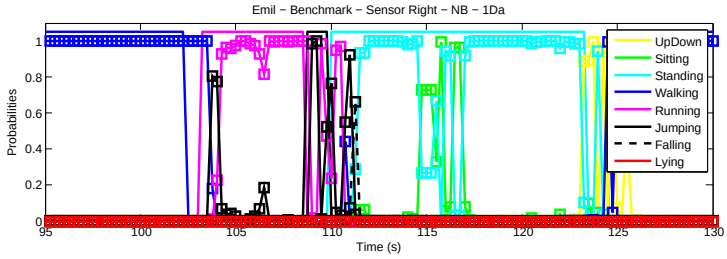


(c)

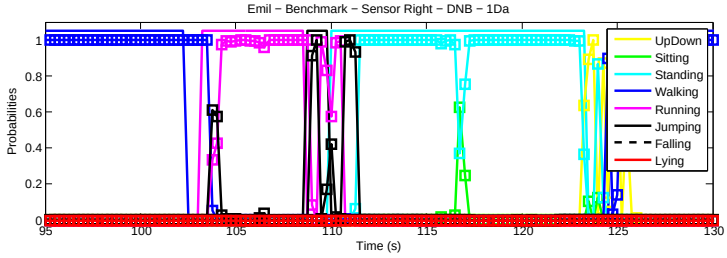


(d)

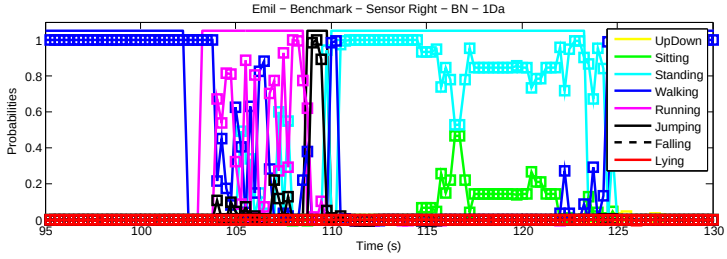
Figure 6.20: Inference results of the sequence *walking, jumping* and *standing* for the male subject, Emil. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.



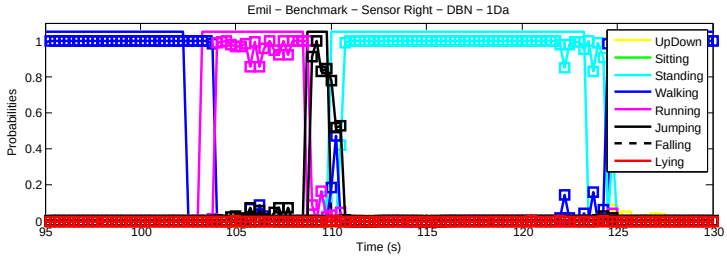
(a)



(b)

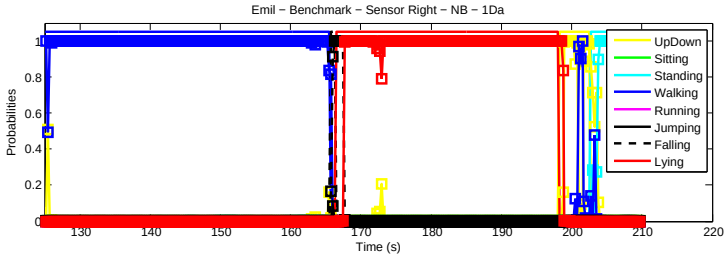


(c)

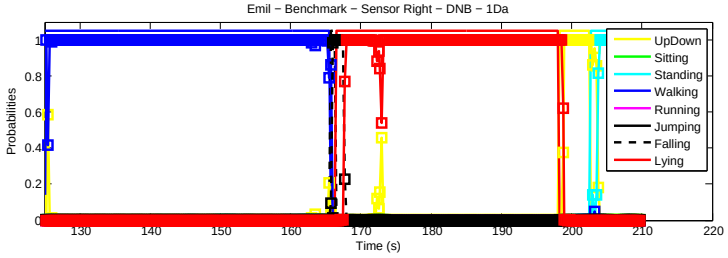


(d)

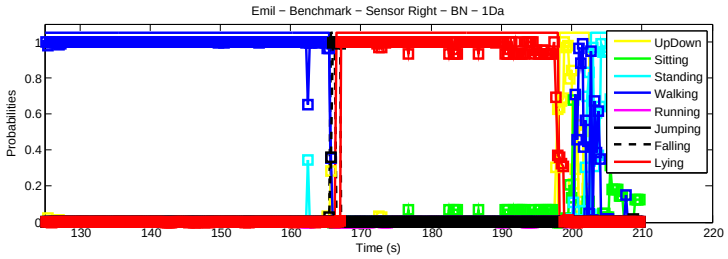
Figure 6.21: Inference results of the sequence *walking, running, jumping* and *standing* for the male subject, Emil. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.



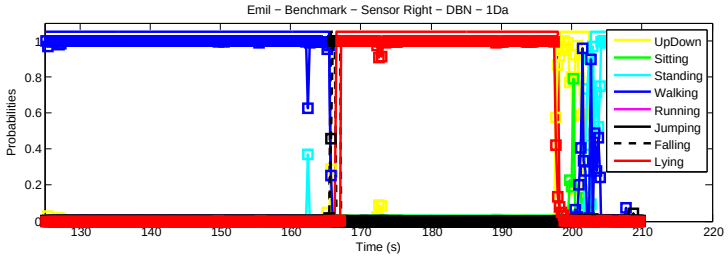
(a)



(b)

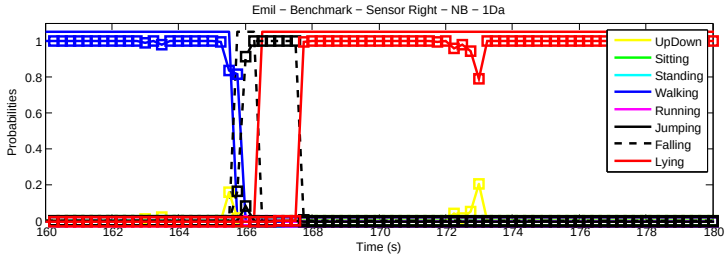


(c)

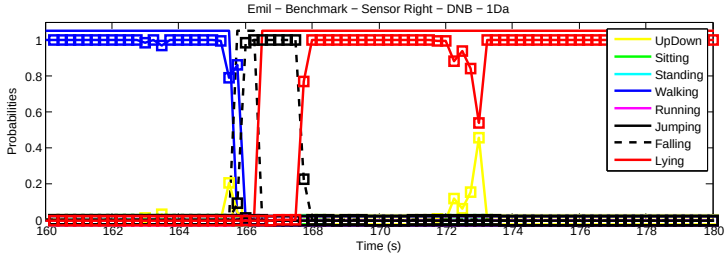


(d)

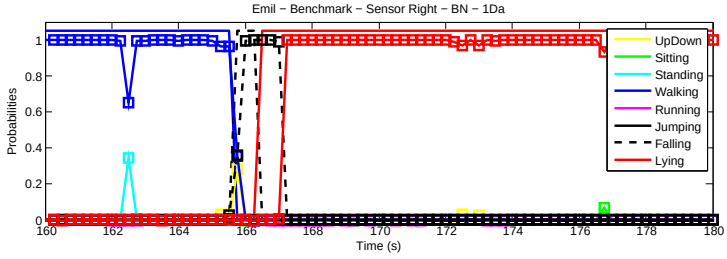
Figure 6.22: Inference results of the sequence *walking, falling* and *lying* for the male subject, Emil. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.



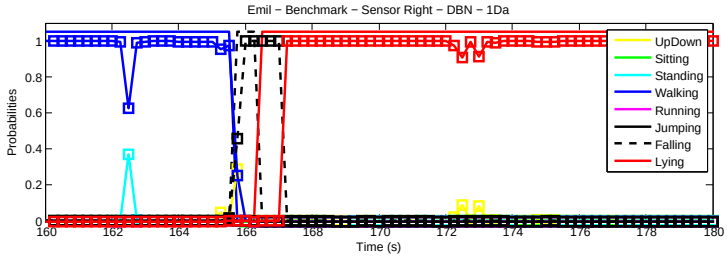
(a)



(b)



(c)



(d)

Figure 6.23: Detail of the inference results of *walking*, *falling* and *lying* for the male subject, Emil. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.

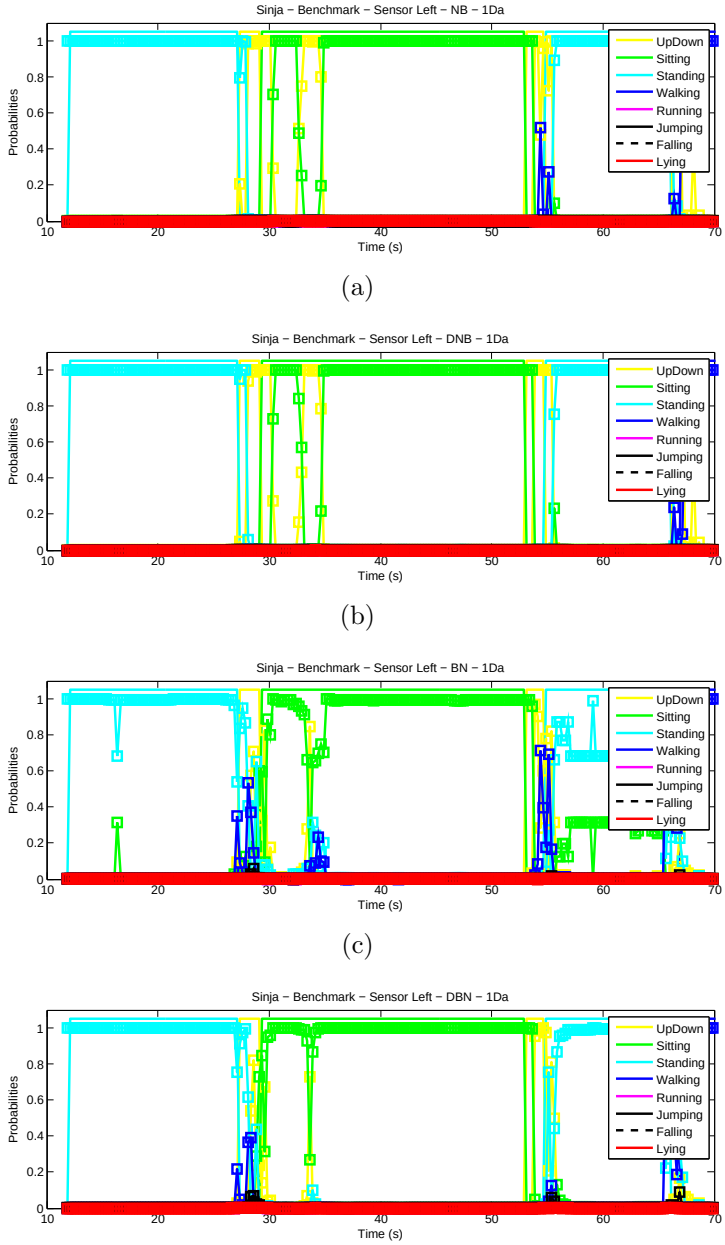
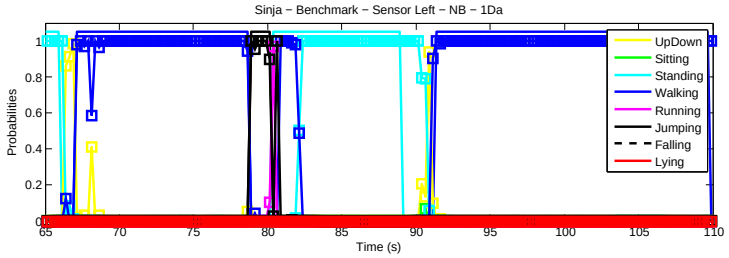
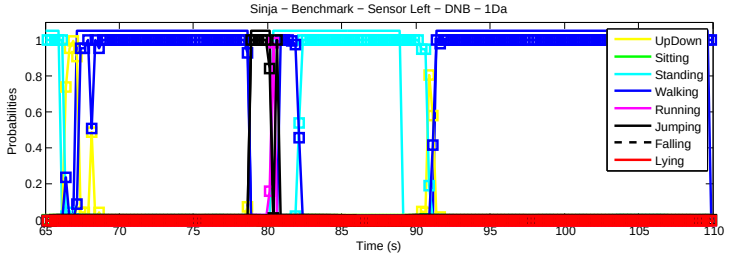


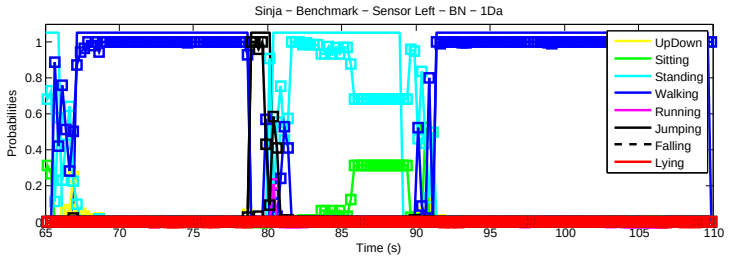
Figure 6.24: Inference results of the sequence *standing*, *sitting* and *standing* for the female subject, Sinja. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.



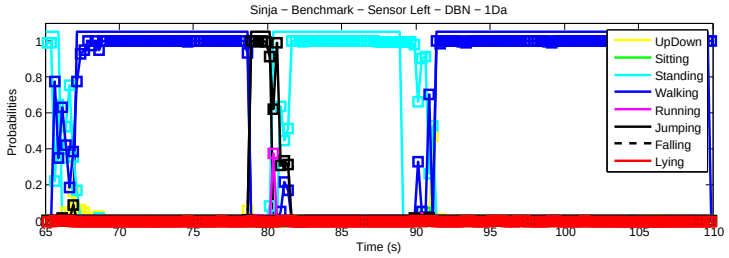
(a)



(b)

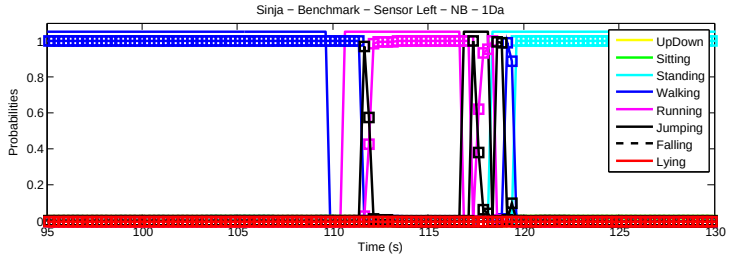


(c)

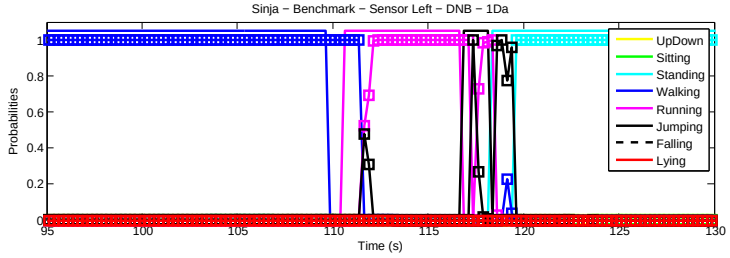


(d)

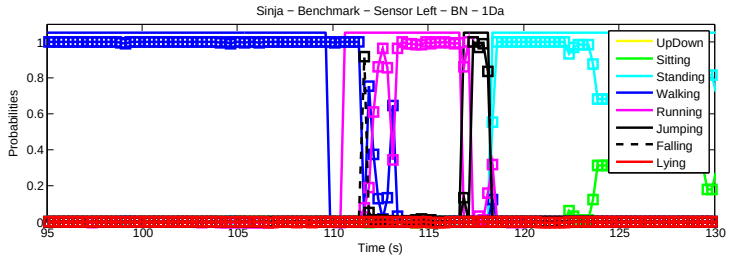
Figure 6.25: Inference results of the sequence *walking, jumping, standing* and *walking* for the female subject, Sinja. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.



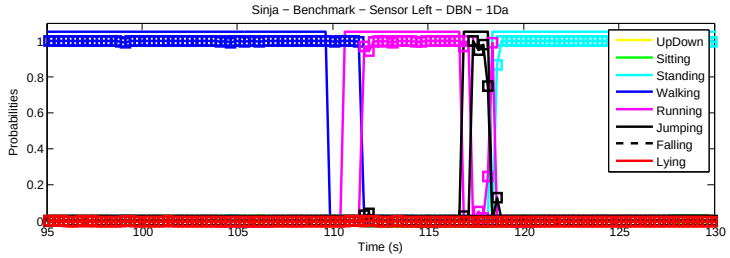
(a)



(b)

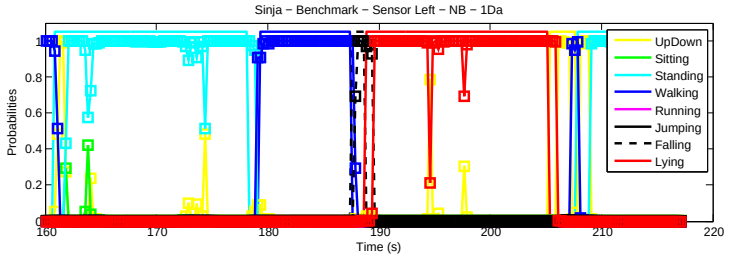


(c)

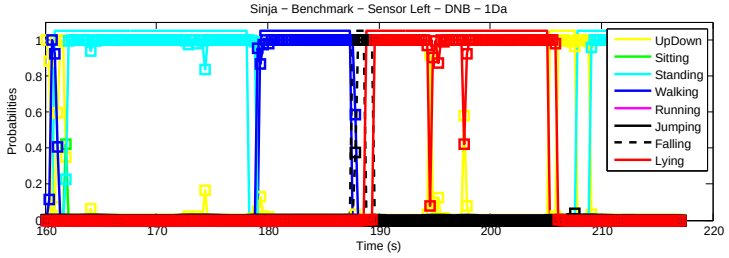


(d)

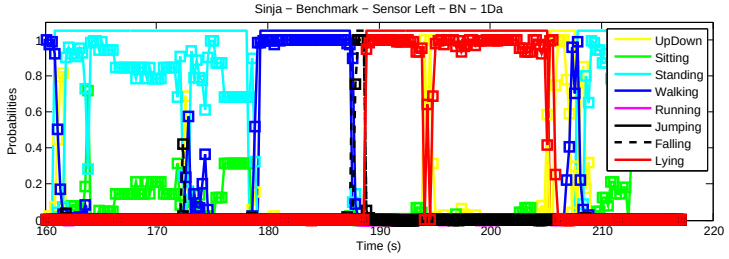
Figure 6.26: Inference results of the sequence *walking, running, jumping* and *standing* for the female subject, Sinja. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.



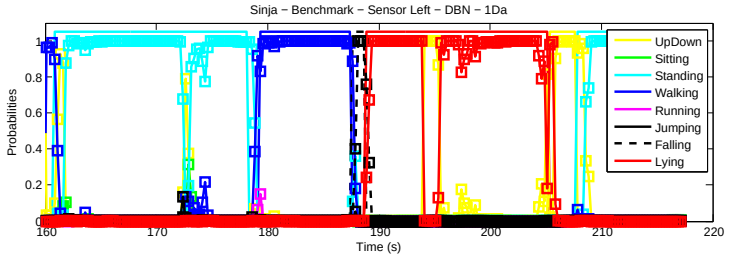
(a)



(b)

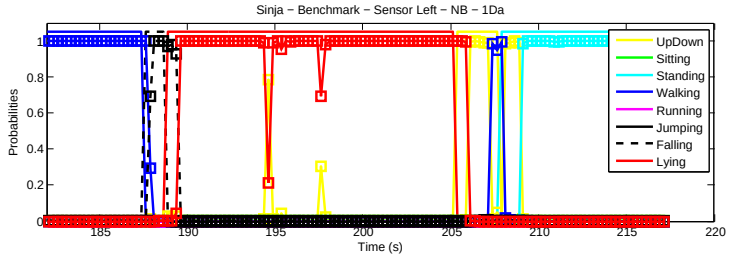


(c)

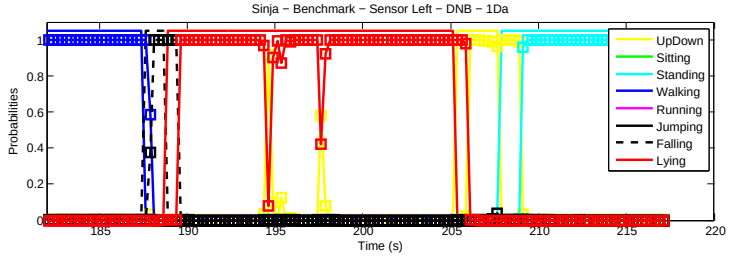


(d)

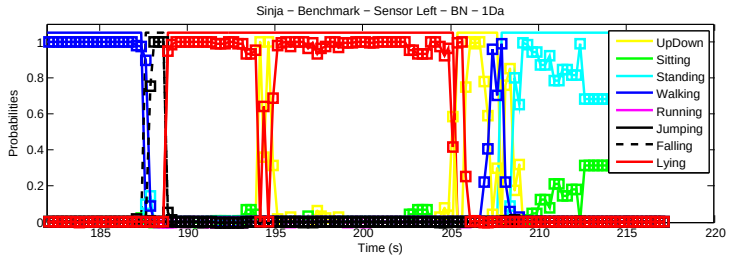
Figure 6.27: Inference results of the sequence *standing, walking, falling* and *lying* for the female subject, Sinja. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.



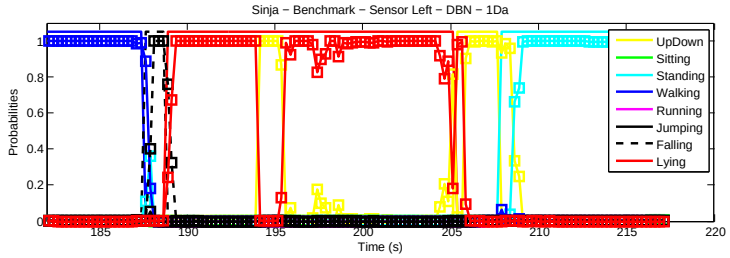
(a)



(b)

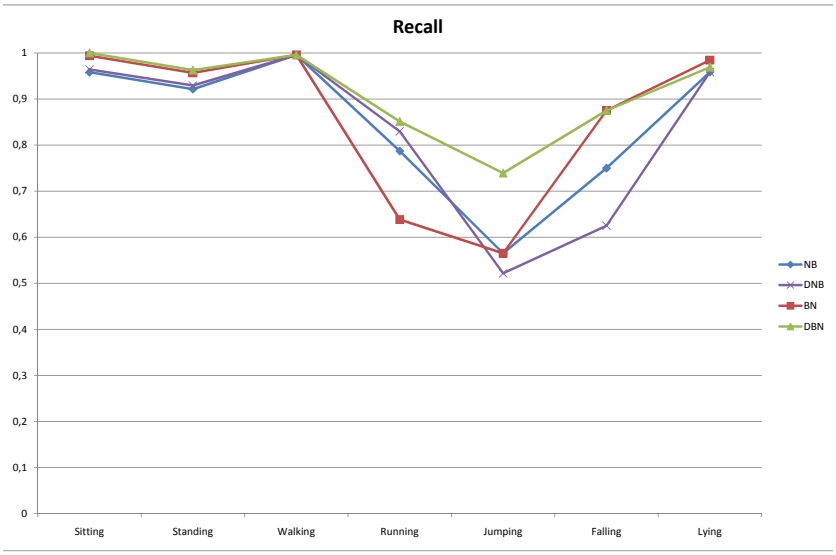


(c)

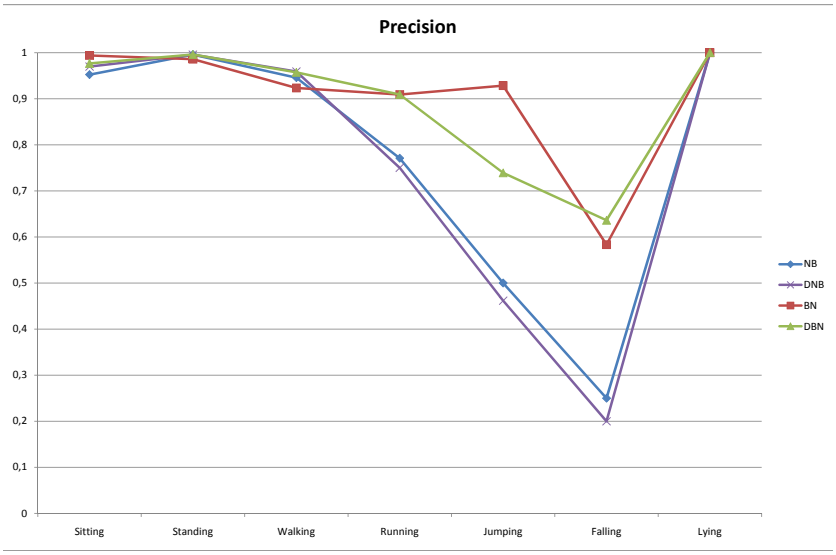


(d)

Figure 6.28: Detail of the inference results of the sequence *walking, falling, lying* and *standing* for the female subject, Sinja. A constant line on top of these figures depicts the ground truth, colors identify the current activity. Below the ground truth and in a scale from zero to one are plotted the estimated probabilities of every activity.



(a)



(b)

Figure 6.29: Recall and precision obtained for every activity and every recognition algorithm for the discretization 1Da. The update frequency of the features is set to 4 Hz. The unrestricted Bayesian network approaches achieve better results than the recognition algorithms based on Naïve Bayes.

a recognition delay of 2 samples decreases the precision and recall of the system significantly. It can be observed how the inference in the beginning of an activity or a transition is influenced by this fact. It indicates a recognition delay in the system.

- The labeling of the ground truth and the data have been done manually, which includes human error as one of the causes of achieving a not 100% accurate result.

To measure the recognition delay of the system, most of the features of the final set are defined for a window length of 128 samples (or 1.28 seconds). It is sensible to postulate that at least the 50% of the window should be inside the current activity in order to achieve an accurate inference result (considering also that features computed over window lenghts of 32 and 64 samples are considered in the inference and at that point of time, they will be contained in the current activity). A 50% of the samples of a window of 128 samples implies that at least 64 samples or 0.64 seconds (at a sample frequency of 100 Hz) of the current activity should have been passed in order to be recognized. If evidence is obtained every 0.25 seconds, the recognition delay approaches over 2 evidence samples. So, precision and recall should be computed without considering the first two evidence samples at the beginning of every activity as an approximation to estimate this recognition delay.

Recall and precision for every activity considering the recognition delay for the BN and DBN approaches, that achieve the best results in average as can be seen in Figure 6.29, for the discretization 1Da and the update frequency of the features at 4 Hz are shown in Table 6.2.

As can be seen in in Table 6.2, recall is improved by the dynamic approach. With respect to precision, the dynamic approach also improves the results (specially for *falling*) in most of the activities except for *jumping* and *sitting*, that show a lightly decrease of the precision with the dynamic approach. In any case, dynamic approach improves the results obtained in the inference of the activities and the quality of the inference.

Taking into account the recognition delay, the results of the inference of the current activity improved, achieving recalls bigger than 0.93 for all the activities.

The evaluation of the system during the transitions is not taken into account, although they were modeled in the system to implement the dynamic model. The recognition of these transitions should be worse than for the

Static unrestricted Bayesian network recognition algorithm							
	Sitting	Standing	Walking	Running	Jumping	Falling	Lying
Recall	0.99	0.96	1	0.69	0.66	1	0.99
Precision	0.99	0.98	0.94	1	1	0.57	1
Dynamic unrestricted Bayesian network recognition algorithm							
	Sitting	Standing	Walking	Running	Jumping	Falling	Lying
Recall	1	0.98	1	0.93	0.93	1	0.98
Precision	0.97	1	0.98	1	0.93	0.8	1

Table 6.2: Precision and recall for every activity considering the recognition delay. Static and dynamic inference of the unrestricted Bayesian network approach is compared for the discretization 1Da. Features are computed at 4 Hz.

activities, as they were not regarded throughout the feature identification process, but only necessary now to improve the results of the dynamic filter.

6.4 Real Time Evaluation

Discretization 1Da is considered and the frequency update of the features was set at 4 Hz for the real time evaluation. To estimate the execution time, 780 iterations of the feature computation process and inference were done in our implementation. The data were processed using an Intel Core 2 Duo microprocessor, E8400, at 3.00 GHz with 2 GB of RAM. The operating system is Windows XP.

The results for the static unrestricted Bayesian network and dynamic unrestricted Bayesian network inference are shown in Table 6.3. The results for the static Naïve Bayes network, dynamic Naïve Bayes network inference and the length of feature computation in our implementation are also provided, in order to compare the complexity of these processes. In this table, the 25th-percentile, the 50-th percentile, the 75-th percentile, the mean, the minimum and the maximum of the execution times obtained are given.

The complexity of a BN is determined by the number of nodes contained and the size of their corresponding CPTs. The memory size of unrestricted Bayesian network and Naïve Bayes network is given in Table 6.4, where the difference of complexity of both networks can be observed.

The feature computation is not time-consuming, as can be seen in Table 6.3. The difference between the inference of Naïve Bayes and the unrestricted

Operation	Q_1 (ms)	Q_2 (ms)	Q_3 (ms)	μ (ms)	Min.	Max.
Feature computation	1.43	1.45	1.47	1.5	1.4	4.1
Static Naïve Bayes inference	0.31	0.319	0.32	0.34	0.29	2.17
Dynamic Naïve Bayes inference	0.33	0.34	0.35	0.36	0.3	3.26
Static unrestricted Bayesian network inference	5.6	7.2	8.3	7.2	3.9	27.7
Dynamic unrestricted Bayesian network inference	6	7.7	9	7.7	4.1	18

Table 6.3: Results related to the execution time of feature computation and inference process. The 25th-percentile, the 50-th percentile, the 75-th percentile, the mean, the minimum and the maximum of the execution times obtained for the feature computation and inference process based on Naïve Bayes and unrestricted Bayesian network are shown. Discretization 1Da is considered and the frequency update of the features was set at 4 Hz. The data were processed using an Intel Core 2 Duo microprocessor, E8400, at 3.00 GHz with 2 GB of RAM.

Network	Memory size
Naïve Bayes network	33,3 KB
Unrestricted Bayesian network	3,62 MB

Table 6.4: Memory size of the Bayesian networks for the discretization 1Da obtained after the training process (unrestricted Bayesian network) and for the assumption of Naïve Bayes.

Bayesian network considering static and dynamic approaches is notable. Inference based on the Naïve Bayes network takes from 0,3 to 0,4 ms. However, the unrestricted Bayesian network takes from 7 to 8 ms considering static and dynamic approaches, due to the complexity of this network. Consequently, the execution time of the inference based on the multiplication of the conditional probabilities of the network depends highly on the complexity of this network. Inference based on the Grid-based filter for the dynamic approach takes around 1 up to 2 ms more than the static approach. Dealing with the HMM increases the inference time, but the main computational cost in terms of execution time comes from the Bayesian network used. In any case, these execution times are very short.

Chapter 7

Conclusions and Outlook

This chapter attempts to compile the most important conclusions that have been reached throughout this work. Besides, future research lines related to this topic are pointed out.

7.1 Conclusions

A new approach for human motion related activities has been presented in this Master Thesis. This approach allows to recognize the user's activity using the information from an inertial measurement unit, placed on the belt. Bayesian methods for activity recognition are applied obtaining, for most of them, reliable and promising results in real-time. Therefore, the list of objectives presented in Chapter 1 has been successfully achieved.

Throughout the stages of this master thesis, different observations and conclusions have been reached and explained in the correspondent chapters. The following paragraphs will point out the most relevant ones:

- Considering different window lengths for the processing of the signal is essential taking into account the duration of the activities that need to be inferred.
- The collected data set to train the networks was sufficient to achieve good recognition results.
- Labeling the data and the sliding window process of the signal (that has been identified as the main source of *noise* in the system) could be rele-

vant processes if classifiers that are not based on statistical algorithms are used.

- The outcome of the sensor was synthesized identifying the body frame vertical and horizontal acceleration as the main source of information, as well as the norm of the acceleration. Considering the body frame approach will probably provide promising results for the distinction of other human motion activities.

- Other features were selected and explained from the point of view of the physics involved in each activity. Also, the classifier used allows the researcher to interpret the final results. This trend of being able of interpreting what happens in the system and the human being during the performance of these activities, has produces a system able to work over subjects whose data were not collected to train the Bayesian networks.

- The final set of features was sufficient to distinguish the activities. Most probable, this final set of features could be reduced achieving as spectacular results in the inference of activity, but the characterization (and not only the distinction) of these activities was one of the objectives of this Master Thesis as, in the future work, new activities could be included, even the *unknown* activity (that will include transitions or other possible activities) and could require a good characterization of all of them.

- Discrete Bayesian networks achieve promising results in the inference of the activity, but the process of the feature discretization and feature reduction should be done carefully. Naïve Bayes seems to be more sensitive to the feature reduction process while machine-learnt Bayesian networks are more sensitive to the feature discretization process. The inference algorithm under the constraint that all input nodes are observed makes them suitable for real-time applications.

- Activities such as *jumping* or *falling* in which different phases can be observed, are better characterized by the unrestricted Bayesian network.

- The frequency update of the features has not been a crucial parameter in the inference quality of the static approaches. The requirement to

set this parameter is related to the training of the network and the Grid-based filter. On the one hand, this parameter is set according to get sufficient evidence in order to train the Bayesian network. On the other hand, the Grid-based filter should achieve better recognition results if it has got more evidences of every activity, specially relevant for short-time activities.

- Dynamic Bayesian state estimation techniques like the Grid-based filter provides useful information that strongly improves the quality of the inference. The transition model implemented is important for achieving good inference results. The cost of implementing this filter is the increment of the execution time with respect to the static approach. A comparison of static and dynamic Bayesian inference on activity recognition with preliminary results of this Master Thesis has been published in [57].

- With respect to the distinction and inference of the activities considered in this approach, the most difficulties that have been dealt are:

  - *Standing* versus *sitting*: they are just distinguished by the attitude of the sensor. The dynamic model is required to achieve good inference results in both activities.
  - Short-time activities inference: the behavior of these activities is difficult to capture in a limited set of features. There is not one single feature that could cluster these activities with respect to the others. The solution was to provide a set of features where all the physical phenomena that occur during these activities are given.
  - Transitions: they were not the objective of this project, but the inference of them, as they are short-time and difficult to characterize, is a challenge in activity recognition.

  Walking and *lying* are the activities that the system is able to recognize in any scenery. *Running* could be confused with *jumping*, as both activities are similar and could be clustered as *doing sports*.

- At the moment, the system provides the user's activity at 4 Hz. In the future, this system could be integrated using the machine-learnt Bayesian networks for activity inference in a microchip (thanks to *MEMS*

technology) and provide the user's activity at a lower rate with lower cost in terms of battery life.

7.2 Outlook

Future research lines in context inference using Bayesian methods are pointed out in the following paragraphs:

- Evaluate the system performance under naturalistic conditions. At least two unknown subject should collect data during one day of their lives. Video data with ground truth or the design of a labeling system in which there is no need of another subject to label the data are recommended. Activity inference real performance would be observed like that.

- Add new activities such as *riding a bike, going up in an elevator, going down in an elevator* or *going into a vehicle*. These activities could be defined as:

  - A sequence of different stages (as has been observed for *jumping*). For example, the activity *going into a vehicle* could be divided into *open the door, sit on the car, start moving, ...*
  - A sequence of physical movements or facts such as *stepping*, *move the trunk with respect to the legs*, *put the body down ...*

  This new system could be modeled as a hierarchical hidden Markov model (HHMM) where the features try to infer the stages of physical facts that occurs during the activities and an activity is the sequence or combination of these stages or physical facts. This level of detail could be unnecessary, depending on how the characterization of these new activities is done.

- Infer high-level activities such as *being in a meeting* or *going to work*. A hierarchical model of inference depending on the information type could be implemented. The system designed in this Master Thesis could be the one of the closest layers to the raw data of the sensor. *Activity* should be inferred at a higher rate while, at a level higher, e.g. the current situation comprising a set of activities, will have a much

lower frequency of change. A normal duration may be within minutes and the update rate decreases significantly.

- Acceleration sensor data could be used by an algorithm for pedestrian dead reckoning applications. This system could be used for tracking and navigation counting the steps of the subjects when *walking* or *running* is inferred. The subject's location will be a powerful information for context inference.

- Use the system in the pocket. To allow slight movements of the sensor without affecting activity inference, updates of the rotation matrix from sensor to body frame should be implemented. Possible ideas to estimate when this rotation matrix should be updated could be based on *walking* inference. *Walking* is highly recognized by the system, mostly from $|a|$ that does not depend on the frame of study, so, there are two options:

  - In the pattern of *walking*, the moment of time when the leg of the sensor hits the floor could be estimated through its study in detail. When the leg is completely straight (this could be detected by a change in the horizontal acceleration while hitting the floor can be estimated using the vertical acceleration), the rotation matrix from sensor frame to global frame is equal to the rotation matrix from sensor frame to body frame. Appendix B shows hypothesis done over a *walking* pattern. Video data and a detailed study of human gait could be useful.
  - Another option is the following one: after *walking*, if the subjects is performing a *static* activity, most probable this activity will be *standing* because *sitting* or *lying* after *walking* are not probable, even possible without any different activity (even for a short time) between both. This point of time will be useful for updating the rotation matrix from sensor to body as well.

- A possible approach, taking into account the results obtained in this Master Thesis, using Bayesian methods could be to perform automatically the process of the feature discretization and reduction. The discretization of the features can be implemented using an algorithm that detects meaningful states on them based on the histograms information of the feature respect of the activity. The process of feature

reduction and selection could be done through the learning of the Bayesian network structure that contains all these features and the target node. The features that are out of the Markov Blanket of the target node are identified as bad, redundant or unnecessary and the rest of the features can be used in this machine-learnt Bayesian network or in a Naïve Bayes Bayesian network. Finally, both recognition algorithms could be evaluated.

- The learning algorithm for unrestricted Bayesian networks tries to fit the data set. This data set has noise due to the processing of the signal taking samples of the past activity. The value of the features during the processing of the sliding window at the beginning of every activity are not relevant until the sliding window is inside the current activity and could be deleted. Possible future work is to use the information of the evolution in time of the features for every activity to filter the noise contained in the final data. This process of filtering should be done for critical activities such as *falling* where the quantity of data of this activity is very low and the noise produced by the sliding window could be important. For activities such as *standing* or *walking*, enough data is recorded and the relevant data to noise ratio is much higher.

- A possible trend in future work could be the personalization of the system: human motion related activities could depend on the character and the age of the subject. Also, the subject's habits could be included in the system for improving the inference. For example, if a subject loves sports and goes *cycling* in summer while *running* in winter, activity inference could be done based on the season of the year and on the subject's preferences. Another example is an old lady that, instead of driving or *going into a vehicle* usually goes *walking* everywhere. This information could be added into the system in terms of probability to provide a better inference of the subject's activity. Another example: if the subject is an old lady, most probable she will not jump and, if she does, it will be during a short-time. Consequently, the length of the sliding window to obtain the features could be decreased. The final device designed for activity recognition could include profiles where the relevant and useful information of the subjects is recorded and the subject selects his/her profile every time he/she wants to use the system.

Bibliography

[1] X. T. B.V., *MTi and MTx User Manual and Technical Documentation*, 2007. [Online]. Available: http://www.xsens.com/

[2] O. J. Woodman, "An introduction to inertial navigation," University of Cambridge Computer Laboratory, Tech. Rep., August 2007.

[3] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive 2004*, pp. 1–17, April 2004. [Online]. Available: http://dx.doi.org/10.1007/b96922

[4] C. Randell and H. Muller, "Context awareness by analyzing accelerometer data," in *ISWC '00: Proceedings of the Fourth IEEE International Symposium on Wearable Computers*. Washington, DC, USA: IEEE Computer Society, 2000, p. 175.

[5] S. Pirttikangas, K. Fujinami, and T. Nakajima, "Feature selection and activity recognition from wearable sensors," in *Ubiquitous Computing Systems*. Springer-Verlag, 2006, pp. 516–527.

[6] J.-Y. Yang, Y.-P. Chen, G.-Y. Lee, S.-N. Liou, and J.-S. Wang, "Activity recognition using one triaxial accelerometer: A neuro-fuzzy classifier with feature reduction," in *Entertainment Computing - ICEC 2007*. International Federation for Information Processing, 2007, pp. 395–400.

[7] C. Lombriser, N. B. Bharatula, D. Roggen, and G. Tröster, "On-body activity recognition in a dynamic sensor network," in *BodyNets '07: Proceedings of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (ICST) 2nd international conference on Body area networks*, ICST, Brussels, Belgium, Belgium, 2007, pp. 1–6.

[8] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Tröster, "Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection," in *EWSN*, ser. Lecture Notes in Computer Science, R. Verdone, Ed., vol. 4913. Springer, 2008, pp. 17–33.

[9] N. Bhargava Bharatula and G. Tröster, "On-body context recognition with miniaturized autonomous sensor button," *Technisches Messen*, vol. 74, pp. 621–628, 2007.

[10] T. Stiefmeier, "Real-time spotting of human activities in industrial environments," Ph.D. dissertation, TU Darmstadt, 2008.

[11] D. A. Winter, *Biomechanics and Motor Control of Human Movement*. Wiley, August 2004. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/047144989X

[12] H. Guo and W. Hsu, "A survey of algorithms for real-time bayesian network inference," in *the joint AAAI-02/KDD-02/UAI-02 workshop on Real-Time Decision Support and Diagnosis Systems*, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.5182

[13] L. R. Rabiner and R. W. Schafer, *Digital processing of speech signals*. Prentice-Hall, Englewood Cliffs, 1978.

[14] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-time signal processing (2nd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1999.

[15] B. Ray, "Now google tracking follows you out of cyberspace. making your life easier by knowing what you're doing," http://www.theregister.co.uk/2009/05/21/google_motion_tracking/, 2009.

[16] M. Weiser, "The future of ubiquitous computing on campus," *Commun. ACM*, vol. 41, no. 1, pp. 41–42, 1998.

[17] A. K. Dey, "Understanding and using context," *Personal Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[18] T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, P. P. Klasnja, K. Koscher, A. LaMarca, J. A. Landay, L. LeGrand, J. Lester, A. Rahimi, A. Rea, and D. Wyatt, "The mobile sensing platform: An embedded activity recognition system," *IEEE Pervasive Computing*, vol. 7, no. 2, pp. 32–41, 2008.

[19] D. T. G. Huynh, "Human Activity Recognition with Wearable Sensors," Ph.D. dissertation, Technische Universität Darmstadt, August 2008.

[20] T. Starner, B. Rhodes, J. Weaver, and A. Pentland, "Everyday-use wearable computers," *IBM Systems Journal*, vol. 35, pp. 618–629, 1999.

[21] T. Stiefmeier, C. Lombriser, D. Roggen, H. Junker, G. Ogris, and G. Tröster, "Event-based activity tracking in work environments," in *Proceedings of the 3 rd IFAWC*. VDE Verlag, 2006, pp. 91–100.

[22] D. Roetenberg, "Inertial and magnetic sensing of human motion," Ph.D. dissertation, Universiteit Twente, 2006.

[23] K. Aminian, P. Robert, E. Buchser, B. Rutschmann, D. Hayoz, and M. Depairon, "Physical activity monitoring based on accelerometry: validation and comparison with video observation," *Medical & Biological Engineering & Computing*, vol. 37, no. 3, pp. 304–308, 1999.

[24] M. Makikawa and H. Iizumi, "Development of an ambulatory physical activity memory device and its application for the categorization of actions in daily life," in *Proceedings of 8th conference on medical informatics*. MEDINFO, 1995, pp. 747–750.

[25] S. Kurata, M. Makikawa, M. Kawato, H. Kobayashi, A. Takahashi, and R. Tokue, "Ambulatory physical activity monitoring system," *Proceedings of MEDINFO 98*, pp. 277–281, 1998.

[26] M. Uiterwaal, E. Glerum, H. Busser, and R. van Lummel, "Ambulatory monitoring of physical activity in working situations, a validation study," *Journal of Medical Engineering & Technology*, vol. 22, no. 4, pp. 168–172, July/August 1998.

[27] K. V. Laerhoven and O. Cakmakci, "What shall we teach our pants?" in *ISWC '00: Proceedings of the 4th IEEE International Symposium on*

Wearable Computers. Washington, DC, USA: IEEE Computer Society, 2000, p. 77.

[28] J. Mantyjarvi, J. Himberg, and T. Seppanen, "Recognizing human motion with multiple acceleration sensors," in *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, vol. 2, 2001, pp. 747–752.

[29] S.-W. Lee and K. Mase, "Activity and location recognition using wearable sensors," *Pervasive Computing, IEEE*, vol. 1, no. 3, pp. 24–32, 2002. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1037719

[30] M. Losch, S. Schmidt-Rohr, S. Knoop, S. Vacek, and R. Dillmann, "Feature set selection and optimal classifier for human activity recognition," in *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, Aug. 2007, pp. 1022–1027.

[31] G. S. Chambers, S. Venkatesh, G. A. W. West, and H. H. Bui, "Hierarchical recognition of intentional human gestures for sports video annotation," *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 2, pp. 1082–1085, 2002.

[32] K. Van Laerhoven and H.-W. Gellersen, "Spine versus porcupine: A study in distributed wearable activity recognition," in *ISWC '04: Proceedgins of the Eighth International Symposium on Wearable Computers*. IEEE Computer Society, 2004, pp. 142–149.

[33] L. E. Dunne, P. Walsh, B. Smyth, and B. Caulfield, "Design and evaluation of a wearable optical sensor for monitoring seated spinal posture," in *ISWC '06: Proceedings of the Tenth IEEE International Symposium on Wearable Computers*. IEEE Press, 2006, pp. 65–68.

[34] S. Bracy, L. E. Dunne, R. Tynan, D. Diamond, B. Smyth, and G. M. P. O'Hare, "Garment-based monitoring of respiration rate using a foam pressure sensor," in *ISWC '05: Proceedings of the Ninth IEEE International Symposium on Wearable Computers*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 214–215.

[35] N. Oliver and F. Flores-Mangas, "Healthgear: A real-time wearable system for monitoring and analyzing physiological signals," *Wearable*

and Implantable Body Sensor Networks, International Workshop on, pp. 61–64, 2006.

[36] T. L. Westeyn, P. Presti, and T. Starner, "Actiongsr: A combination galvanic skin response-accelerometer for physiological measurements in active environments." in *ISWC '06: Proceedings of the Tenth IEEE International Symposium on Wearable Computers*. IEEE Press, 2006, pp. 129–130.

[37] J. Farringdon, A. J. Moore, N. Tilbury, J. Church, and P. D. Biemond, "Wearable sensor badge and sensor jacket for context awareness," in *ISWC '99: Proceedings of the Third IEEE International Symposium on Wearable Computers*. Washington, DC, USA: IEEE Computer Society, 1999, p. 107.

[38] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," *Pervasive Computing*, pp. 158–175, 2004.

[39] G. Bieber and C. Peter, "Using physical activity for user behavior analysis," in *PETRA '08: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*. New York, NY, USA: ACM, 2008, pp. 1–6.

[40] S. S. Intille, L. Bao, E. M. Tapia, and J. Rondoni, "Acquiring in situ training data for context-aware ubiquitous computing applications," in *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2004, pp. 1–8.

[41] J. Lester, T. Choudhury, and G. Borriello, "A practical approach to recognizing physical activities," in *Pervasive Computing*, 2006, pp. 1–16.

[42] N. Kern, B. Schiele, and A. Schmidt, "Multi-sensor activity context detection for wearable computing," in *Proceedings EUSAI, LNCS*, 2003, pp. 220–232.

[43] K. C. Smith, "Bayesian methods for visual multi-object tracking with applications to human activity recognition," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Lausanne , Switzerland, 2007,

thèse sciences Ecole polytechnique fédérale de Lausanne EPFL, no 3745 (2007), Faculté des sciences et techniques de l'ingénieur STI, Section de génie électrique et électronique, Institut de génie électrique et électronique IEL (Laboratoire de l'IDIAP). Dir.: Hervé Bourlard, Daniel Gatica-Perez.

[44] A. Raj, A. Subramanya, D. Fox, and J. A. Bilmes, "Rao-blackwellized particle filters for recognizing activities and spatial context from wearable sensors." in *ISER*, ser. Springer Tracts om Advanced Robotics, vol. 39. Springer, 2006, pp. 211–221.

[45] L. Liao, D. Fox, and H. Kautz, "Extracting places and activities from GPS traces using hierarchical conditional random fields," *Int. J. Rob. Res.*, vol. 26, no. 1, pp. 119–134, 2007.

[46] D. Minnen, T. Starner, I. Essa, and C. Isbell, "Discovering characteristic actions from on-body sensor data," in *ISWC '06: Proceedings of the Tenth IEEE International Symposium on Wearable Computers*. Los Alamitos, AC, USA: IEEE Computer Society, 2006, pp. 11–18.

[47] A. Subramanya and A. Raj, "Recognizing activities and spatial context using wearable sensors," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006.

[48] M. Stikic, K. V. Laerhoven, and B. Schiele, "Exploring semi-supervised and active learning for activity recognition," in *ISWC '08: Proceedings of the Twelfth IEEE International Symposium on Wearable Computers*. IEEE Press, 2008, pp. 14, 19, 107.

[49] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford, "A hybrid discriminative/generative approach for modeling human activities," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 766–772.

[50] L. Bao, "Physical Activity Recognition from Acceleration Data under Semi-Naturalistic Conditions," Master Thesis of Massachusetts Institute of Technology, August 2003.

[51] E. K. Antonsson and R. W. Mann, "The frequency content of gait," *J. Biomechanics*, vol. 18, no. 1, pp. 39–47, 1985.

[52] C. Angeloni, P. O. Riley, and D. E. Krebs, "Frequency content of whole body gait kinematic data," *IEEE Transactions on rehabilitation engineering*, vol. 2, no. 1, 1994.

[53] I. Kingma, H. M. Toussaint, D. A. Commissaris, M. J. Hoozemans, and M. J. Ober, "Optimizing the determination of the body center of mass." *J. Biomech*, vol. 28, no. 9, pp. 1137–42, 1995. [Online]. Available: http://www.biomedsearch.com/nih/Optimizing-determination-body-center-mass/7559685.html

[54] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. The American Institute of Aeronautics and Astronautics, 2004.

[55] J. Mao, "Optimal orthonormalization of the strapdown matrix by using singular value decomposition," *Computers & Mathematics with Applications*, vol. 12, no. 3, pp. 309–375, 1986.

[56] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[57] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1053964

[58] F. V. Jensen, *Bayesian Networks and Decision Graphs*, ser. Information Science and Statistics. Springer, July 2001. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387952594

[59] D. Heckerman, "A tutorial on learning with bayesian networks," Learning in Graphical Models, Tech. Rep., 1995.

[60] G. F. Cooper and E. Herskovits, "A bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 09, no. 4, pp. 309–347, October 1992. [Online]. Available: http://www.ingentaconnect.com/content/klu/mach/1992/00000009/00000004/00422779

[61] K. Frank, M. Röckl, M. J. Vera Nadales, P. Robertson, and T. Pfeifer, "Comparison of exact static and dynamic bayesian context inference

methods for activity recognition," in *Proceedings of Workshop for Managing Ubiquitous Communications and Services 2010*, 2010, pp. 41–47.

[62] C. Huang and A. Darwiche, "Inference in Belief Networks: A Procedural Guide," *International Journal of Approximate Reasoning*, vol. 15, no. 3, pp. 225–263, 1996.

[63] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley & Sons, August 2006.

[64] S. Arulampalam, S. Maskell, and N. Gordon, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.3641

[65] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[66] X. T. B.V., *MTi and MTx Quick Setup*, 2007. [Online]. Available: http://www.xsens.com/

[67] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Computer Graphics and Applications*, vol. 25, no. 6, pp. 38–46, 2005.

[68] V. Albiol, A.; Naranjo and J. Prades, *Tratamiento Digital de la Señal*. Universidad Politécnica de Valencia, 1999.

[69] N. I. D. Z., "Windowing: Optimizing FFTs using window functions," http://zone.ni.com/devzone/cda/tut/p/id/4844, 2008.

[70] ——, "The fundamentals of FFT-Based signal analysis and measurement in labview and labwindows/cvi," http://zone.ni.com/devzone/cda/tut/p/id/4278#toc2, 2009.

[71] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, June 2005. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1455106

[72] R. N. Moll, M. A. Arbib, and A. J. Kfoury, *An introduction to formal language theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1988.

Appendix A

Related Work

Some details of related work on activity recognition are shown in the tables below.

Bao, [20]	**Description**
Activities	*Walking, walking while carrying items, sitting and relaxing,working on computer, standing still, eating or drinking, watching TV,reading, running, bicycling, stretching, strength-training, scrubbing, vacuuming, folding laundry, lying down and relaxing, brushing teeth, climbing stairs, riding elevator, riding escalator.*
Sensors	Accelerometers.
Sensor Location	Left thigh, right ankle, left arm, right wrist, right hip.
Features	mean value, energy, frequency-domain entropy, correlation between two axes and between all pairwise.
Sample Frequency	76.25 Hz.
Windowing	512 samples with 50% overlap.
Data set	20 people under semi-naturalistic conditions.
Classifier	Nearest Neighbor, Naive Bayes & Decision Tree.
Accuracy	Best result: 84.26%.

Table A.1: Description of reference [20].

Randell, [10]	**Description**
Activities	*Walking, walking upstairs, walking downstairs, sitting, running, standing.*
Sensors	Accelerometers 2D, GPS.
Sensor Location	Pocket.
Features	RMS for each axis & integrated value of each axis over the last 2 seconds.
Sample Frequency	5 Hz.
Data set	10 people under laboratory conditions.
Classifier	Neural network with backpropagation.
Accuracy	85% - 90%

Table A.2: Description of reference [10].

Pirttikangas, [14]	**Description**
Activities	*Clean white board, read a newspaper, stand still, sit and relax, sit and watch TV, drink, brush teeth, lie down, vacuum,clean, type, walk, climb stairs, descend stairs, elevator up, elevator down, run, cycle.*
Sensors	3D Accelerometers, compass, ambient light, force sensor, heart rate.
Sensor Location	Right thigh, right wrist, left wrist, necklace.
Features	Mean value, standard deviation, mean crossings, correlation coefficient between axes.
Sample Frequency	10 Hz.
Windowing	7 samples.
Data set	13 people under semi-naturalistic conditions.
Classifier	Multilayer perceptrons, kNN.
Accuracy	Best result: 90.61%.

Table A.3: Description of reference [14].

Jhun-Ying, [21]	**Description**
Activities	*Walking, running, scrubbing, standing, working on computer, vacuuming, brushing teeth, sitting.*
Sensors	3D Accelerometers.
Sensor Location	Dominant wrist.
Features	mean, interquartile range, mean absolute deviation, correlation between the axes, root mean square, standard deviation, variance, energy. All computed for each axis.
Windowing	512 samples with 50% overlap.
Data set	7 people under laboratory conditions.
Classifier	LDA & FSS.
Accuracy	Best result: 92.86 +/- 5.91%.

Table A.4: Description of reference [21].

Lombriser, [25]	**Description**
Activities	*Drinking water, moving a computer mouse, writing on a white board, opening a drawer, opening a cupboard, typing on a keyboard, writing with a pen.*
Sensors	Accelerometer, light sensor, microphone.
Sensor Location	Shoulder, abdomen, back, ankle, arm, chest, wrist.
Features	mean, energy, variance, standard deviation, fluctuation, mean gradient, absolute gradient, mean crossing rate.
Sample Frequency	32 Hz.
Windowing	80 samples with 70% overlap.
Data set	9 people under laboratory conditions.
Classifier	J48/C4.5, kNN, Naive Bayes & Bayesian network.
Accuracy	Best result: 98%.

Table A.5: Description of reference [25].

Lombriser, [32]	**Description**
Activities	*Write on notepad, open hood, close hood, check gaps on the front door, open left front door, close left, front door, close both left door, check trunk gaps, open and close trunk, check steering wheel.*
Sensors	19 Accelerometers.
Sensor Location	All over the arms.
Features	sign of the acceleration magnitude (positive, negative or null).
Data set	1 person under laboratory conditions.
Classifier	Meta-classifier with Four states HMM, Naive Bayes and dynamic sensor selection.
Accuracy	98% using all the accelerometers and 80% on average using one accelerometer.

Table A.6: Description of reference [32].

Bhargava, [22]	**Description**
Activities	*Open/close a drawer, drinking water from a cup, typing on a computer keyboard, moving the computer mouse,open/close a cupboard, writing with a pen, writing on a white board.*
Sensors	3D Accelerometer, light sensor, microphone.
Sensor Location	Wrist, chest, shoes.
Features	mean, variance, energy, fluctuation.
Sample Frequency	32 Hz.
Windowing	80 samples with 80% overlap.
Data set	7 people under laboratory conditions.
Classifier	J18, Bayesian network, Naive Bayes & kNN.
Accuracy	Between 85 and 91% recognition online.

Table A.7: Description of reference [22].

Stiefmeier, [8]	**Description**
Sensors	7 IMU.
Sensor Location	Torso, both hands and in both upper, lower arms.
Features	A total of 35 features: Torso: mean, variance, minimum & maximum of the pitch angle, variance of the roll angle, variance and maximum of the gyroscope of the vertical axis. Right & left upper & lower arm: mean & variance of the pitch angle, start and stop angle of pitch. Right & left hand: mean & variance of the acceleration over three axes, mean and variance of the roll angle, start and stop angle of roll in a given segment.
Feature Selection	35 Naive Bayes for feature selection. Given a threshold, if the result of the classifier is above it, then it will be selected for the classification of the given activity.
Classifier	String matching.
Test 1	Repairing a bike.
Activities	*pumping, turning pedals, mark unbalances, dismount back light, assemble back light.*
Sample Frequency	50 Hz.
Data set	3 people under laboratory conditions.
Accuracy	82.7%.
Test 2	Car production.
Activities	*open hood, close hood, open trunk, check trunk, close trunk, fuel lid, open left door, close left door, open right door, close right door, open two doors, close two doors, mirror, check trunk gaps,lock check left, lock check right, check hood gaps, open spare wheel box, close spare wheel box, writing.*
Data set	8 people under laboratory conditions.
Results	Precision, recall and f-measure are provided for every class (see reference for further details).

Table A.8: Description of reference [8].

Appendix B

Human Motion

This Appendix shows signals recorded during the performance of the activities. Video data was not recorded during this Master Thesis, so the analysis and hypotheses about the nature of the signals and physics behind them cannot be corroborated. Hopefully, this Appendix could be useful for future work on the recognition of human motion related activities as well as Pedestrian Dead Reckoning (PDR).

B.1 Static Activities

Figure B.1 shows acceleration signals during the performance of several activities including *standing* and *sitting*. Several characteristic can be observed from the signals represented:

- $|a|$ does not provide any information for the distinction between *static* activities apart from the fact of measuring just the gravity value (see Figure B.1(a)).
- The body frame (BF) contains useful information due to the attitude of the sensor while the global frame (GF) loses this information.
- On the one hand, it can be observed that *standing* is characterized by a $a_{vert_{BF}} \approx g\ m/s^2$ where g is the local gravity value and $|a_{horiz_{BF}}| \approx 0\ m/s^2$. On the other hand, *sitting* changes the attitude of the sensor and the information of how gravity splits into these signals is the clue for distinguishing between these two activities apart from the transitions

between them. Note that depending on the clothing and the position of the sensor, it will be easier to distinguish between *standing* and *sitting*, but as soon as the attitude of the sensor while *sitting* is equal to *standing*, there is not any relevant or reliable information to distinguish them apart of past activities.

With respect to *lying*, the same conclusions can be extracted from Figure B.8. In this case, the acceleration of the gravitational field of the Earth is measured in the horizontal plane of human body while the vertical acceleration is around zero. This activity can be distinguished from the rest due to this fact.

B.2 Walking

Walking acceleration signals are not easy explainable as the movement of the human body while *walking* is complex. However, interesting characteristics of the signals can be observed for future applications for PDR or activity recognition.

Figure B.2 (a) shows $|a|$ and Figure B.2 (b) the absolute value of its DFT coefficients. $|a|$ is almost symmetrical except for the peaks of the acceleration that the sensor feels when the heel and the tips of the feet hit the floor while accelerating forward. Main frequency component for this subject is at 2.02 Hz and corresponds to the periodicity of every step. A little component in frequency is noticed at 1 Hz due to the periodicity of the pattern every two steps. As the sensor is placed on the right or left part of the body, the acceleration felt when the leg that carries the sensor hits the floor respect to the leg that does not carry it is lightly different and this frequency component appears. During *walking*, the GF and BF contain almost the same information as can be observed in Figure B.2 because the human body is standing straight in flat terrain.

More details about steps during *walking* are given in Figure B.3:

- The first perception in the signal is the number of steps. A total of five steps can be observed and they are marked in time in Figure B.3. They are characterized by a minimum of $|a_{horiz_{BF}}|$ and $a_{vert_{BF}}$. This information can be used for PDR.

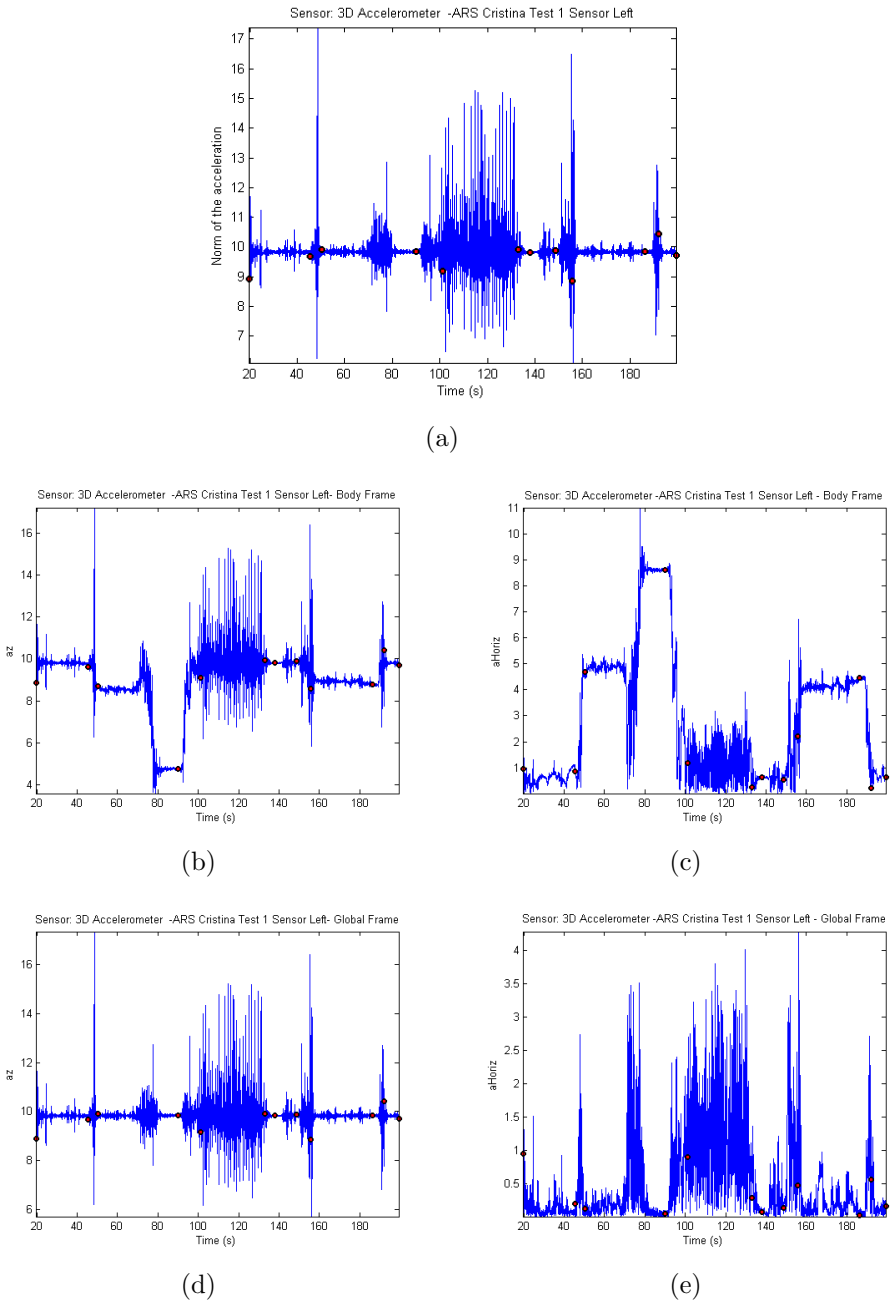


Figure B.1: Standing and sitting analysis. Signals (a) $|a|$ (b) $a_{vert_{BF}}$ (c) $|a_{horiz_{BF}}|$ (d) $a_{vert_{GF}}$ and (e) $|a_{horiz_{GF}}|$ are plotted during *standing - getting down - sitting - getting up - walking - standing - getting down - sitting - getting up - standing*. The red dot markers show the beginning of these activities.

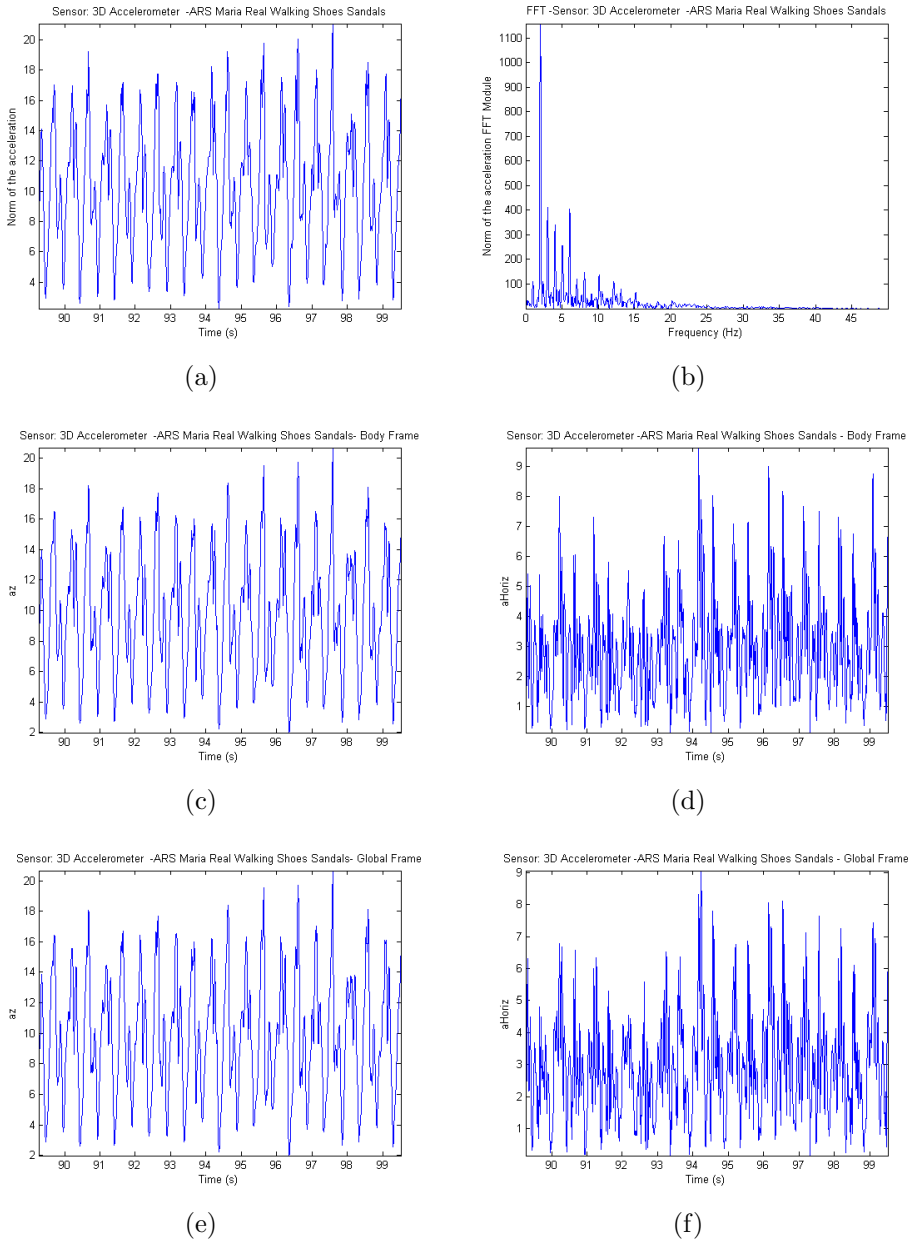


Figure B.2: *Walking* analysis. Signals (a) $|a|$ (b) Absolute value of the discrete Fourier transform coefficients of $|a|$ (c) $a_{vert_{BF}}$ (d) $|a_{horiz_{BF}}|$ (e) $a_{vert_{GF}}$ and (f) $|a_{horiz_{GF}}|$ are plotted during *walking*. Both frames contain approximately the same information as the GF and the BF are the same as soon as the human body attitude is *standing*.

- The sensor is placed on the belt either on the right part of the body or on the left part of it. When the foot hits the floor, a drastic change in the velocity of its movement occurs in a short time. It is the reason why acceleration peaks can be noticed in $a_{vert_{BF}}$. Figure B.3 (a) marks these acceleration peaks when the heel hits the floor (pointed out as $*$) and when the tips of the foot touch the ground ($**$). The acceleration value of the moment where the person touches the ground can vary depending on the kind of shoes or the ground. The steps shown in Figure B.3 were recorded while the subject was carrying sandals in outdoor environment. During this work, *walking* and *running* data were collected in indoor and outdoor environment including walks and runs in the countryside and using different shoes (boots, sandals and sport shoes mainly).

- From these acceleration signals, some hypotheses about the step with the leg corresponding with the part of the body where the sensor is placed can be extracted. It can be observed that the pattern *heel-tip* is very clear in the second and fourth step where $|a_{horiz_{BF}}|$ presents a maximum. The pattern *heel-tip* performed by the step given with the leg where the sensor is not placed on will be noticed as well in the signal but this acceleration should be transmitted through the joints of the leg and the hip conforming a different pattern in the signal. Finally, the most suitable hypothesis is that the sensor has been placed in the part of the body whose foot has touched the ground during the second and fourth steps.

- $a_{vert_{BF}}$ will provide information about the *up-down* movement of the hip while *walking*. This movement could depend on how far the sensor is placed from the human body center of mass (CM) during *standing* position. Also, it will depend on the trousers used as well as the gender of the subject. This movement of the hip while *walking* could be sensed significantly if the sensor is placed near the pocket of subject's trouser and the position of the belt in this trouser is low.

- With respect to $|a_{horiz_{BF}}|$, the maximum value has been analyzed before. However, minimum values can be observed and are marked in time in Figure B.3 (b). These minimum values occur just before the pattern *heel-tip* of the leg from the part of the body where the sensor

is not placed (marked in such figure with *) and right after the pattern *heel-tip* of the leg where the sensor is placed. $|a_{horiz_{BF}}|$ takes a noticeable value between these two minimums that could correspond to the acceleration produced in the body when the other leg (where the sensor is not) is moving forward. Finally, after the pattern *heel-tip* of this leg there is a minimum that could be caused by this leg being straight and the other one starting to move.

B.3 Running

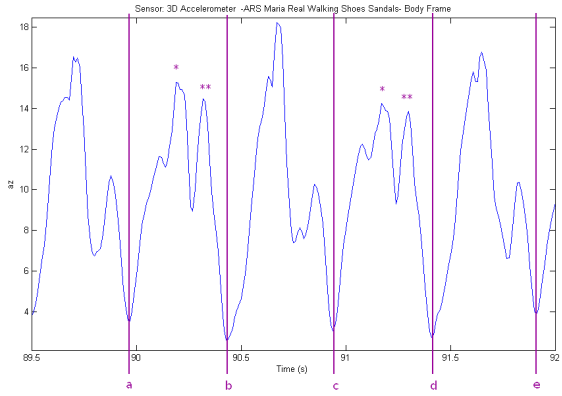
Signals from *running* are shown in Figure B.4 and Figure B.5. Some conclussions can be extracted from the observation of these signals:

- Peaks of $|a|$ are bigger due to the higher value of the horizontal acceleration and the drastic change in velocity that occurs while hitting the floor with the feet. In this case, this signal is not symmetrical and the mean value of it is a meaningful feature as shown in Figure B.4 (a).
- Its spectrogram shows how the main frequency component remains almost constant (see Figure B.4 (b)).
- Main frequency component observed in Figure B.4 (c) is at 2.83 Hz, bigger than for *walking* as the duration of one step is lower for *running* than for *walking*.
- The GF and the BF contain almost the same information as when the person is *standing* in flat terrain.

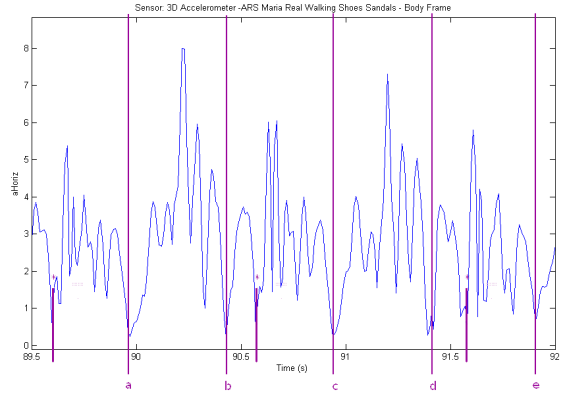
Details of $a_{vert_{BF}}$ and $|a_{horiz_{BF}}|$ during 8 steps are provided in Figure B.6:

- Steps can be differentiated thanks to the peaks in the acceleration. A total of 8 steps are noticed in both signals. Steps can be counted for PDR.
- High values of $|a_{horiz_{BF}}|$ are shown in every step while the feet touches the ground as the person takes impulse to go on running increasing its horizontal acceleration.
- Pattern *heel-tip* can be observed as well in some steps, but not always.

(a)

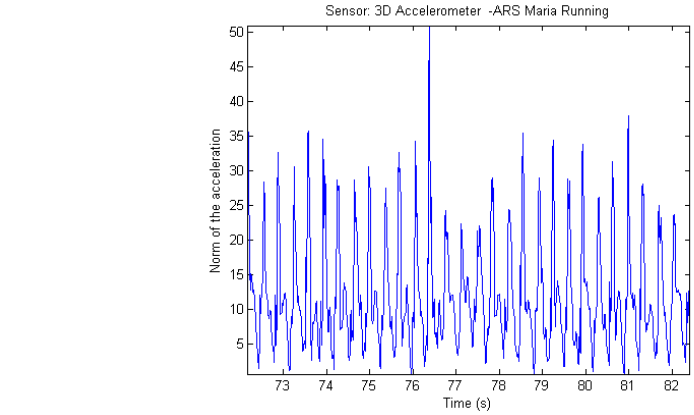


(b)

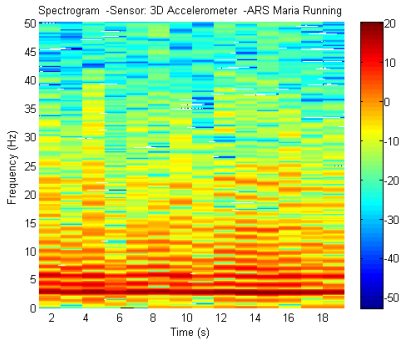


(c)

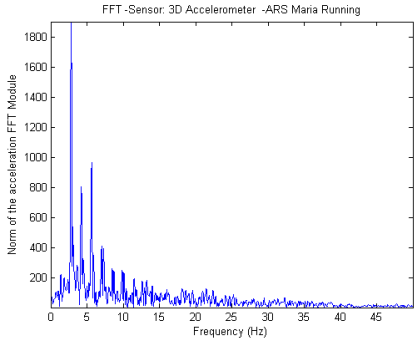
Figure B.3: *Walking* step analysis (a) A total of 5 steps is represented. Blue foot represents the leg that carries the sensor. If the foot is straight, it is the static leg. However, if the foot has some inclination, it is moving. To corroborate this analysis, video data is recommended. (b) $a_{vert_{BF}}$ and (c) $|a_{horiz_{BF}}|$ during these five steps. Significant features of the signals are shown.



(a)



(b)



(c)

Figure B.4: *Running* analysis (a) Peaks of $|a|$ are bigger due to the higher value of the horizontal acceleration and the drastic change in velocity that occurs while hitting the floor with the feet. (b) Spectrogram of $|a|$ where the main frequency component remains almost constant (c) The absolute value of the DFT coefficients of $|a|$. Main frequency component is at 2.83 Hz and corresponds to the periodicity of every step.

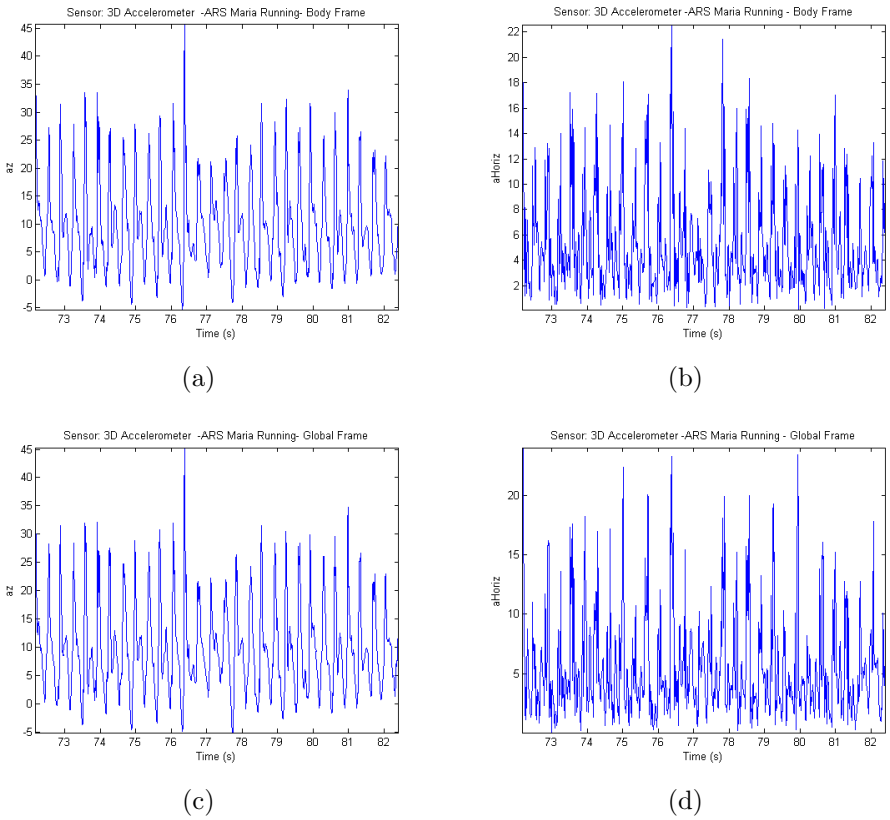
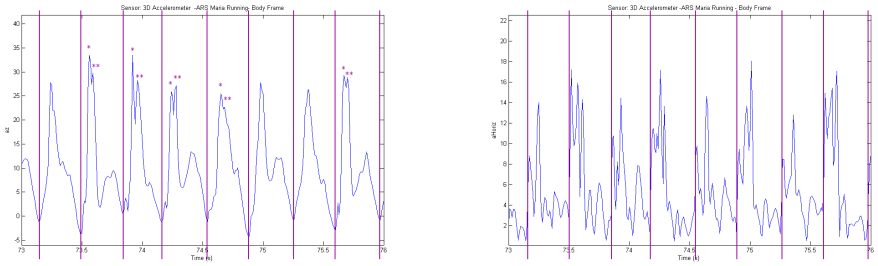


Figure B.5: Running analysis. Signals (a) $a_{vert_{BF}}$ (b) $|a_{horiz_{BF}}|$ (c) $a_{vert_{GF}}$ (d) $|a_{horiz_{GF}}|$ during *running*. Both frames contain the same information.

- Second, fourth, sixth and eighth steps could have been taken with the leg that does not carry the sensor as the *up-down* movement of the hip can be noticed in the pattern increasing and decreasing $a_{vert_{BF}}$ after hitting the floor.



(a) (b)

Figure B.6: *Running* step analysis (a) $a_{vert_{BF}}$ and (b) $|a_{horiz_{BF}}|$ are plotted. A total of 8 steps are identified. The maximum value of $|a_{horiz_{BF}}|$ occurs while or right after hitting the floor.

B.4 Jumping

Figure B.7 shows relevant information about *jumping*:

- $|a|$ plotted in Figure B.7 (a) follows the pattern of $a_{vert_{BF}}$ as the most significant acceleration information during this activity is contained in the vertial axis of the body.
- The BF and GF contain almost the same information except for the horizontal acceleration that is different.
- With respect to the characterization of *jumping*, interesting information related to physical facts can be gained dividing this activity in several phases:
    1. *Taking impulse* where the subject flexes the legs to prepare to jump. This phase is optional as not always *jumping* is performed in this way and the value of $a_{vert_{BF}}$ decreases.

2. *Flying up* that implies the compression of human body and the increment of $a_{vert_{BF}}$ until a point where the body starts to fall. At this point, the decompression of the body could produce a strong force when the next phase is starting. At this point of time, a high value of $|a_{horiz_{BF}}|$ is also observed but it is only characteristic of *jumping forward*. If the person jumps vertically, $|a_{horiz_{BF}}|$ will not behave equally. For this reason, *jumping* is recognized using vertical acceleration.
3. *Free falling* where the body is not connected to the ground so the accelerometer is involved in a *free fall*. This physical fact is reflected properly in $a_{vert_{GF}}$ whose acceleration value goes to zero during several milliseconds. In this case, $a_{vert_{BF}}$ also shows this behavior as both frames while *jumping* can be approximated as well.
4. *Hitting the floor* where a peak in the acceleration is felt. It can be observed as well, how the human body compresses lightly, as soon as the floor is hit, decreasing the value of $a_{vert_{BF}}$, and decompresses after achieving its stationary value of gravity after a small oscillation of $a_{vert_{BF}}$.

B.5 Falling

Finally, *falling* is the last activity considered in this Master Thesis and acceleration signals during this activity are shown in Figure B.8. The main characteristics of *falling* are described below:

- It implies a high rotation in the horizontal plane, a high value of $|w_{horiz_{BF}}|$ as well as a drastic change in body attitude.
- Both, $a_{vert_{BF}}$ and $|a_{horiz_{BF}}|$ present high values that can be observed clearly in Figure B.8.
- *Free falling* cannot be observed in this activity, as the subject has contact with the floor when it is starting to fall, and the phase, where the subject is flying in the air, does not have a significant duration to observe this phenomenon.

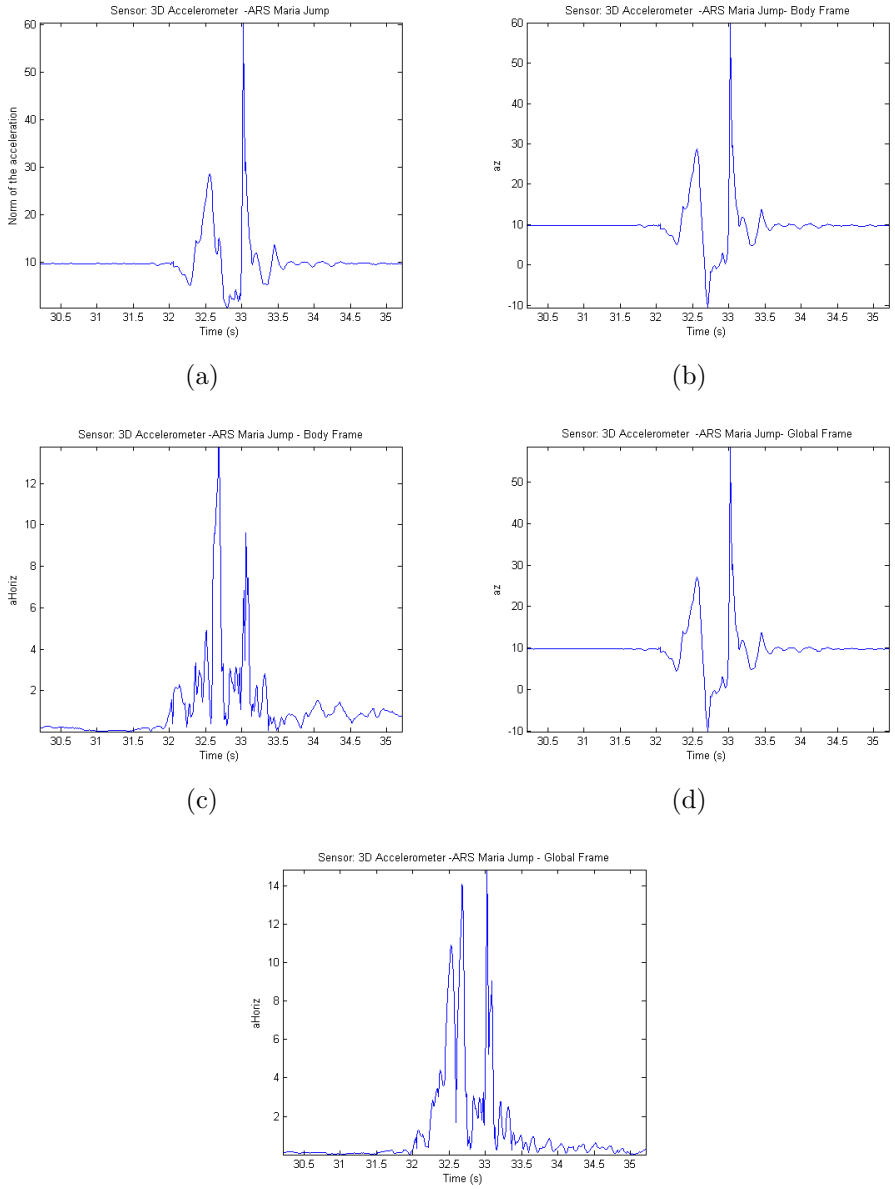


Figure B.7: *Jumping* analysis. Signals (a) $|a|$ (b) $a_{vert_{BF}}$ (c) $|a_{horiz_{BF}}|$ (d) $a_{vert_{GF}}$ and (e) $|a_{horiz_{GF}}|$ during the sequence *standing - jumping - standing.*

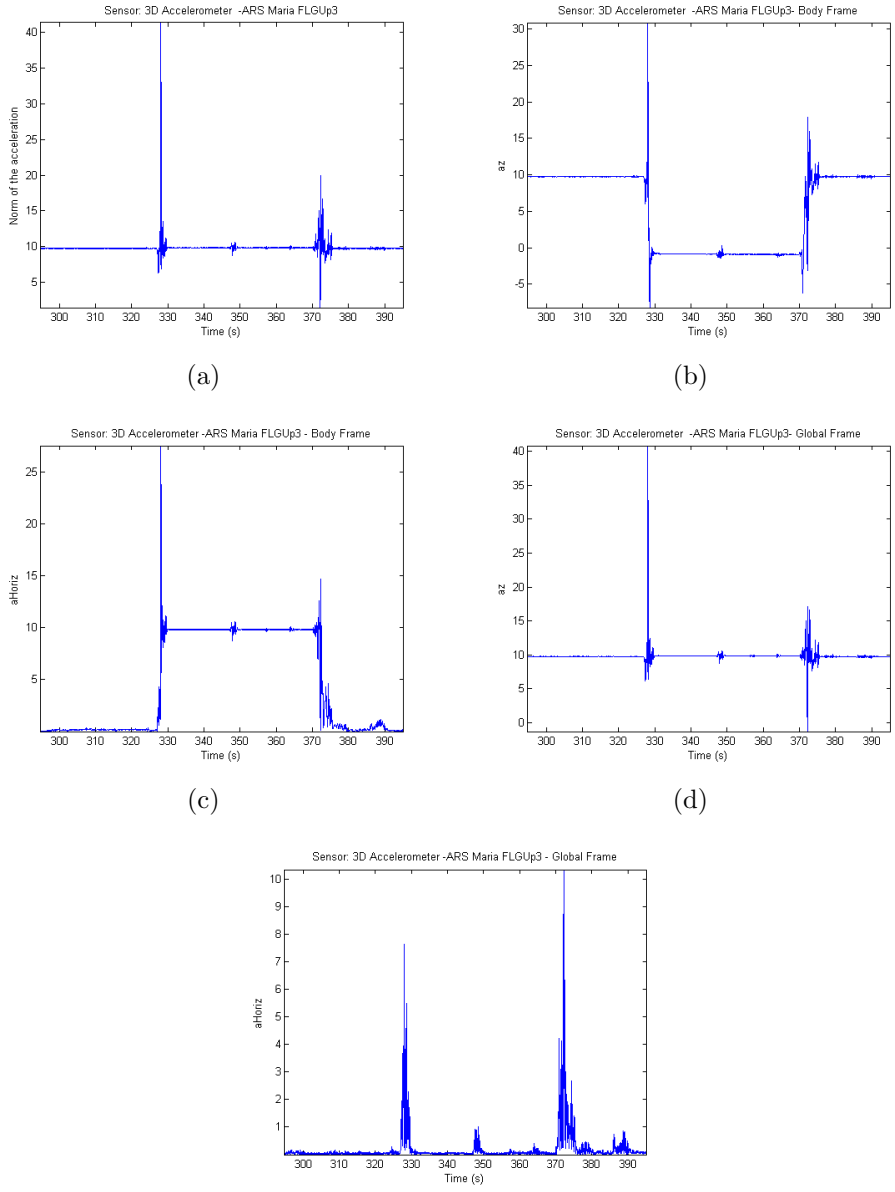


Figure B.8: *Falling* and *lying* analysis. Signals (a) $|a|$ (b) $a_{vert_{BF}}$ (c) $|a_{horiz_{BF}}|$ (d) $a_{vert_{GF}}$ and (e) $|a_{horiz_{GF}}|$ during a sequence of *standing - falling - lying - getting up - standing.*

Appendix C

Data Set Structure

This is the explanation of the contents of the MATLAB variable $ARS\_DLR\_DataSet$[1]:

1. The name of every cell contained in this MATLAB file is composed by:

   $ARS\_[name]\_[otherinformation]$

   where,

   ARS corresponds to *Activity Recognition System*.
   $[name]$ corresponds to the person who performed the test.
   $[otherinformation]$ could be the name of the test, the sensor position, the activity... Any kind of information.

2. Every cell named as before has four positions:

   - $ARS\_Name\_Otherinformation\{1\}$ contains a matrix of double data with the measurements from the accelerometers, gyroscopes and magnetometers of the IMU.
   - $ARS\_Name\_Otherinformation\{2\}$ contains a matrix of double data with the direction cosine matrix extracted from the sensor.
   - $ARS\_Name\_Otherinformation\{3\}$ is a cell array where the identifiers of the activities realized in that log are included.
   - $ARS\_Name\_Otherinformation\{4\}$ is a vector with indices for the matrix of double with the measurements.

[1] MATLAB variables $ARS\_DLR\_DataSet\_v2$ and $ARS\_DLR\_Benchmark\_Data\_Set$ have the same data structure.

3. $ARS\_Name\_Otherinformation\{1\}$ contains a matrix with 10 columns and a different number of rows depending on the test length:

   - 1st column: the time extracted from the sensor in seconds. For example: a value 15.6 means that 15.6 seconds have passed since the sensor started to transmit data.
   - 2nd column: the acceleration in the X axis measured by the sensor.
   - 3rd column: the acceleration in the Y axis measured by the sensor.
   - 4th column: the acceleration in the Z axis measured by the sensor.
   - 5th column: the angular velocity in the X axis measured by the sensor.
   - 6th column: the angular velocity in the Y axis measured by the sensor.
   - 7th column: the angular velocity in the Z axis measured by the sensor.
   - 8th column: the magnetic field in the X axis measured by the sensor.
   - 9th column: the magnetic field in the Y axis measured by the sensor.
   - 10th column: the magnetic field in the Z axis measured by the sensor.

4. $ARS\_Name\_Otherinformation\{2\}$: It contains a matrix with 10 columns and a different number of rows depending on the test length. The first column is the time extracted from the sensor in seconds, too. It should be the same time vector as in $ARS\_Name\_Otherinformation\{1\}$.

5. $ARS\_Name\_Otherinformation\{3\}$: It contains a cell array where every position is a string. Every string is the identifier of the activity. The possible values of these strings are:

```
'RUNNING' = "running"
'WALKING' = "walking"
'JUMPING' = "jumping"
'JUMPVRT' = "jumping vertically"
```

```
'JUMPFWD' = "jumping forward"
'JUMPBCK' = "jumping backward"
'STNDING' = "standing"
'SITTING' = "sitting"
'XLYINGX' = "lying"
'FALLING' = "falling"
'TRANSUP' = "getting up"
'TRANSDW' = "going down"
'TRNSACC' = "accelerating"
'TRNSDCC' = "deccelerating"
'TRANSIT' = "other transition or irrelevant information"
```

For example, if the cell contains three strings as: *STNDING RUNNING STNDING* means that the person performed these three activities in this order. In order to know from which row of the matrix of measurements and the attitude matrix he or she performed the activity, $ARS\_Name\_Otherinformation\{4\}$ should be checked.

6. $ARS\_Name\_Otherinformation\{4\}$:

   It is a vector with the index of the rows that indicate the beginning and ending of an activity. The format is the following:

   $[t1_0\ t1_f\ t2_0\ t2_f\ t3_0\ t3_f\ t4_0\ t4_f\ ...tn_0\ tn_f]$

   where,

   ti_0 for $i = 1...n$ contains the beginning of activity i.
   ti_f for $i = 1...n$ contains the end of activity i.

   Notice that the length of this vector is twice the length of the cell containing the strings with the activities.

The following code describes a MATLAB code example to rotate the acceleration from sensor frame to global frame.

```
%EXAMPLE: MATLAB code to rotate the acceleration from the sensor frame
% to the global frame in the beginning of the first activity.

%%%%%%%%%% BEGINNING EXAMPLE %%%%%%%%%%%%%%%%%%
```

```
measurements = ARS_Name_Otherinformation{1};
% Contains the data from the sensor.
attitude = ARS_Name_Otherinformation{2};
activity = ARS_Name_Otherinformation{3};
changesOfActivities = ARS_Name_Otherinformation{4};

% The begining of the first activity is:
index = changesOfActivities(1);

% The measurements of the acceleration are:

ax = measurements(index,2); % It will give the acceleration
% in the X-axis measured by the sensor
ay = measurements(index,3); % It will give the acceleration
% in the Y-axis measured by the sensor
az = measurements(index,4); % It will give the acceleration
% in the Z-axis measured by the sensor

as = [ax ay az]; % Vector of the acceleration measurements that will be
% multiplied by the direction cosine matrix

% And the direction cosine matrix:

Cb = [attitude(index,2) attitude(index,5) attitude(index,8);
 attitude(index,3) attitude(index,6) attitude(index,9);
  attitude(index,4) attitude(index,7) attitude(index,10)];
  % it is the direction cosine matrix for the index row

% So finally:

an = Cb*as'; % The vector has been rotated to the Global Frame

% And this acceleration was for the 1st activity whose identifier is in:
activity_name = activity{1};

%%%%%%%%%% END EXAMPLE %%%%%%%%%%%%%%%%%%
```

Appendix D

Final Set Examples

In this appendix, more examples of different activities and features are shown in order to provide the reader with a clearer picture of the distinction between the activities considered in this work. This appendix will complement the information about the classification of the activities using the final set of features that was provided in Section 4.8. It will show how to distinguish all the activities giving more examples in the distinction between *static* activities and between *static* and *walking*, which cannot be seen from past examples. The appendix is organized as follows:

- Firstly, results for differentiating *static* activities (*standing, sitting, lying*) are shown in Figures D.1, D.2 and D.3.
- Examples for the distinction of *walking* and *static* activities will be included in Figures D.4 and D.5.
- Figure D.6 is related to the distinction of *running* and *walking*. There are clear examples in order to distinguish between *running* and *static* activities in Section 4.8, so in this appendix this case is not covered widely.
- Next, differentiation between *walking, running, jumping* and *falling* is given in Figures D.7,D.8,D.9, D.10 and D.11.
- Finally, Figures from D.12 to D.15 are included in order to show the behavior of the transitions *up* and *down* in contrast to the activities through the 19 features selected for the final set.

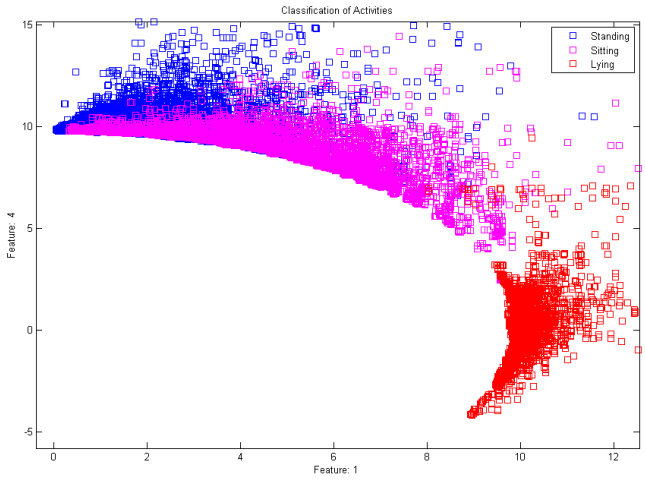


Figure D.1: Example of distinction between *static* activities. This distinction is made through acceleration signals of the body frame and considering the attitude of the sensor.

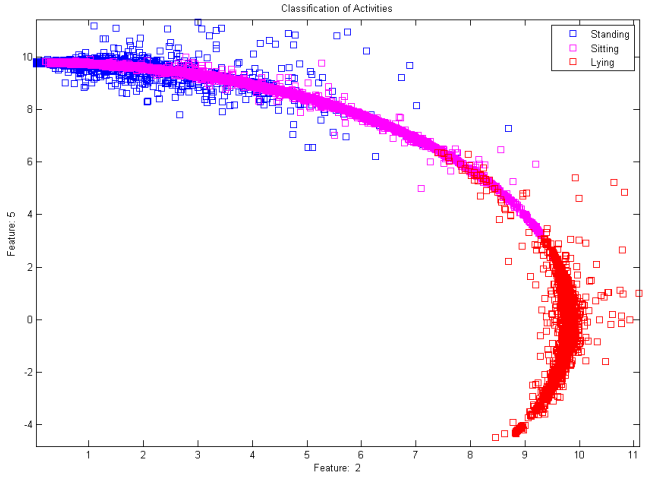


Figure D.2: Distinction of different *static* activities: *standing*, *sitting* and *lying*. The area in common between *standing* and *sitting* could be confusing for the system and corresponds when the attitude of the sensor is sited in such a way that the sensor attitude is similar to the sensor attitude during *standing*.

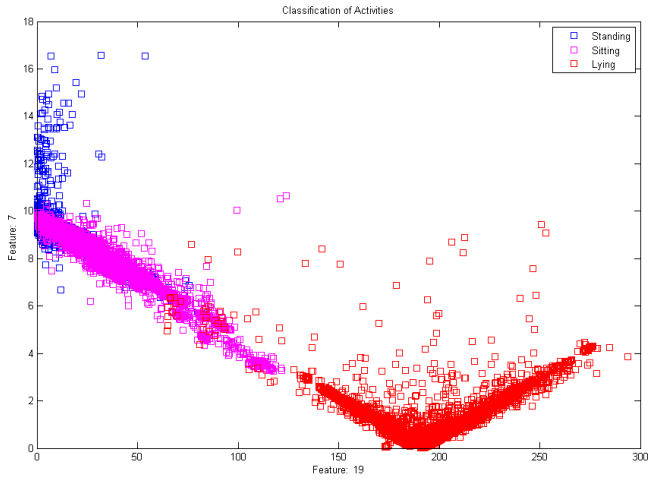


Figure D.3: Distinction of *lying* from the rest of activities: the attitude of the sensor and acceleration signals of the body frame give good information.

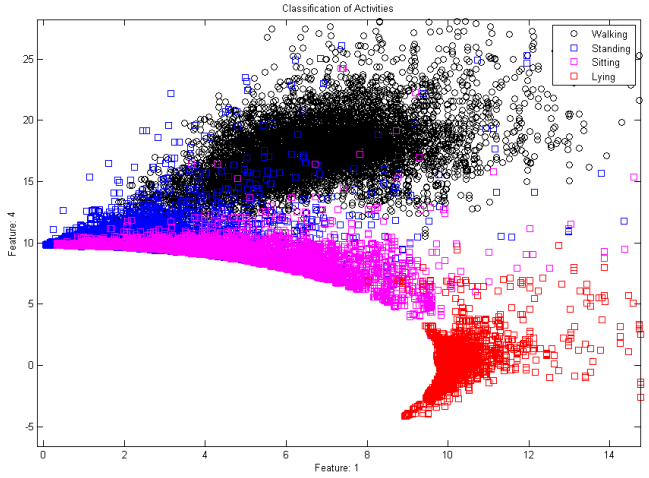


Figure D.4: Distinction between *static* activities and *walking*. In this figure, a part of the *static* activity samples go to the same range as *walking* due to the sliding window that processes the signal and takes samples of other activities. They can be considered in statistical terms as *outliers*.

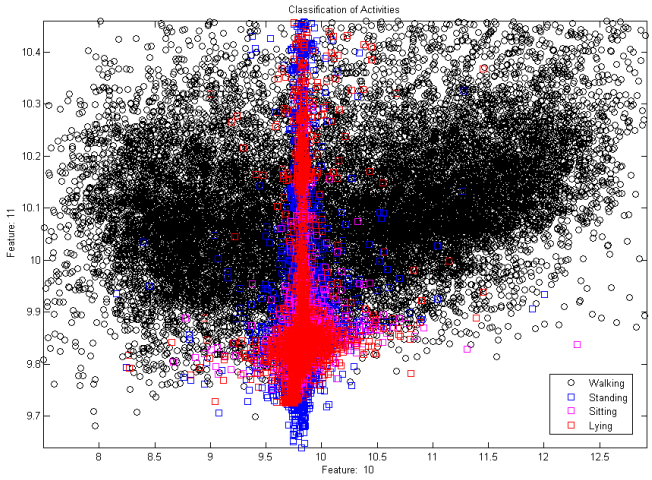


Figure D.5: Distinction between *static* activities and *walking*. *Static* activity samples are around the local gravity value in both features. However, specially for Feature 11 that is computed over a window of 512 samples, the effect of the sliding window produces *static* activity samples considered as *outliers*.

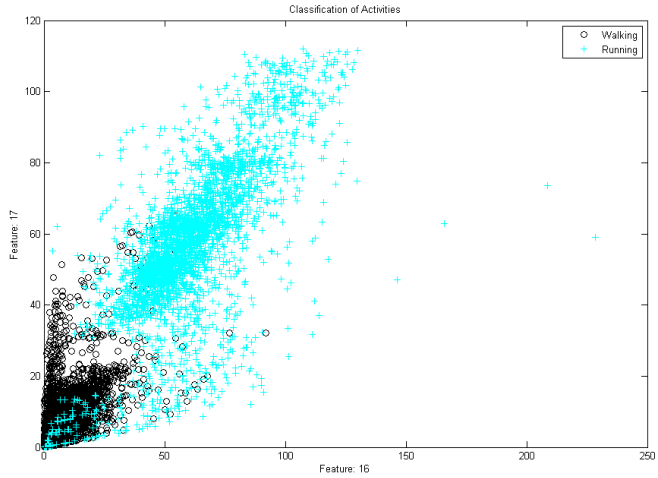


Figure D.6: *Walking* and *running* samples that are contained in the data set are plotted. There is a clear boundary between them combining the information of both features.

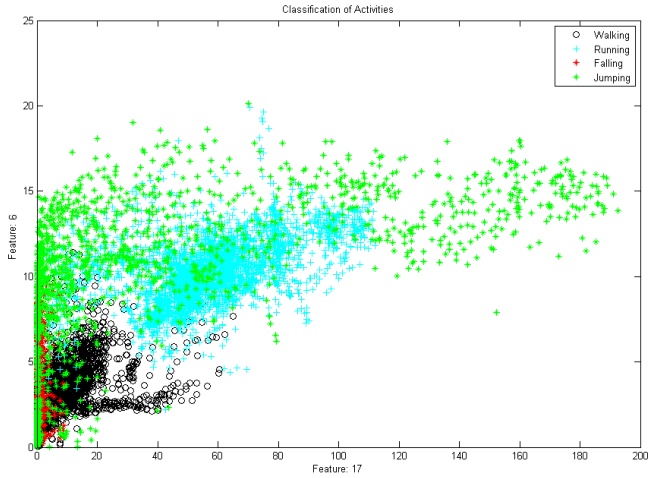


Figure D.7: Distinction of *walking* from *jumping* and *running*. With respect to *falling*, Section 4.8 shows examples where *walking* can be distinguished from *falling* thanks to the change of the attitude of the body.

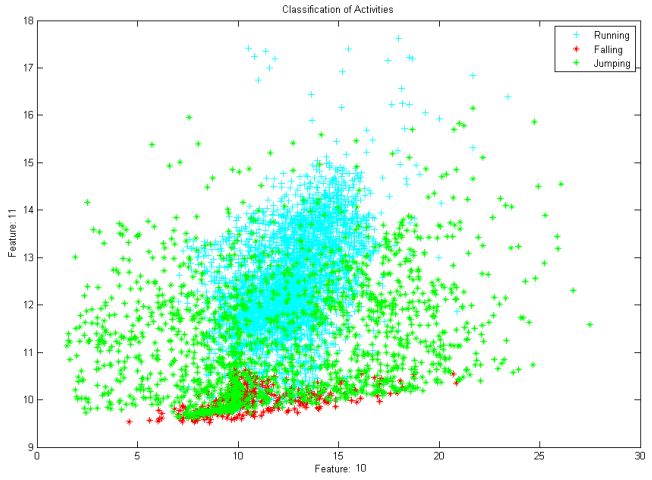


Figure D.8: Distinction between *jumping* and *running*.

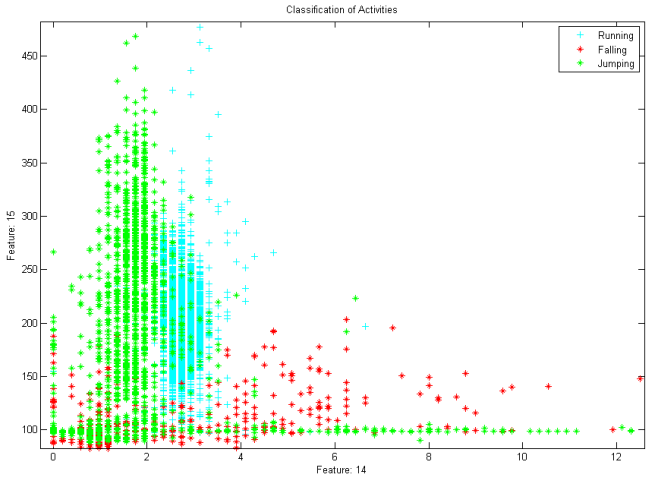


Figure D.9: The main frequency component of $|a|$ is the clue in order to distinguish *running* from *jumping*. It is helpful for the distinction of *running* and *falling* as well.

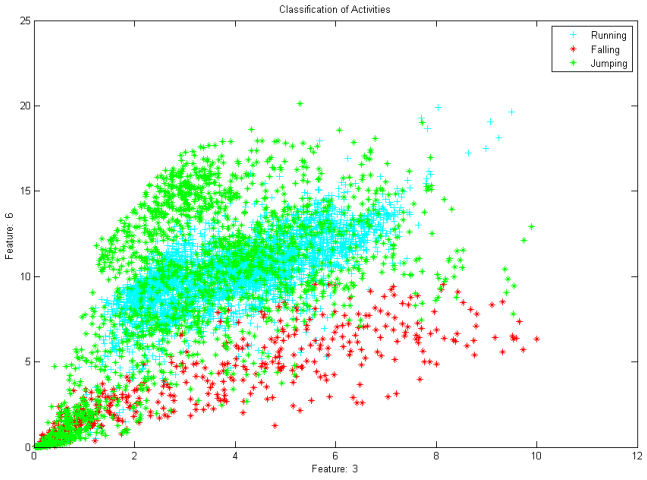


Figure D.10: Distinction between *falling* and *jumping* based on the body frame. The lower values correspond to the first phases of *falling* and *jumping* where the floor had not been hit yet.

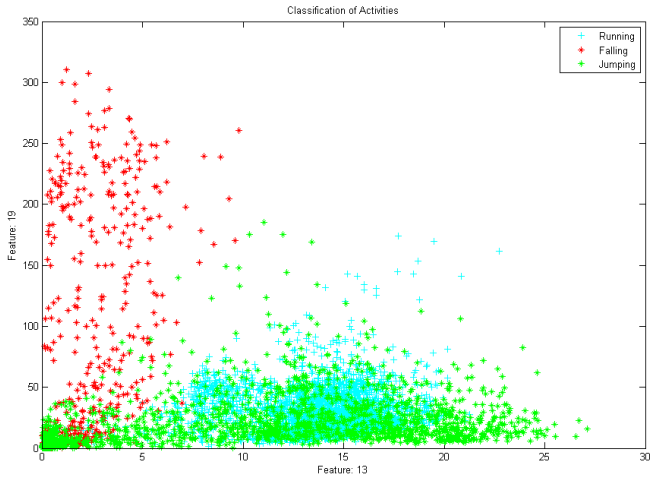


Figure D.11: Example of distinction between *falling* and *jumping* using the attitude of the sensor.

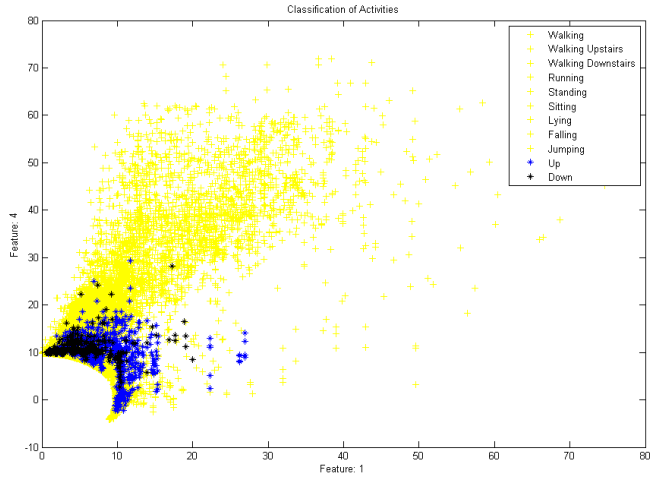
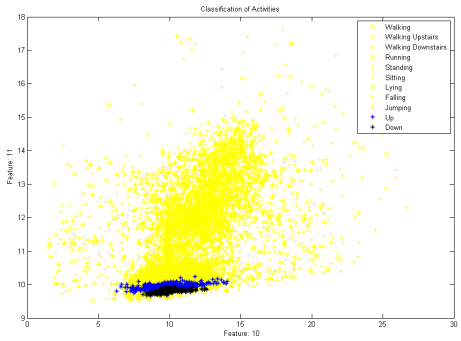
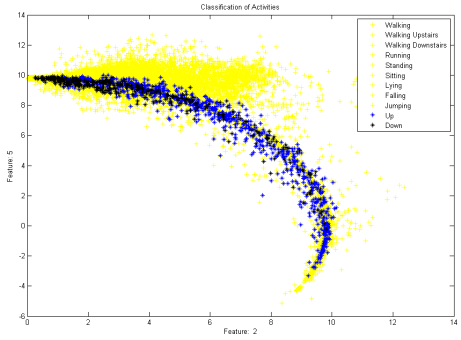


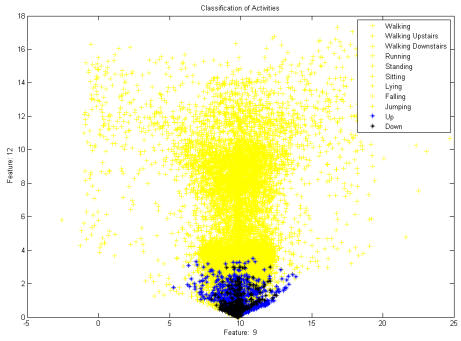
Figure D.12: Transitions *up* and *down* in contrast to the rest of the activities (first part).Two features of the final set of features are represented for the transitions *up* and *down* in contrast to the rest of the activities. These transitions are difficult to characterize, but can be summarized as a change in the attitude of the body with a higher value of acceleration regarding *static* activities but not as high as *non-static* activities.



(a)

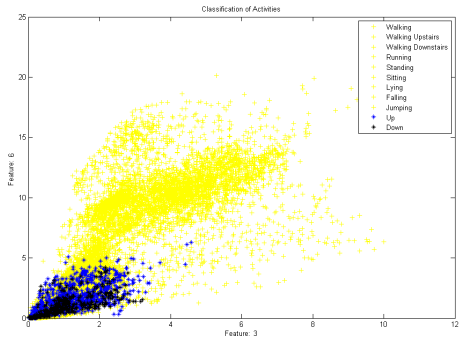


(b)

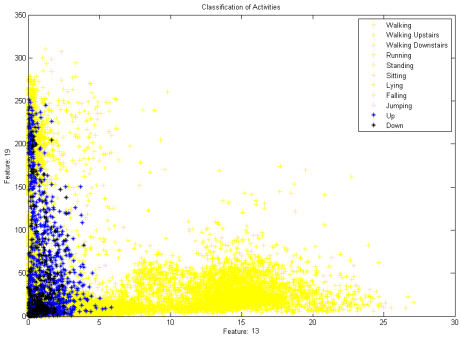


(c)

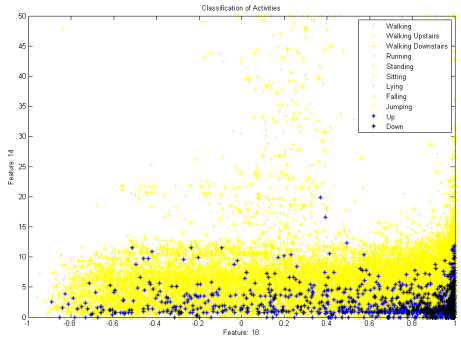
Figure D.13: Transitions *up* and *down* in contrast to the rest of the activities (second part).



(a)

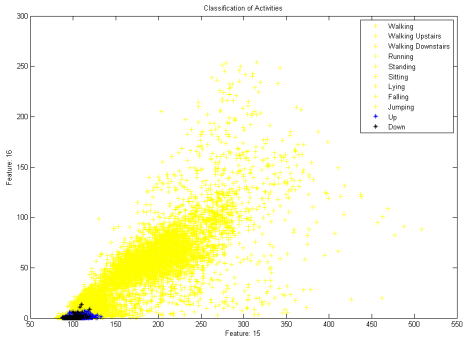


(b)

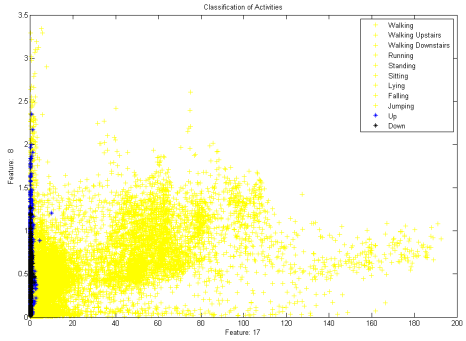


(c)

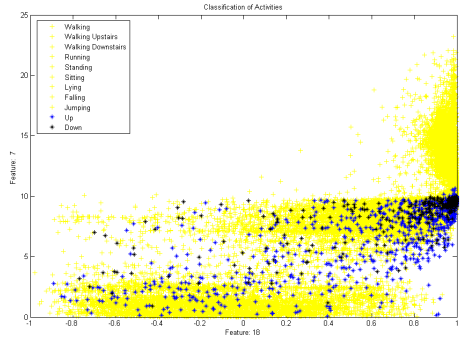
Figure D.14: Transitions *up* and *down* in contrast to the rest of the activities (third part).



(a)



(b)



(c)

Figure D.15: Transitions *up* and *down* in contrast to the rest of the activities (fourth part). The last features of the final set are plotted for the transitions in contrast to the activities.

Appendix E

MATLAB GUI for observing information signals

A MATLAB GUI (see Figures E.1 and E.2 for details) has been created in order to plot all sources of information required for the extraction of the features as well as the evolution of the features in time. Every log stored during the collection of data for the data set (see Appendix C) has attached its matrix with all the potential features computed in this Master Thesis. This GUI allows the user to observe signals related to acceleration and angular velocity in each frame and the evolution in time of one of the features.

To use it, the data structure explained in Appendix C must be used. With respect to the observation in time of the features, matrices are created, where the features are computed over different window lengths. One row of this matrix corresponds to all the features computed at a window length. One column represents the evolution in time of this feature as contains the same feature computed at different points of time.

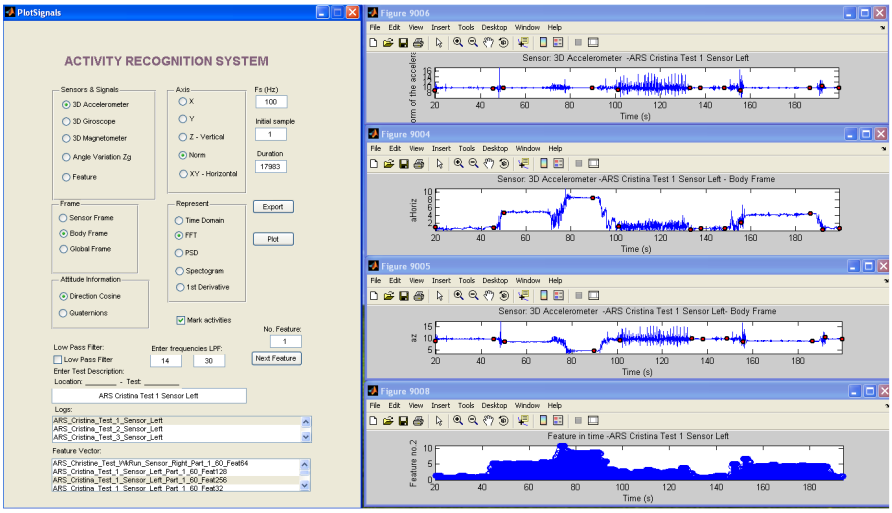


Figure E.1: MATLAB GUI for observing the information signals and the features in time.

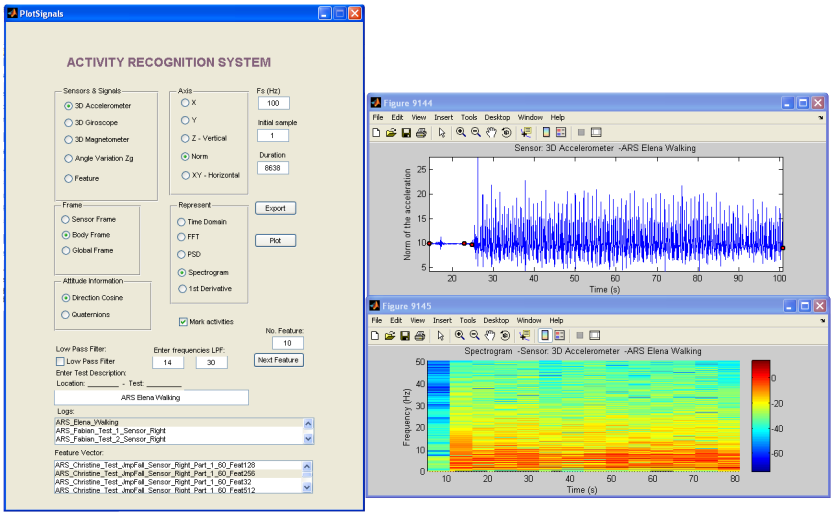


Figure E.2: MATLAB GUI for observing the information signals and the features in time. Another example.

Appendix F

MATLAB GUI for observing features

This appendix shows two MATLAB GUIs created to plot the features in 2D. Figure F.1 shows the GUI developed for plotting all the features computed in this Master Thesis over different window lengths. Figure F.2 shows the MATLAB GUI that only plots the final set of features whose window length is already set.

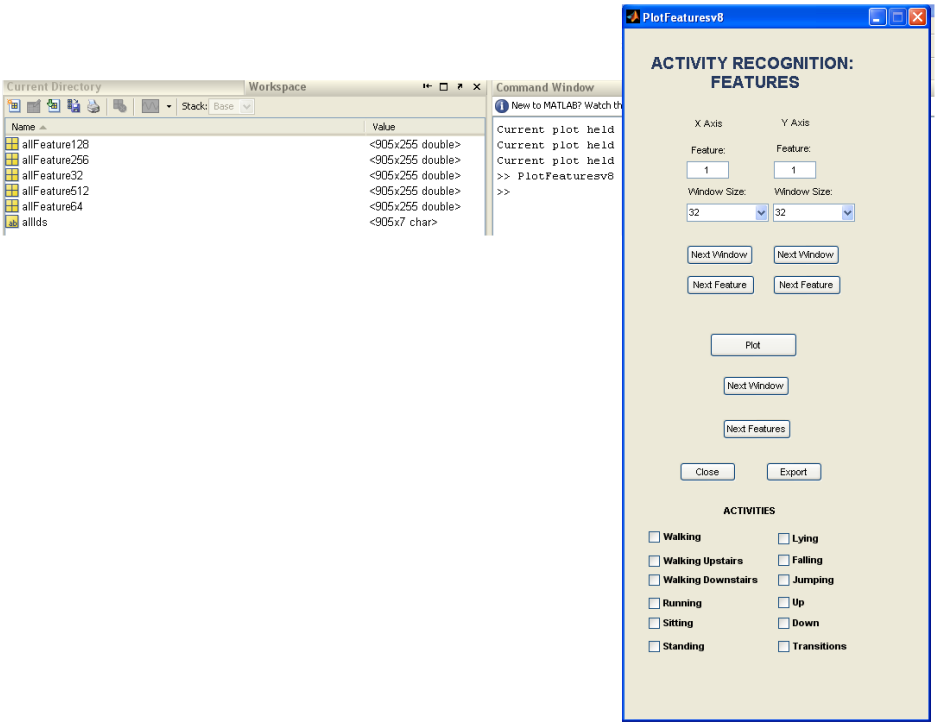


Figure F.1: MATLAB GUI for plotting the potential features in 2D. The window length for all the features can be selected. This GUI reads the MATLAB variables shown in the figure. Every variable contains the same features computed for different window lengths and one variable contains the information of the activity performed. One row of the matrices with the feature is the result of computing all the features at a point of time, where the activity shown by the same row of the matrix with the information related to the activities was performed. One column corresponds to one feature.

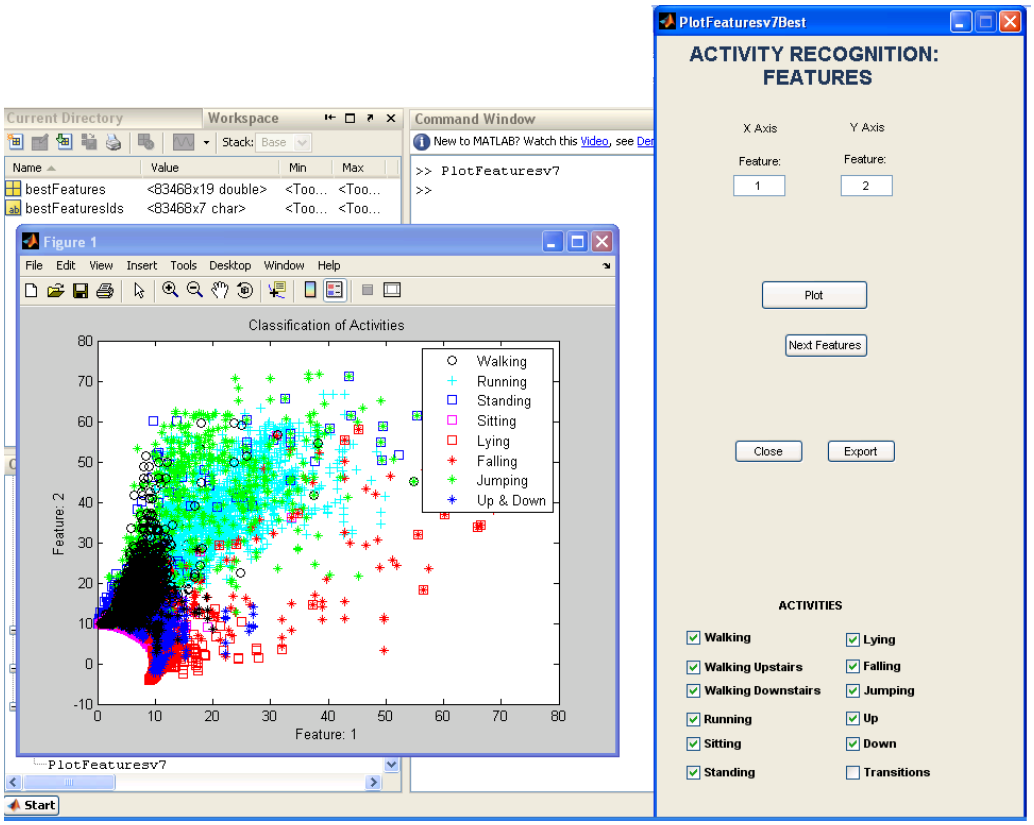


Figure F.2: MATLAB GUI for observing the final selection of best features. This GUI needs to know the values of the final set of features and the activity performed when these values were computed. To provide this information to the GUI, two matrices are created. The matrix *bestFeatures* contains the value of the final set of features that are computed from the data set collected in this work. One row of this matrix corresponds to all the features computed at an already set window length. One column represents the evolution in time of this feature as it contains the same feature computed at different points of time. The number of columns is equal to 19, that is the size of the final set of features. The information related to the activity that was performed in a point of time is given by the matrix *bestFeaturesIds*, in which one row is a string that represents the activity that was performed at that point of time. Both matrices are related: the first row of the matrix *bestFeatures* corresponds to the values of the final set of features that were computed during the performance of the activity indicated in the first row of the matrix *bestFeaturesIds*. The same occurs for the second row and so on.

Appendix G

Format Data Files

Every input or output file for the Java framework described in Chapter 5 is presented in this appendix. The aspects treated and explained in the following sections about every file are:

- ***Reference***: identifier used in this report or short name related to its function in the framework. The reference to some of the files that will be presented in this appendix are in Chapter 5.
- ***Name***: name or part of its name to identify it in the directory code.
- ***Type***: it can be an input or an output file from the point of view of the Java framework (see Chapter 5).
- ***Generation directory***: directory where it will be generated.
- ***Generation process***: it describes the current MATLAB script or the class or method of the Java code that generates this file.
- ***Destination directory***: possible destination directories where it should be included to be used for a script or program.
- ***Destination process***: it gives the MATLAB scripts or the Java processes that require this file.
- ***Contents***: brief description of the contents of this file and aim of the file.

- ***Structure***: if the file is used for other processes, its structure is presented using formal grammar. Every line of the file is a word that is generated by starting with the start symbol and then applying the production rules until no more nonterminal symbols are present. A complete reference in formal grammar is provided by Moll in [72].
- ***Other information***: extra information that is relevant.
- ***Example***: an example of every file will be provided.

The generation or destination directories will be given relative to the root directory of the Java or MATLAB code. If *JAVA* is written, it means it is the Java root directory. If it is *MATLAB*, then is the root directory of the MATLAB code.

G.1 Measurements or Benchmark Files

- ***Reference***: *measurementFiles* or *benchmarkFiles*.
- ***Name***: *receivedData*.
- ***Type***: input/output.
- ***Generation directory***: *JAVA*.
- ***Generation process***: main class of Java package *de.dlr.ARS.dataSetReceiver*.
- ***Destination directories***:
  - *JAVA/resources/activityLogs*
  - *JAVA/resources/activityBenchmark*
  - *MATLAB/BenchmarkFiles*
  - *MATLAB/MeasurementFiles*
  - *MATLAB/CheckJava*
- ***Destination processes***:
  - MATLAB script *ScriptProcessMeasurementFiles_1stPart* to include this log with data set format explained in Appendix C.

  - MATLAB script *ScriptProcessBenchmarksFiles_1stPart* to include this log with the benchmark data set format that is the same as data set format.
  - Off-line working mode of Java software (see Section 5.1.3.1 for details). This working mode can be used:
    * to store the numerical value of the features in *featuresFinalSetValuesFile* in order to compare it with MATLAB numerical value of the features. The script to execute to check these values is *Script_ResearchPart_CheckBestFeatureValuesWithJava*.
    * to store the inference results value of the system in *resultsInferenceFiles* that will be described in the following sections.

- ***Contents***: sensor data.
- ***Structure***:
  - $N = \{message, sensorID, time, a_x, a_y, a_z,$
    $w_x, w_y, w_z, m_x, m_y, m_z, sensorTemperature, time, C_{sg}, w\}$.
  - $\Sigma = \{r\}$.
  - P:
    $S -> \ numberOfMessage\ w\ sensorID\ w$
    $time\ w\ a_x\ w\ a_y\ w\ a_z\ w$
    $w_x\ w\ w_y\ w\ w_z\ w\ m_x\ w\ m_y\ w\ m_z\ w$
    $sensorTemperature\ w\ time\ w\ C_{sg}\ r$
- ***Other information***:
  - The subject should have the sensor already mounted to the belt, so the rotation matrix from the sensor frame to the body frame can be extracted from the first message.
  - The subject should remain in *standing* position for at least five seconds.
  - The message ID in which the person has been *standing* for at least five seconds, should be indicated. When this message is read, initial conditions are computed (see Section 4.8.7 and inference starts. The rotation matrix from sensor frame to body frame is updated.

```
0 1 0.000000e+000 -9.673672e+000 -7.202574e-001 9.814224e-001 3.609904e-
002 -2.593390e-002 3.372237e-002 9.311107e-001 1.168395e-001 -9.628174e-
002 0.000000e+000 0.000000e+000 -7.729283e-002 -9.805736e-002 -9.921746e-
001 9.965416e-001 -3.804732e-002 -7.387277e-002 -3.050581e-002 -
9.944531e-001 1.006590e-001
1 1 1.000000e-002 -9.644431e+000 -6.623190e-001 9.879425e-001 2.353214e-
002 -1.802413e-002 3.368314e-002 9.049698e-001 1.224882e-001 -1.167681e-
001 0.000000e+000 1.000000e-002 -7.696275e-002 -9.824983e-002 -9.921812e-
001 9.965603e-001 -3.824890e-002 -7.351485e-002 -3.072700e-002 -
9.944264e-001 1.008556e-001
2 1 2.000000e-002 -9.644452e+000 -6.187243e-001 9.840786e-001 3.851762e-
003 -2.596157e-002 2.362368e-002 9.068897e-001 1.215972e-001 -1.106520e-
001 0.000000e+000 2.000000e-002 -7.695024e-002 -9.896480e-002 -9.921111e-
001 9.965607e-001 -3.832101e-002 -7.347280e-002 -3.074747e-002 -
9.943528e-001 1.015732e-001
```

Figure G.1: *Emil_Benchmark_Sensor_Right receivedDataThread-4* content.

- ***Example***: see Figure G.1.

G.2 Labeling Files

- ***Reference***: *labelingFiles*.
- ***Name***: *outTimestamps*.
- ***Type***: input/output.
- ***Generation Directory***: *JAVA*.
- ***Generation process***: main class of Java package *de.dlr.ARS.dataSetReceiver*.
- ***Destination directories***:
  - *MATLAB/MeasurementFiles*.
  - *MATLAB/BenchmarksFiles*.
- ***Destination processes***:
  - MATLAB script *ScriptProcessMeasurementFiles_1stPart* to include this log with data set format explained in Appendix C.
  - MATLAB script *ScriptProcessBenchmarksFiles_1stPart* to include this log with the benchmark data set format that is the same as the format of the data set.

- ***Contents***: number of message where the activity or transition starts and the identifier of that activity or transition.
- ***Structure***:
  - $N = \{numberOfMessage, activityID, w\}$.
  - $\Sigma = \{r\}$.
  - P:
    $S -> \ numberOfMessage\ w\ activityID\ r$
- ***Other information***: this file is generated with the measurement or benchmark files.
- ***Example***: see Figure G.2.

```
2043 STNDING
6447 TRANSDW
6733 SITTING
10954 TRANSUP
11200 STNDING
12457 TRNSACC
12783 WALKING
14441 JUMPVRT
14515 STNDING
```

Figure G.2: *Emil_Benchmark_Sensor_Right outTimestamps* content.

G.3 Groundtruth Files

- ***Reference***: *groundtruthFiles*.
- ***Name***: *Groundtruth*.
- ***Type***: input.
- ***Generation Directory***: *MATLAB/QuantizationX/...*
- ***Generation process***: MATLAB script *ScriptProcessMeasurement-Files_3rdPart_QuantizationX*.
- ***Destination directories***: *JAVA/resources/activityData/QuantizationX/...*

- ***Destination processes***: training set is set from these files. The class *TestUsingBayesianNetworksAsClassifiers* in package *de.dlr.kn.classifiers* uses it for the training of the classifier (see Section 5.2 for details).

- ***Contents***: states of all the discrete random variables (RVs) involved including the target RV to infer.

- ***Structure***: The first line contains the name of the random variables. The rest of the file are the states of these random variables.

  - $N = \{concept, t\}$.
  - $\Sigma = \{r\}$.
  - P:
    $S -> \ concept\ t\ S$
    $S -> \ concept\ r$

  where *concept* can be:

  - *nameRV* for the first line of the file.
  - *stateRV* for the rest of the file.

- ***Other information***:

  - In the name of the file, information such as the frequency update of the features and the discretization are given.
  - Three kinds of files are generated using the MATLAB script commented above:
    * Groundtruth files per person and test.
    * Groundtruth files for four cross-validation.
    * Groundtruth files for four cross-validation with data from only one subject.
  - *nameRV* can be followed by two kind of suffixes:
    * If the name of the RV is followed by the string $\_DNAOG$, it means that the RV does not allow outgoing arrows. It can be a child of other RVs but it cannot be a parent of other RV.

- ∗ If the name of the RV is followed by the string *_AOOG*, it means that the RV allows only outgoing arrows. It cannot be a child of other RVs but it can be a parent of other RV.

- ***Example***: see Figure G.3.

f13	f14	f15	f16	f17	f18	f19	activity_AOOG
0.96+	0.6-2.2	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing
0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9	Standing

Figure G.3: *ARS . . . Quantization1Da_Best_Features_Groundtruth* content.

G.4 Evidence Files

- ***Reference***: *evidenceFiles.*

- ***Name***: *Evidence.*

- ***Type***: input.

- ***Generation Directory***: *MATLAB/QuantizationX/. . .*

- ***Generation process***: MATLAB script *ScriptProcessMeasurement-Files_3rdPart_QuantizationX*

- ***Destination directories***: *JAVA/resources/activityData/QuantizationX/. . .*

- ***Destination processes***: test set is set from these files. The class *TestUsingBayesianNetworksAsClassifiers* in package *de.dlr.kn.classifiers* uses it for the evaluation of the classifier (see Section 5.2 for details).

- ***Contents***: states of all the discrete random variables involved except the target random variable to infer

- ***Structure***: The first line contains the name of the random variables. The rest of the file are evidence.

- $N = \{concept, t\}$.
- $\Sigma = \{r\}$.
- P:

  $S -> \ concept \ t \ S$

  $S -> \ concept \ r$

where *concept* can be:

- *nameRV* for the first line of the file.
- *stateRV* for the rest of the file.

- ***Other information***: every evidence file requires another file with the groundtruth for the evaluation process. The considerations for *groundtruthFiles* are valid for the evidence files, too.
- ***Example***: see Figure G.4.

f12	f13	f14	f15	f16	f17	f18	f19
<2	0.96+	0.6-2.2	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9
<2	0.96+	3.75+	<101	<1	<1	<0.1	9.6-9.9

Figure G.4: *ARS . . . Quantization1Da_Best_Features_Evidence* content.

G.5 Inference Result Files

- ***Reference***: *resultsInferenceFiles*.
- ***Names***:
  - *Results_BN*
  - *Results_DBN*
  - *Results_NB*
  - *Results_DNB*

- ***Type***: output.
- ***Generation Directory***: *JAVA/resources/activityBenchmark*.
- ***Generation process***: files are created depending on the settings of the Java program for activity recognition (see Section 5.1.3.7 for more details).
- ***Destination directories***: *MATLAB/BenchmarkFiles*.
- ***Destination processes***: *ScriptProcessBenchmarksFiles_2ndPart* MATLAB script for the evaluation of the system
- ***Contents***: number of message where evidence was given to the system and results of the inference for every state of the target random variable and every recognition algorithm defined. In our case, static and dynamic Naïve Bayes and unrestricted Bayesian networks.
- ***Structure***:
  - The first line of the file is generated as follows:
    - $N = \{numberOfMessage, nameStateRV, t\}$.
    - $\Sigma = \{r\}$.
    - P:
      $S -> \ numberOfMessage\ t\ W$
      $W -> \ nameStateRV\ t\ W$
      $W -> \ nameStateRV\ r$
  - The rest of the file is generated as follows:
    - $N = \{numberOfMessage, nameStateRV, t\}$.
    - $\Sigma = \{r\}$.
    - P:
      $S -> \ message\ t\ W$
      $W -> \ probabilityStateRV\ t\ W$
      $W -> \ probabilityStateRV\ r$
- ***Other information***: name of the files are defined in the class *IMUBelt_ARS* (see Section 5.1.3.7).
- ***Example***: see Figure G.5.

```
NumberOfMessage  Sitting    Standing   Walking    Running    Jumping
      Falling    Lying UpDown
1475  7.339534653626076E-9   0.9998793070744314    2.3514543827171924E-10
      6.827314090806685E-14  8.65893088896067E-5   3.409497648003547E-5
      8.737541514010644E-10  1.9169635048373382E-10
1500  7.339534653626076E-9   0.9998793070744314    2.3514543827171924E-10
      6.827314090806685E-14  8.65893088896067E-5   3.409497648003547E-5
      8.737541514010644E-10  1.9169635048373382E-10
1525  7.339534653626076E-9   0.9998793070744314    2.3514543827171924E-10
      6.827314090806685E-14  8.65893088896067E-5   3.409497648003547E-5
      8.737541514010644E-10  1.9169635048373382E-10
1550  7.339534653626076E-9   0.9998793070744314    2.3514543827171924E-10
      6.827314090806685E-14  8.65893088896067E-5   3.409497648003547E-5
      8.737541514010644E-10  1.9169635048373382E-10
1575  7.339534653626076E-9   0.9998793070744314    2.3514543827171924E-10
      6.827314090806685E-14  8.65893088896067E-5   3.409497648003547E-5
      8.737541514010644E-10  1.9169635048373382E-10
1600  3.5891939459567413E-9  0.9998976498571284    4.496277923198395E-10
      3.3384424297942537E-13 7.102952188075046E-5  3.131554744251863E-5
      6.622717683308742E-10  3.721208020646786E-10
1625  7.339534653626076E-9   0.9998793070744314    2.3514543827171924E-10
      6.827314090806685E-14  8.65893088896067E-5   3.409497648003547E-5
      8.737541514010644E-10  1.9169635048373382E-10
```

Figure G.5: *Emil_Benchmark_Sensor_Right_Results_BN* content for the discretization 1Db.

G.6 Features Final Set Values Files

- ***Reference***: *featuresFinalSetValuesFiles*.
- ***Names***: *FeaturesFinalSetValues*.
- ***Type***: output.
- ***Generation Directory***: *JAVA/resources/activityLogs*.
- ***Generation process***: Files are created depending on the settings of the Java program for activity recognition (see Section 5.1.3.7 for more details).
- ***Destination directories***: *MATLAB/CheckJava*.
- ***Destination processes***: In order to compare the value of the features obtained with the JAva code and the MATLAB code, the MATLAB script *Script_ResearchPart_CheckBestFeatureValuesWithJava* should be used.

- ***Contents***: Every line contains the numerical value of the features that belong to the final set of features used for inference. In our case, 19 values are given.
- ***Structure***:
  - $N = \{valueFeature, w\}$.
  - $\Sigma = \{r\}$.
  - P:
    $S -> \ valueFeature\ w\ S$
    $S -> \ valueFeature\ r$
- ***Other information***: name of the file is defined in the class *IMUBelt_ARS*.
- ***Example***: see Figure G.6.

```
0.1661041468699766 9.927913202291807 9.83287143050812 9.82939350630457
0.09659201717984647 9.82758196836097 9.83274724485057
0.022759538329722703 0.03641380868892552 0.02357459471836458
0.02466205000784072 4.867773030008433E-7 0.9999750987230426 8.0078125
98.136079148253 2.5178516349275766 2.47830128887641 0.013652544789294601
9.827610243918398
0.1511509444107547 9.927913202291807 9.829988356736173 9.829573114779903
0.07840262221569433 9.829480655659486 9.829914589410027
0.022137502522924735 0.034695357676824856 0.023110900864101787
0.02547917838255387 4.773859682379399E-4 0.9999831968537023 8.203125
98.11755712474017 2.5215525480478176 2.5815877539715095
0.013155550963537162 9.82950782459238
0.14376416191131752 9.927913202291807 9.829374104768236 9.829618054012478
0.06489965785856398 9.830067532772908 9.82932331888576
0.02175676380906834 0.03259391433407214 0.021988635371516716
0.0220213437553749 0.0010315087126947966 0.9999928293629501 9.1796875
98.13930878482645 2.5172648395017454 2.617488206466101
0.011325053004736314 9.830092125659942
```

Figure G.6: *FeaturesFinalSetValues* content.

G.7 Network DAG Files

- ***Reference***: *networkDAG*.
- ***Names***: *Network*.
- ***Type***: output.

- ***Generation Directory***: *JAVA/resources/classification/...* The folders where these files are stored depend on approach, quantization and test. Possible directories are:
  - *.../NaïveBayes/QuantizationX/ApparentError*
  - *.../NaïveBayes/QuantizationX/GeneralTrainingXvalidation*
  - *.../StaticBayesianNetwork/QuantizationX/ApparentError*
  - *.../NaïveBayes/QuantizationX/GeneralTrainingXvalidation*
- ***Generation process***: The class *TestUsingBayesianNetworksAsClassifiers* in package *de.dlr.kn.classifiers* generates it after the training process of the network (see Section 5.2 for details).
- ***Destination directories***: *JAVA/resources/seedNetworks.*
- ***Destination processes***: The class *TestUsingBayesianNetworksAsClassifiers* in package *de.dlr.kn.classifiers* can set this network structure as the seed network for the learning algorithm (see Section 5.2 for details).
- ***Contents***: length of probability tables and network structure
- ***Structure***: This file contains a first part with information about the length of the probability tables and a second part including network structure. Both parts are splitted using two carriage return characters.
  - The first part of the file is generated as follows:
    - $N = \{nameRV, Length\ of\ probability\ table:, length, valueFeature, w, t\}$.
    - $\Sigma = \{r\}$.
    - P:
      $S -> \ nameRV\ t\ w\ Length\ of\ probability\ table:\ w\ length\ r$
  - The second part is generated as:
    - $N = \{sourceRV, destinationRV, —>, w\}$.
    - $\Sigma = \{r\}$.

* P:
  $S -> \ sourceRV \ w \ —>w \ \ destinationRV \ r$

- ***Example***: see Figure G.7.

```
f6    Length of Probability Table: 48
f7    Length of Probability Table: 72
f8    Length of Probability Table: 40
f9    Length of Probability Table: 40
f12   Length of Probability Table: 48
f11   Length of Probability Table: 56
f10   Length of Probability Table: 48
f16   Length of Probability Table: 32
f15   Length of Probability Table: 40
f14   Length of Probability Table: 32
f13   Length of Probability Table: 32
f19   Length of Probability Table: 56
f18   Length of Probability Table: 56
f17   Length of Probability Table: 48
f1    Length of Probability Table: 64
f3    Length of Probability Table: 64
activity_AOOG    Length of Probability Table: 8
f2    Length of Probability Table: 32
f5    Length of Probability Table: 64
f4    Length of Probability Table: 32


activity_AOOG ---> f6

activity_AOOG ---> f7

activity_AOOG ---> f8

activity_AOOG ---> f9

activity_AOOG ---> f12

activity_AOOG ---> f11
```

Figure G.7: *Test_NaiveBayes_Quantization1Da . . . _Network* content.

G.8 Evaluation of the Classifier Files

- ***Reference***: *evaluationClassifiers*.
- ***Names***: *Evaluation_Classifier*.

- ***Type***: output.
- ***Generation Directory***: *JAVA/resources/classification/...*. The folders where these files are stored depend on approach, quantization and test. Possible directories are:
  - *.../NaïveBayes/QuantizationX/ApparentError*
  - *.../NaïveBayes/QuantizationX/GeneralTrainingXvalidation*
  - *.../StaticBayesianNetwork/QuantizationX/ApparentError*
  - *.../NaïveBayes/QuantizationX/GeneralTrainingXvalidation*
- ***Generation process***: the class *TestUsingBayesianNetworksAsClassifiers* in package *de.dlr.kn.classifiers* generates it after giving all evidence of the test set to the classifier and checking the output (see Section 5.2 for details).
- ***Contents***: confusion matrix and other parameters to evaluate the classifier.
- ***Example***: see Figure G.8.

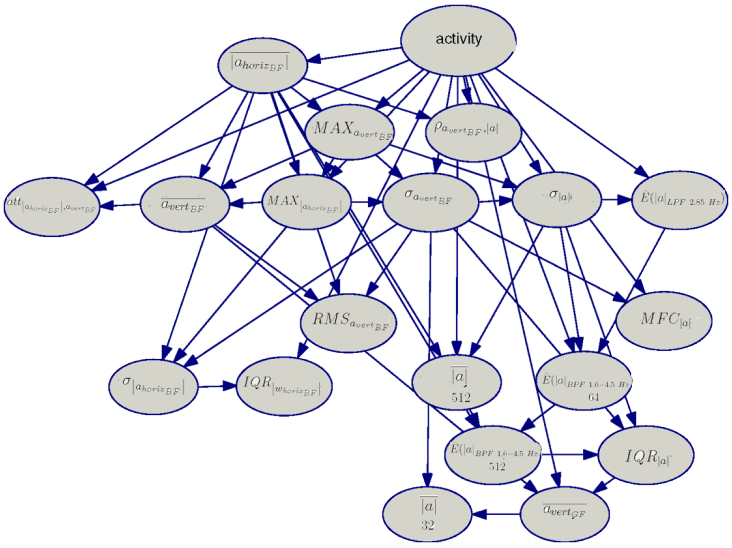
```
CONFUSION MATRIX:

          Sitting Standing Walking Running Jumping Falling Lying UpDown
Sitting     19871    12613      15       0       0     104   498   1592
Standing     2620    60682     385      44     396     176     0   1761
Walking        15      240   39197     143     425     188     0   1448
Running         0       19      76    8514     182      42     0     31
Jumping         8      288     142     419    3051     210     0    297
Falling         3       99      58       0      14     732    24    193
Lying           8        0       0       0       0     325 15629    299
UpDown        157      359     241       0       0     282   307   1835


 ACCURACY: 0.8482556721151501


 CLASSIFICATION ERROR: 0.15174432788484996
Note: If the test set is  the training set,  this error is called 'apparent classification error'. If the test set  is
diffferent, it  is called 'true classification error'.

  CLASS:      RECALL:      PRECISION:      F-SCORE (B=0.5):      F-SCORE (B=1):      F-SCORE (B=2):
Sitting     0.5727668405730263     0.8760691297063751     0.7921719648224779     0.6926710239651417     0.6153765159116529
Standing    0.9185335432308065     0.8167160161507402     0.8352327783650458     0.8646376563791286     0.8961885182953484
Walking     0.9409688880353371     0.9771401505570873     0.9696851250791638     0.9587134645958175     0.9479873076067291
Running     0.9605144404433213     0.9335526315789474     0.9388232180663372     0.9468416370106761     0.9549982053122756
Jumping     0.6910532276330691     0.75  0.7374196355198916     0.7193209949310386     0.7020894698085542
Falling     0.6518254674977738     0.3555123846527404     0.3910674217330911 7     0.4600879949717159     0.5586933292627079
Lying 0.9611340015989177     0.9496293595819663     0.9519082016737115     0.9553470460588649     0.9588108267281785
UpDown      0.5768626218170386     0.2461105151502145923     0.2779881836085441 6     0.3450220926953088     0.4546580773042616
```

Figure G.8: *Test ...._SampleUpdate_10_Evaluation_Classifier* content.

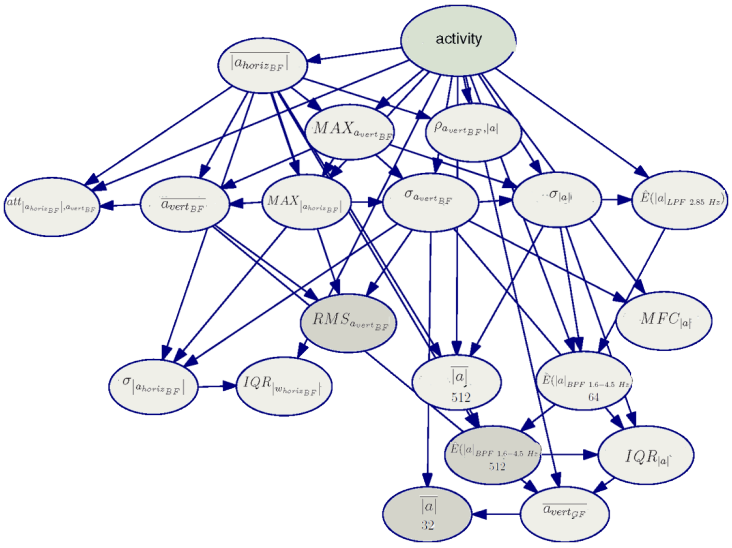
Appendix H

Maximal reduction of a machine-learnt Bayesian network if all input values are observed

As explained in this Master Thesis in Section 4.9.4, machine-learnt Bayesian networks can be reduced under the constraint that all input values are observed. This appendix shows the maximal reduction of the machine-learnt Bayesian network explained in Chapter 6 for efficient inference of motion related activities. See Figures H.1 to H.3 for an explanation of how the reduction of the machine-learnt Bayesian network have been done. During this process, the initial network of 19 nodes and 49 edges has been reduced to 16 nodes and 32 edges, decreasing significantly the complexity of the network.

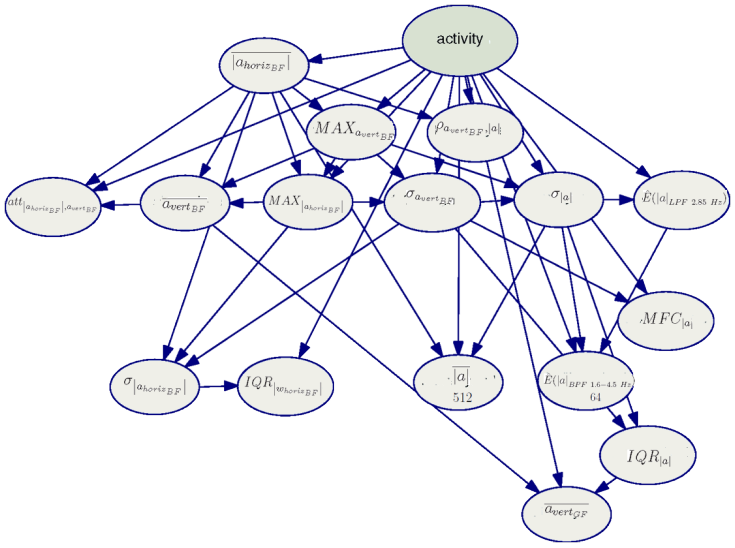


(a)

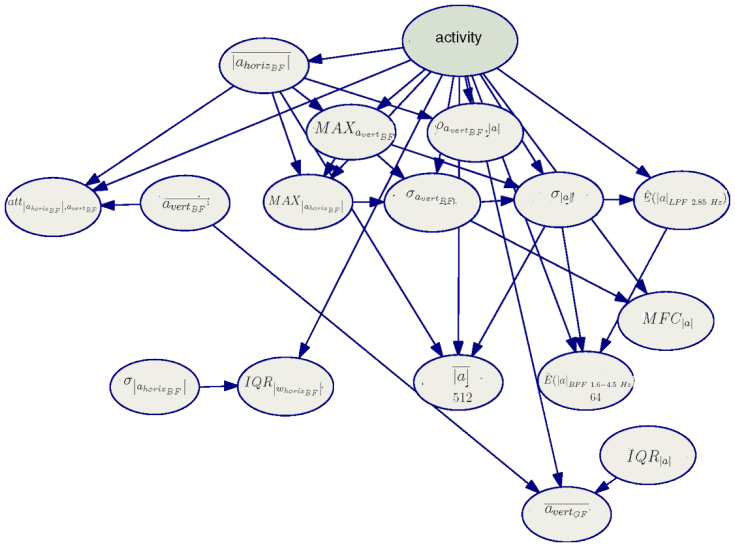


(b)

Figure H.1: (a) Machine-learnt Bayesian network structure. It has 19 nodes and 49 edges (b) Identification of the features that are out of the MB of *activity*.



(a)



(b)

Figure H.2: (a) Three nodes and nine edges were removed as nodes out of MB of *activity* are irrelevant (b) Eight incoming edges to parents of children removed.

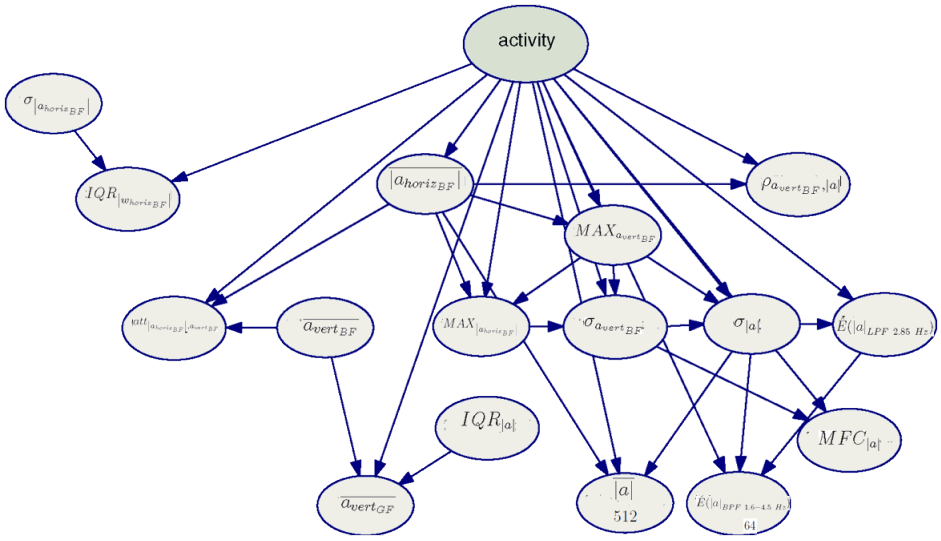


Figure H.3: Identification of two different subnets. The final network has 16 nodes and 32 edges.

Appendix I

Confusion Matrices

This appendix shows the confusion matrices for the discretization 1Da considering that the features are updated at 4 Hz. Tables from I.1 to I.4 are obtained through the resubstitution and four cross-validation tests that consider the activities equiprobable. Tables I.5 and I.6 contain the confusion matrices for unrestricted Bayesian networks and dynamic unrestricted Bayesian networks from the two unknown subjects after considering the recognition delay. More details about these tests can be found in Chapter 6.

	Sitting	Standing	Walking	Running	Jumping	Falling	Lying	Up & Down
Sitting	8950	4013	7	0	0	37	129	740
Standing	2393	22905	160	11	160	83	0	745
Walking	5	103	15596	61	187	90	0	618
Running	0	7	26	3418	71	11	0	12
Jumping	4	121	49	158	1262	68	0	109
Falling	3	36	18	0	5	300	8	80
Lying	4	0	0	0	0	167	6195	139
Up & Down	78	119	56	0	0	102	103	808

Table I.1: Confusion matrix for the static Naïve Bayes resubstitution test and the discretization 1Da. The final set of features is updated at 4 Hz.

	Sitting	Standing	Walking	Running	Jumping	Falling	Lying	Up & Down
Sitting	9207	4080	5	1	16	29	145	393
Standing	1909	23450	124	23	184	181	0	586
Walking	13	125	15756	61	139	146	0	420
Running	1	6	26	3433	71	5	0	3
Jumping	4	119	39	47	1465	60	0	37
Falling	3	24	12	0	9	377	4	21
Lying	6	0	0	0	0	72	6186	241
Up & Down	75	104	38	1	16	35	44	953

Table I.2: Confusion matrix for the static unrestricted Bayesian network resubstitution test and the discretization 1Da. The final set of features is updated at 4 Hz.

	Sitting	Standing	Walking	Running	Jumping	Falling	Lying	Up & Down
Sitting	8928	4022	6	0	0	41	129	750
Standing	2413	22872	162	10	159	99	0	742
Walking	4	109	15576	71	199	96	0	605
Running	0	8	28	3400	84	13	0	12
Jumping	4	120	49	164	1258	65	0	111
Falling	3	37	19	0	6	290	8	87
Lying	2	0	0	0	0	187	6191	125
Up & Down	78	121	57	0	0	118	102	790

Table I.3: Confusion matrix for the static Naïve Bayes four cross-validation test and the discretization 1Da. The final set of features is updated at 4 Hz.

	Sitting	Standing	Walking	Running	Jumping	Falling	Lying	Up & Down
Sitting	8961	4250	10	1	15	36	145	458
Standing	2040	23223	144	21	240	200	1	588
Walking	15	150	15642	75	179	150	0	449
Running	1	10	38	3300	169	16	1	10
Jumping	5	145	50	57	1403	69	0	42
Falling	6	26	18	1	33	304	26	36
Lying	20	0	0	0	1	96	6142	246
Up & Down	103	124	54	5	28	51	56	845

Table I.4: Confusion matrix for the static unrestricted Bayesian network four cross-validation test and the discretization 1Da. The final set of features is updated at 4 Hz.

	Sitting	Standing	Walking	Running	Jumping	Falling	Lying
Sitting	162	0	1	0	0	0	0
Standing	1	476	13	0	0	1	0
Walking	0	0	458	0	0	0	0
Running	0	3	9	30	0	1	0
Jumping	0	2	3	0	10	0	0
Falling	0	0	0	0	0	4	0
Lying	0	0	0	0	0	1	188

Table I.5: Confusion matrix for the static unrestricted Bayesian network aproach using data extracted from two unknown subjects. The discretization selected is 1Da. The final set of features is updated at 4 Hz. Transitions are not evaluated. Recognition delay has been considered.

	Sitting	Standing	Walking	Running	Jumping	Falling	Lying
Sitting	163	0	0	0	0	0	0
Standing	2	479	5	0	1	0	0
Walking	0	0	458	0	0	0	0
Running	0	0	3	40	0	0	0
Jumping	0	0	1	0	14	0	0
Falling	0	0	0	0	0	4	0
Lying	2	0	0	0	0	1	186

Table I.6: Confusion matrix for the dynamic unrestricted Bayesian network aproach using data extracted from two unknown subjects. The discretization selected is 1Da. The final set of features is updated at 4 Hz. Transitions are not evaluated. Recognition delay has been considered.