Sian Lun Lau

Towards a user-centric context aware system

Empowering users through activity recognition using a smartphone as an unobtrusive device

kassel university press

Sian Lun Lau

Towards a user-centric context aware system: empowering users through activity recognition using a smartphone as an unobtrusive device

kassel university press

This work has been accepted by the faculty of Electrical Engineering and Computer Science of the University of Kassel as a thesis for acquiring the academic degree of Doktor der Ingenieurwissenschaften (Dr.-Ing.).

Supervisor: Prof. Dr.-Ing. Klaus David, Universität Kassel
Co-Supervisor: Prof. Dr. rer. nat. Daniela Nicklas, Carl von Ossietzky Universität Oldenburg

Defense day: 22. November 2011

Bibliographic information published by Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at http://dnb.d-nb.de.

Zugl.: Kassel, Univ., Diss. 2011
ISBN print: 978-3-86219-244-1
ISBN online: 978-3-86219-245-8
URN: http://nbn-resolving.de/urn:nbn:de:0002-32456

Printed in Germany

Acknowledgement

I am immensely grateful for all the assistance and advice, without which I would not have made it to the completion of this thesis.

First and foremost, I would like to express my deepest gratitude to Prof. Dr.-Ing. Klaus David for his support, constant encouragement and willingness to discuss and consult. He has never given up on pushing me to move forward and pursue the goal I have chosen and started years ago. Also, I am grateful to Prof. Dr. rer. nat. Daniela Nicklas for agreeing to be my second adviser. I would also like to thank Prof. Dr. Kurt Geihs and Prof. Dr.-Ing. Dieter Wloka for accepting the invitation to review this thesis.

The team at ComTec throughout the years has been a great help and inspiration to me. I would like to thank Dr. Niklas Klein for his timely help in many areas towards the last phase of the thesis. I am also thankful to Andreas Pirali, who was always available for quick discussion and feedback. The students I have supervised and worked with have also been a great help to me, especially Stefan Reiser, Yaqian Xu. I would also like to thank the participants that had contributed to the data collection.

Furthermore, I am truly grateful to my beloved wife, Tze Ying and the little angel Hua En. Thank you for your patience and understanding particularly for the last one and a half year. Without your sacrifices, I would not be able to complete this race on my own. I would also like to express my gratitude to our family for their support, and our dear Christian brothers and sisters who have supported me through words and prayers.

And finally, I am ever grateful to God, for His providence, guidance and comfort.

Abstract

Context awareness is a research area that aims to utilise contexts to ease a user's tasks and hence fulfil his needs. Newer and more innovative approaches in different aspects, such as communications, human-computer interactions, applications and systems have been investigated and proposed to enable the acquisition of implicit information from available sensor data. This information, commonly defined as context, supports context aware systems to act and react based on the available contexts and their changes.

In getting user acceptance for the proposed approaches and ideas based on context awareness, it is important to ensure that the target realisations, both software and hardware, are user-centric. The hardware devices should be seamlessly integrated in the users' environment, without requiring them to make noticeable adjustments or even compromise their daily lives. From the software point of view, users should be given means for adequate understanding and overview of a context aware system as well as sufficient control for potential creation, modification and management of the desired services and functions. We name the envisioned system a user-centric context aware system.

With this motivation, we have investigated in this thesis, selected areas in context awareness for potential unobtrusive approaches that may be built based on past experiences and techniques. We have reviewed and discussed definitions and basic concepts related to context awareness. An active and established area is activity recognition using sensor devices and classification methods. Instead of applying numerous sensor devices, as observed in many previous investigations, we are proposing the use of a smartphone with its built-in accelerometer as an example solution. The proposed solution includes software and hardware that are meant to be unobtrusive to users. The possibilities of using a smartphone to enable recognition of basic activities are discussed, such as walking, going up the staircase, sitting etc.

In this thesis, the proposal is tested experimentally via evaluations on real data obtained from 15 test users. A prototype application is developed to demonstrate and evaluate the selected classification methods for the designated recognition tasks. The evaluations investigate factors and techniques that are deemed applicable and suitable for the vision of an unobtrusive user-centric context aware system.

Zusammenfassung

Kontextsensitivität ist ein Forschungsgebiet, dessen Ziel es ist, Kontexte zu nutzen, um Nutzer-Aufgaben zu unterstützen und Nutzer-Bedürfnisse zu erfüllen. Neuere und innovativere Ansätze in verschiedenen Gebieten, wie Kommunikationstechnik, Mensch-Computer-Interaktion, Anwendungen und Systeme wurden untersucht und vorgeschlagen, um den Erwerb von impliziten Informationen aus verfügbaren Sensordaten zu ermöglichen. Diese Informationen, die gemeinhin als Kontext bezeichnet werden, ermöglichen es kontextbezogenen Systemen auf der Grundlage der verfügbaren Kontexte und ihrer Veränderungen zu handeln und zu reagieren.

Damit die vorgeschlagenen, auf Kontextsensitivität basierten Konzepte und Ideen von den Anwendern akzeptiert werden, ist es wichtig sicherzustellen, dass die angestrebten Lösungen, sowohl Soft- als auch Hardware, anwenderorientiert sind. Die Hardware-Geräte sollten sich nahtlos in die Benutzer-Umgebung integrieren, ohne dass die Nutzer spürbare Anpassungen vornehmen oder sogar in ihrem Alltag Kompromisse eingehen müssen. Aus der Software-Sicht sollten Nutzern Mittel zur Verfügung gestellt werden, damit sie ein ausreichendes Verständnis und einen Überblick über das kontextbezogene System erlangen können, bzw. die gewünschten Dienste und Funktionen erstellen, ändern und verwalten können. Wir nennen das anvisierte System ein nutzerzentriertes kontextbezogenes System (User-centric Context Aware System).

Mit dieser Motivation haben wir in dieser Dissertation, in ausgewählten Bereichen, Kontextsensitivität für potenzielle unauffällige Ansätze untersucht, die auf der Grundlage vorheriger Erfahrungen und Techniken aufgebaut werden können. Wir haben Definitionen und grundlegende Konzepte, die inhaltlich mit der Kontextsensitivität zusammenhängen, überprüft und uns damit auseinandergesetzt. Ein aktiver und etablierter Bereich ist die Aktivitätserkennung mit Sensor-Geräten und Klassifikationsverfahren. Anstatt, wie in vielen früheren Untersuchungen beobachtbar, zahlreiche Sensoreinrichtungen einzusetzen, schlagen wir den Einsatz von Smartphones mit integriertem Beschleunigungssensor als Beispielslösung vor. Die vorgeschlagene Lösung beinhaltet Nutzung von Soft- und Hardware, die für Nutzer unmerklich sein sollten. Möglichkeiten, über die Nutzung eines Smartphones, grundlegende Tätigkeiten wie gehen, die Treppe hinauflaufen, Sitzen usw. zu erfassen, werden hier diskutiert.

In dieser Dissertation wird der Vorschlag experimentell, durch Auswertungen realer, von 15 Testpersonen erhaltener Daten getestet. Eine prototypische Anwendung wurde entwickelt, um die ausgewählten Klassifizierungsmethoden für die betrachteten Erkennungsaufgaben zu demonstrieren und zu evaluieren. Die Auswertungen untersuchen Faktoren und Techniken, die als anwendbar und für die Vision von einem unauffälligen, nutzerzentrierten kontextbezogenen System geeignet sind.

Contents

Abbreviations and Notations

The list below gives an overview of abbreviations and notations used throughout this thesis. Each notation is accompanied with a brief explanation and the page number of its first occurrence.

Notation	Explanation	Page
Ho	The null hypothesis used in this thesis that the two accuracies are not significantly different, compared using the Wilcoxon signed-rank test	82
Δ	Difference between the compared values	82
α	The significance level used for the Wilcoxon signed-rank test. In this thesis, α is equal to 0.05	82
$\bar{x}$	Average accuracy	81
3D	Three Dimensional	25
6MWT	6-minute walk test	24
ADL	Activities of daily living	21
API	Application Programming Interface	43
BN	Bayesian Network	71, 72
C4.5	Decision tree learning algorithm by Ross Quinlan	67
CA	Context Acquisition	30
CARMA	Context Aware Remote Monitoring Assistant	3
CASM	Context Aware Service Manager	31
CASS	Context aware Sub-Structure	11
CHF	Chronic heart failure	24
CMF	Context Management Framework	11
CoBrA	Context Broker Architecture	11
CP	Context Processing	30–32
CS	Context Server	32
DAG	directed acyclic graph	71
DL	Description length	69

Notation	Explanation	Page
DT	Decision Tree	66
ECG	Electrocardiogram	24
FFT	Fast Fourier transform	57, 58
GPS	Global Positioning System	28
HMM	Hidden Markov Model	26
ICA	Independent Component Analysis	58
IR	Infrared	22
J48	Weka's implementation of Quinlan's C4.5 revision 8 in Java	66, 67
KD-Tree	k-Dimensional tree	70
KNN	k-nearest neighbour	68, 69
LHS	Left-hand side	68
MavHome	Managing An Intelligent Versatile Home	24
MDL	Minimum Description Length	68
MEMS	Micro-electro-mechanical system	42
MIT	Massachusetts Institute of Technology	21
MSB	Multi-modal sensor board	22
MSP430	Texas Instrucment's Mixed Signal Processor 430	23
NB	Naïve Bayes	71, 72
p-value	The p-value from the Wilcoxon signed-rank test	82
PARCTAB	Palo Alto Research Center Incorporated Tab	8
PCA	Principal Component Analysis	58
PDA	Personal Digital Assistant	21
QP	Quadratic programming	73
RFID	Radio-frequency identification	15, 16
RHS	Right-hand side	68
RIPPER	Repeated Incremental Pruning to Produce Error Reduction	68

Notation	Explanation	Page
SA	Service Action	31, 32
SAX	Symbolic Aggregate approXimation	26
SEE	Service Execution Environment	30
SMASH	SMArt SHirt	23
SMO	Sequential Minimal Optimization	73
SMS	Short Messaging Service	31
SoA	Service-oriented Architecture	29
SOCAM	Service-Oriented Context Aware Middleware	11
SPOT	Sun Small Programmable Object Technology	86
SVM	Support vector machine	72
TEA	Technology Enabling Awareness	23
UMTS	Universal Mobile Telecommunications System	29
WEKA	Waikato Environment for Knowledge Analysis	67
WLAN	Wireless Local Area Network	29

1 Introduction

In the early 90s, Mark Weiser proposed the vision of ubiquitous computing [1]. He envisioned a future where computers eventually disappear and users are no longer required to interact with computers like how we use desktop computers today. This vision challenges researchers to look into newer and more innovative approaches in different aspects, such as communications, human-computer interactions, applications and systems. An area that explores various approaches that enable devices and systems to understand information in ubiquitous computing is context awareness [2].

The concept of a context aware system has the intention to utilise contexts to ease a user's tasks and hence fulfill his needs. In many investigations, there were a lot of effort focused in enabling the acquisition of implicit information from available sensor data in order to allow the supporting context aware system to act and react based on the information and its changes. In other words, instead of requesting the user to instruct the system to perform a task, the system should carry out this task automatically and in a timely manner based on the current situation and conditions of the user and his environment. Therefore, the system needs to have the capability to recognise and understand the situation and conditions - and such capability can be best described as *awareness*.

In the late 90s, more contributions were made by researchers in defining design issues. This brought us closer to the vision where devices can detect contexts and context changes. In the past 10 years, the development of context aware related studies has moved into various application domains such as tourism, health care, office and home automation. Through these different efforts, some of these results are slowly making its way into consumer products. Location-based services are a good example, where users can use location estimation techniques to enjoy services that adapt and react based on location changes.

The progress achieved so far is by no means an end to the research of context awareness. What we see in various investigations and their prototype applications as stand alone and limited solutions, will soon be integrated into newer advanced and consumer ready hardware and software. A good example is the popularity and ubiquity of smartphones today. Smartphones are mobile phones that offer advanced computing ability like a personal computer. They are also usually equipped with multiple sensors that enable enhanced features such as automatic change of screen-orientation or brightness adjustment. With expanding features, increasing comput-

ing powers, newer sensors and innovative communication interface, a smartphone is an ideal device and platform for potential context aware applications.

The goal of this thesis is to investigate how we can empower users with unobtrusive context aware devices. We aim to propose suitable ways to design and implement a context-aware system that uses a smartphone as an unobtrusive device for the users. Analysis, evaluations and a proof-of-concept implementation are carried out to investigate whether the goal is achievable.

The following section elaborates the key challenges identified within the scope of this thesis. This is followed by a list of problem statements and the contributions.

1.1 Problem statements

Context awareness is an attractive vision because it enables devices to "understand" implicit information that takes place around the human users through the use of contexts. This characteristic enables systems to act and react beyond threshold-based triggering, which the latter is commonly implemented today in various systems to automate system behaviours. By enabling the acquisition and interpretation of context, a context aware system utilises the available contexts to offer at least the following advanced features:

- Learning and understanding of user and service behaviours.
- Recording and presentation of the obtained contexts.
- Intelligent adaptation through reasoning or prediction of available contexts of the user and his environment.
- Potential further inference of implicit contexts based on the available contexts that are not directly measurable using sensors.

To date, the above features are still active research areas in finding suitable techniques to apply them in the many application domains and environments.

One of the challenges in designing and developing a context aware system is to have a system that fulfils users' needs and at the same time ensure users' acceptance for the system [3]. The issue of obtrusiveness is, to the best of our knowledge, not often discussed in many of the previous investigations in the area of context aware system. If a context aware system is found obtrusive to a user, either from the perspective of hardware or software, it is likely that the user is reluctant to adopt that system. [4] has defined 8 different dimensions of obtrusiveness for the field of home telehealth technology, which are physical, usability, privacy, function, human interaction, self-concept, routine and sustainability. These dimensions are also applicable for the area of context awareness to answer the highlighted challenges.

This thesis contends that users can adopt and use context aware systems in a unobtrusive manner. Up to now, many investigations have been focusing on innovative

ways that demonstrate the potential and usefulness of such systems. However, these proposed techniques and approaches are not always practical for the daily lives of the users.

In particular, the acquisition of sensor data and context information, the investigated sensor devices, such as multiple body-worn sensors as well as vision and audio based sensors, may be often seen as obtrusive and can bring inconvenience and discomfort to the users. This may likely lead to the rejection of the newly introduced technology by the users. In order to avoid this, we need to investigate approaches that are unobtrusive and more likely to be accepted by the target users.

In this thesis, we address the issue with suitable sensors and computing devices for a user-centric context aware system. It is more likely for a user to accept a new technology, if the proposed technology is seen as suitable and does not require the user to alter his or her normal habits and lifestyle. Therefore, we propose the use of a smartphone as a suitable sensor and computing device.

1.2 Contributions

This thesis provides a definition of a user-centric context aware system that is unobtrusive for common users. Besides the definition, requirements are identified for the design and implementation of such a system.

Based on the definition and requirements provided, activity recognition is selected as the area of investigation. An architecture is proposed and developed for the user-centric context aware system. The generalised approach consists of well-defined core function components. This allows a more flexible realisation of such a system using one or more devices, depending on the designated implementation environment.

We select commercially available smartphones as suitable unobtrusive devices. The acceleration data from the smartphones are used for the designated activity recognition. We recapitulate supervised classification algorithms that are previously used for activity recognition using multiple accelerometers. We propose the use of meta-level classifiers to improve recognition accuracy by combining two or more base-level classifiers. Evaluations are performed using real data to find out which classifier performs best.

Apart from the comparison of classifiers, we also provide an investigation on the impacts of pre-processing and feature extraction conditions have on the selected classifiers. The comparisons provide further understanding on how one can improve the performance of the classifiers in a realisation of the recognition function in a user-centric context aware system.

The results of this thesis are the Context Aware Remote Monitoring Assistant (CARMA) prototype and evaluation tools. They are developed and tested in two national projects Middleware-Plattform für die Realisierung internetbasierter telemedizinischer Dienste (MATRIX) [5] and VENUS [6].

1.3 Outline of the thesis

This thesis is organised as follows. In Chapter 2, the concepts and definitions related to a user-centric context aware system are elaborated. The chapter first presents definitions and introduction to context, context awareness and the related applications as well as systems. Section 2.3 defines the user-centric context aware system and presents the requirements for the system. This is followed by the discussion on the topic of activity recognition, which is the selected area of focus in the thesis.

Chapter 3 discusses the idea of using unobtrusive sensor devices for the purpose of activity recognition. Section 3.1 presents the related work in this area. It covers selected previous investigations grouped in three categories: sensors used, applications and recognition techniques. The discussion is followed by the introduction of using smartphones for activity recognition and the architecture for the proposed user-centric context aware system. Questions, which this thesis intend to investigate and answer, are presented in Section 3.4.

Chapter 4 covers the different steps necessary for the preparation of the accelerometer of a smartphone for the designated recognition. This includes data collection (Section 4.2), pre-processing of acceleration data (Section 4.3) and feature extraction (Section 4.4).

The evaluations performed are presentqed in Chapter 5. The chapter begins with the introduction to the selected base- and meta-level classifiers. This is followed by the comparisons and discussions of the evaluation results. Empirical comparisons are performed to identify suitable classifiers and the recommended pre-processing conditions and features.

Chapter 6 continues with further evaluations where selected issues on recognition performances in a real implementation are discussed. Section 6.2 compares recognition accuracies using additional data. Performance factors such as speed and influence of battery life of a smartphone are covered in Section 6.4.

In Chapter 7 the conclusion and outlook are presented.

1.4 Publications

Parts of the work conducted within the scope of this thesis have been published in conferences or workshops and as a book chapter in a book. These publications are as follows:

- K. David, S. L. Lau, and B. N. Klein, "Social Link App - a new communication paradigm empowering mobile users," *IEEE Vehicular Technology Magazine*, Vol. 6, No. 3, Sept. 2011, pp. 80-87.
- B. N. Klein, S. L. Lau, and K. David, "Evaluation of the influence of time synchronisation on classification learning based movement detection with accelerometers," in *Proc. 2011 IEEE/IPSJ 11th International Symposium on*

Applications and the Internet (SAINT), Munich, Germany, 18-21 July 2011, pp.56-64.

- S. L. Lau and K. David, "Enabling context aware services in the area of AAC," in *Assistive and Augmentive Communication for the Disabled: In telligent Technologies for Communication, Learning and Teaching*, L. B. Theng, Ed. IGI Global, February 2011, ch. 6, pp. 159–192.
- S. L. Lau, I. König, K. David, B. Parandian, C. Carius-Düssel, and M. Schultz, "Supporting patient monitoring using activity recognition with a smartphone," in *Proc. 7th International Symposium on Wireless Communication Sys- tems (ISWCS), 2010*, Sept. 2010, pp. 810 –814.
- S. L. Lau and K. David, "Movement recognition using the accelerometer in smartphones," in *Proc. Future Network and Mobile Summit 2010*, Florence, Italy, June 16-18 2010, pp. 1–9.
- S. L. Lau, N. Klein, A. Pirali, I. König, and K. David, "Implementation of a user-centric context-aware playground," in *Proc. Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2009 (WowKiVS 2009)*, vol. 17. Electronic Communications of the EASST, 2009.
- S. L. Lau, N. Klein, A. Pirali, I. König, O. Drögehorn, and K. David, "Making service creation for (almost) everyone," in *Proc. ICT-Mobile Summit 2008*, Stockholm, June 2008.

References

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 66–75, January 1991.

[2] B. N. Schilit, N. I. Adams, and R. Want, "Context-aware computing applications," in *Mobile Computing Systems and Applications*, Dec 1994, pp. 85–90.

[3] D. Zhang, B. Adipat, and Y. Mowafi, "User-Centered Context-Aware Mobile Applications - The Next Generation of Personal Mobile Computing," *Communications of the Association for Information Systems*, vol. 24, no. 3, 2009.

[4] B. K. Hensel, G. Demiris, and K. L. Courtney, "Defining obtrusiveness in home telehealth technologies: A conceptual framework," *Journal of American Medical Informatics Association*, pp. 428–431, 2006.

[5] MATRIX, "MATRIX - Middleware-plattform für die realisierung internetbasierter telemedizinischer dienste," available online at http://www.forschungsprojekt-matrix.de/, last checked on 1st July 2011.

[6] VENUS, "VENUS - gestaltung technisch-sozialer vernetzung in situativen ubiquitären systemen," available online at http://www.uni-kassel.de/einrichtungen/iteg/venus/, last checked on 1st July 2011.

2 User-centric context aware system

This chapter gives an overview of context awareness and the important concepts as well as terms related to the design and realisation of a user-centric context aware system. This overview serves as a basis for the focus and proposal of this thesis.

2.1 Context and context awareness

The goal of this work is to investigate suitable approaches that can be applied in the development of a user-centric context aware system. Before we introduce the concept of such a system, let us first take a look at the meaning of *context* and *context aware*. Context, according to Schilit et al. [1], is defined as location, identities, nearby people and objects, and changes to those objects. This was followed by subsequent work that further investigated the definition of context, context awareness and its potential applications. Hull et al. viewed context as different aspects of the current situation of the user [2]. Researchers at the University of Kent referred to context as the user's location, environment, identity of people around the user, time and temperature [3] [4].

Many of these definitions were given either as synonyms for context or by example, i.e. they were defined as aspects of information needed for their prototypes or applications. A more widely accepted and used definition for context was later given by Dey and Abowd [5]:

Definition 2.1.1: Context

> Context is any information that can be used to characterise 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.

The definition was not intended to specify how context should be modeled in an implementation. It gives a generalised view on the idea of context. We view this abstract but useful definition as a general basis for all potential context aware applications. The evolution of context in the work after Dey has given a clearer picture how context can be applied in an implementation. Henricksen pointed out in [6] that Dey and Abowd did not clarify what was meant by *"the situation of an*

entity". "The situation" can be seen as a complex but definable concept. The Oxford online dictionary defines situation[1] as follows:

Definition 2.1.2: Situation

1. a set of circumstances in which one finds oneself; a state of affairs
2. the location and surroundings of a place
3. *formal* a position of employment; a job

The first two definitions of the word *situation* can be applied to context awareness. For a person, his situation can refer to what he is doing, where he is, and his condition at that particular moment. The situation information of a system, both software and hardware, can represent its current state, executable functions or even system-related information. In other words, context provides a mean for human users and systems to understand relevant information around them.

In some of the earlier research, contexts were mainly used to reveal further information for desired automatic change of system behaviour. For example, the Active Badge system [7] allows the receptionists to see a user's possible last detected location. Schilit et al. [1] demonstrated in their Palo Alto Research Center Incorporated Tab (PARCTAB) system, a context aware application that is able to present information to users based on proximity to services. Devices can be turned on or reconfigured according to the location of users and selected services can be executed automatically. In these examples, users can see which contexts have been recognised by the application (e.g. location of a user). At the same time, devices and services were initialised and executed based on the obtained contexts. Such automatic execution of devices and services is known as service adaptation. Since the adaptation takes place due to the obtained contexts or their changes, we can define a context aware system as follows:

Definition 2.1.3: Context aware system

A context aware system is a system that delivers and understands the available contexts perceived from the users and the surrounding through the use of sensor information. It also performs the appropriate service adaptations based on these contexts and their changes.

If the definition of context is adopted as abstract as it is, the information measured and provided by a sensor is considered as a context. The interpreted information based on this sensor measurement is also a context. In our work, we intend to draw some distinction between these different types of contexts. We propose to use three different levels of abstraction to categorise contexts. The abstraction levels can be seen as a way to view context from a computational point of view [8]. The first hand information obtained from a sensor device can be referred to as raw data. The

[1] Available online at
`http://oxforddictionaries.com/definition/situation`, last checked 1st November 2010.

processing of this raw data produces low level context. The low level context can be further processed to obtain high-level context. The process is illustrated in Figure 2.1 on page 9.

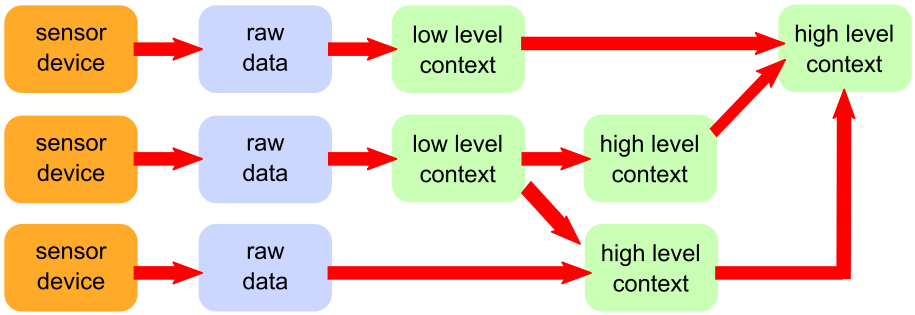


Figure 2.1: Three levels of abstraction for contexts.

We regard a low-level context as a direct interpretation of the information obtained from a sensor source. It gives a semantic meaning to the obtained value in order to allow further usage of this context. For example, the temperature sensor gives a reading of the voltage potential differences that represent the current temperature of the object or environment it measures. Given the corresponding calculation to this value, one can then obtain the current temperature with a desired unit, such as 23°C.

The third abstraction level of context is called the high-level context. It expresses the information that is usually interpreted from one or more low-level contexts. For example, a person may regard the surrounding temperature 23°C as warm. One can also conclude that a person is busy, when he is located in his office while his computer is turned on and is currently not idle. Usually, high-level contexts are semantically understandable and are implicitly perceived by human automatically. In context awareness, the computers are expected to be able to produce and use these contexts. In other words, the computers can potentially "understand" what and how a human thinks and perceives.

To the best of our knowledge, the distinctions and definitions for raw data, low-level contexts and high-level contexts are not often discussed in the literature. Nevertheless, these distinctions can provide designers and users of context aware systems a better understanding of the required contexts in a desired implementation and usage. The abstraction levels are useful to allow human users to understand what a system has processed and interpreted. For a computing device, the different abstractions are basically still strings, where these strings are results from the processing of the sensor data [8].

Besides the three abstraction level of contexts, the concept of contexts can also be understood by the categorization of the aspects of context. For example, in [1], Schilit et al. mentioned that three important aspects of contexts are *"where you are, who you are with, and what resources are nearby"*. Mayrhofer adopted and

extended the definition of Dey by categorizing contexts into different aspects such as geographical, physical, organizational, social, user, task, action, technological and time [9]. Sigg presented a more complete picture on the aspects of contexts by naming 14 different aspects of contexts [8]. He grouped these 14 aspects into 5 main aspects, which are time, location, constitution, environment and identity. The aspects of context help us to have a clearer picture how contexts are defined and applied in a context aware system. At the same time, it also clarifies the ambiguity that may occur when the definition of Dey is adopted.

The process to obtain low- and high-level contexts from sensor data is commonly known as context modeling. A context model defines contexts that are understandable by machines and users. Strang and Popien [10] presented a summary on most relevant context modeling approaches. These approaches were key-value, markup scheme, graphical, object oriented, logic based and ontology based models. Though the authors concluded that ontology based models are most expressive and suitable approach for their requirements, we regard the choice of context modeling approach is a domain and application specific. One needs to analyse his own application and requirement in order to identify the most suitable modeling approach. There are cases where an ontology is required for its expressiveness, but there are also cases where simple key-value implementations will suffice. To achieve the balance between user control and automation, the selection of a user accessible and comprehensible context model are factors to be considered.

Within the scope of this thesis, we focus on a specific aspect of context, which is the activity of a user. The expected service adaptation in a context aware system is usually dependent on what a user is doing and his situation at a given time. Besides the context of time and location, this aspect of context is seen as equally important in the context aware environment because a user is constantly performing a certain activity[2].

The word "activity[3]", according to the online Oxford dictionary , is defined as follows:

Definition 2.1.4: activity

1. a condition in which things are happening or being done.
2. busy or vigorous action or movement.
3. an action taken in pursuit of an objective.
4. a recreational pursuit.
5. the degree to which something displays its characteristic property or behaviour.

For the use of activity recognition in context awareness, the first three definitions can be seen as appropriate. The concept of *activity* describes things that are hap-

[2]technically, *no activity* is still seen as a valid context regarding the activity state of a user

[3]Available online at `http://www.oxforddictionaries.com/definition/activity`, last checked 9th June 2010.

pening around a person. By including the second and third definitions, the word activity extends the concept to additional details of a given activity. These details include the involved actions or movements and possible description of the objective for the occurrence of a given activity.

2.2 Context aware applications and systems

When we look back at the beginning of and the development of context awareness since two decades ago, there had been a lot of important investigations and effort in bringing the vision into working prototypes and applications. The Active Badge system [7] and the PARCTAB work [1] are two of the earliest context aware applications. A similarity among these earliest applications is the focus on location-aware adaptations and functions. In other words, the location information is the main contexts considered and applied. These systems are commonly known as location-based services today. Another common demonstration of location-based context aware application is a tour guide system. As tourists move around different attractions in a town, the tour guide can make use of the location context to present useful tourist information of different nearby highlights. Some examples are the work of [11] and [12]. Currently, there are already a number of emerging location-based services such as FourSquare[4] and Google Latitude[5] that are made available as products and services for the masses.

As mentioned by Schmidt et al. [13], there are more contexts than location. This can be observed in later work after the end of the 1990s. For example, the CybreMinder system [5] is a context aware application that sends out reminders based on time, location and situational contexts. Several other domains that have adopted the context aware approaches are smart homes [14], personalization [15] and health care [16, 17]. Nevertheless, regardless of how these investigations have selected the relevant contexts for the respective application domains, we can conclude that the choices and types of the selected contexts are consistent with the definitions of context and context aware system mentioned earlier on page 7 and 8.

There are also different context aware frameworks proposed in the past years. These frameworks include the Context Toolkit [18], the Context Management Framework (CMF) [19], the Context Broker Architecture (CoBrA) [20], the Context aware Sub-Structure (CASS) middleware [21] and the Service-Oriented Context Aware Middleware (SOCAM) [22]. As we analyse and observe these different context aware frameworks, there are a number of similarities between them. These systems may use different names or categorizations of their functions, but it is possible to summarise them in a generalised layer framework for understanding purposes. This framework, as illustrated in Figure 2.2, gives an overview of the core functions of the different roles and components found in a context aware system.

[4] Available online: http://www.foursquare.com/, last checked 1st August 2010

[5] Available online: http://www.google.com/latitude, last checked 1st August 2010

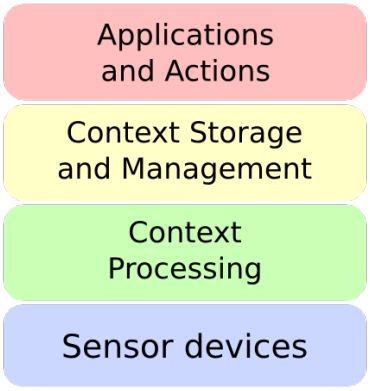


Figure 2.2: A generalised context aware layer framework.

The first layer is where the sensors are found. The sensors are responsible for the sensing of information that can be used to derive usable contexts. Commonly used sensors are the physical sensors, such as temperature, barometer or accelerometer. This type of sensor information usually provides raw data or low level contexts. Besides that, it is also possible to extract information from software and hardware and offer them via virtual sensors. Examples of virtual sensor information are personal calendar entries and smartphone call status. The contexts provided by virtual sensors are generally low level or high level contexts.

The processes that derive contexts are found in the second layer. Context processing includes a number of functions, such as filtering, context learning, context interpretation, context reasoning and prediction. Many context awareness related research projects focus in this layer. On top of context processing is the context storage and management layer. It deals with the issues on how the derived and produced contexts are stored and managed. Components, such as context repository and context broker, are found here. The final layer utilises components from the second and third layers for potential applications and actions. The available contexts for a specific application domain may be used by respective applications for display and adaptation purposes. It is also possible that a context aware application automatically invokes desired actions that are triggered by context-changes.

Currently, a conventional system usually requires human user input in order to provide appropriate services that satisfy the user's needs. In cases where service wishes are repetitive, there are systems that *"remembers"* the wishes, so that at a recurrence of these wishes, the system can provide the functions in an automatic manner. Current conventional systems usually use two direct methods to automate certain processes. Firstly, they remember the necessary settings at the point where the users last used the system. Typical examples are like radio or entertainment systems that remember last played station or song. A basic assumption made here is that the user wishes to continue where he was in a previous usage. Secondly, the systems support the utilization of more complex settings by allowing users to

program or customise their service behaviours. By stating and defining how, what and when the system should perform the desired functions, a system can execute these service wishes as defined. This ability that enables a system to act and respond according to pre-defined instructions is commonly known as adaptivity. In our work, we view this ability as passive adaptivity, because such ability requires direct human input and intervention in order to achieve and enable potential adaptation.

The vision of context aware computing intends to bring such adaptation and automation to the next level. Instead of providing just passive adaptivity, a context aware system aims to provide active adaptivity. In other words, the service wishes and behaviours should be recognised by the system in order to learn and recognise them correctly. Once this is done, the system can then provide appropriate adaptations that correspond to the user's needs. There should be a minimum amount of direct input to enable this ability.

2.3 User-centric context aware system

We observed that many of the past investigations were made strongly based on the design and thoughts of the system developers. The designers and developers were responsible for the assumptions and decisions set on behalf of the potential users. The desired prototypes and systems were then built based on these assumptions and decisions. While this is perhaps unavoidable for most of the cases, if the issue is addressed and considered from the beginning of design and development of a context aware system, the end product may be more likely to be well received by the designated users.

In our vision we foresee a user-centric environment. The complexity and technicality should be appropriately reduced and hidden, if not removed, so that users can have the necessary control of the system without being overwhelmed. This vision has resulted as the idea of a user-centric context aware system. In this thesis, we define the term "user-centric context aware system" as:

Definition 2.3.1: User-centric context aware system
A context aware system that is designed and developed with the users and their needs placed at the centre.

Based on this definition, we address the following design requirements for a user-centric context aware system:

- Users and their needs should be taken into consideration during the design of a context aware system.
- Users should be empowered to be able to use and to have control over the system and its behaviours.
- Users should be comfortable with the deployment and usage of the system.

- The system should deliver reliable performance and results, so that users do not relinquish it just because it fails to deliver.

The first requirement is usually considered in many past investigations. Most of the investigations have proposed solutions that automate tasks and functions on behalf of the users upon context changes. Nevertheless, there are also approaches where systems automatically provide users with information and service adaptations that are relevant only to the opinion of the developers [23]. The second requirement is however not always found [24]. In investigations such as [1], [12] or [25], the proposed solution and applied methods for context models are static in nature. There are also approaches that react fully automatic on behalf of the users. The users are not provided with the means to alter the provided adaptation possibilities. If there is an appropriate method that can at least give users a better understanding and overview of what the context aware system is going to perform, it may reduce the chances where users feel a lack of control.

The third requirement takes users' acceptance and comfort for a context aware system into consideration. A context aware system can be designed in such a way where multiple sensors and devices work seamlessly to acquire useful contexts. If the system utilises body-word sensors, the user is required to wear and carry these sensors throughout the whole day. He may likely to reject the system if the setup of the sensors and devices is troublesome or makes him feel uncomfortable by wearing them, or if both sensors and devices need frequent replacement because they run out of battery. Similarly, the fourth requirement also emphasises on user experience from the perspective of performance. A context aware system should deliver reliable performance and results that fulfill the promised features it claims to deliver. If it creates mistakes and frustration too regularly, the users will give up using the system.

Summarizing the third and fourth requirements, one important factor we want to highlight is the possibility for a context aware system being obtrusive. The word *obtrusive* is defined in the online Oxford dictionary [6] as follows:

Definition 2.3.2: obtrusive
noticeable or prominent in an unwelcome or intrusive way.

One potential hindrance for a user-centric context aware system is the obtrusiveness of the system. A system can be obtrusive in the usage and control of its functions. Complicated interfaces or control mechanisms may cause the users to feel frustrated. Devices can also be obtrusive in different ways, such as the wearing and placement of multiple sensor devices, inconveniences caused by slow response time or short battery life of a device, as well as the control and management of the devices that require constant user attention and intervention. In other words, the obtrusiveness factor in a context aware system brings inconvenience to the users or

[6] Available online at http://www.oxforddictionaries.com/definition/obtrusive, last checked 31th May 2011

requires them to make changes to their normal routines and habits.

The proposed vision of Mark Weiser had already emphasised on this factor. In his paper [26] he used the expressions "invisible" and "fade into the background" to describe how unobtrusive technologies are necessary in a future context aware environment. We believe an acceptable balance needs to be and can be obtained. On the one hand, the system should be "invisible" to the users so that no unnecessary interaction or direct user input is required. On the other hand, users should not feel lost amidst these "invisible" technologies. This vision motivates us to formulate a user-centric context aware system that emphasises on the use of techniques and approaches that are unobtrusive and user-centric.

2.4 Activity Recognition

As mentioned in the Section 2.1, this thesis focuses on the activity context of a user. Activity context can be seen as a subset of situation information, because it explains what a user is doing at a certain time. The activity context can be further broken down into more detailed categories. An activity is usually time and location dependent. A user's action or movement can also be grouped as his activity. Since activities are not always directly measurable, activity recognition techniques are applied to enable the acquisition of users' activity contexts. We define the word *recognition*, based on the definition provided by the online Oxford dictionary [7], as follows:

Definition 2.4.1: recognition

> The action or process of recognizing or being recognised, in particular the identification of a thing or person from previous encounters or knowledge.

Similarly, a context aware system that provides activity recognition is able to detect a person's current action or movement based on the available information. This information is usually obtained from the available sensors that are placed either on a user and/or around him in the environment where he is situated. The selected activity recognition is responsible to find correlations and relationships in these sensors data in order to discover the corresponding activity contexts. Examples of sensors used in past investigations for the area of activity recognition are accelerometer, microphone, camera, heart rate belt or Radio-frequency identification (RFID) tags. The types of desired activities related contexts include movements, tasks, locations and presence information (such as busy, available or away). When we analyze these investigations, we can categorise these sensors into three big groups according to their designated deployments:

[7] Available online at http://www.oxforddictionaries.com/definition/recognition, last checked 31th May 2011

- ***Wearable sensors***
  In the first category a user is usually equipped with one or more devices, placed at different parts of the body. Each device may have more than one sensor built-in, together with the necessary processing and communication modules. The processing modules enable simple or even intensive computation and manipulation of sensor data. The communication modules are responsible to transmit the obtained and processed sensor data to a remote device for further processing and storage tasks.

  The wearable sensors are responsible to record sensor values and their changes, which should generally correlate to the activities undertaken by the wearer. By applying appropriate algorithms, one can find useful information from these sensor values that can eventually be used to detect the corresponding activities.
- ***Environmental sensors***
  It is also possible to recognise activities in a given place such as a room. Sensors such as camera, microphone and RFID tags are deployed to provide implicit information that can be related to the activities carried out by the occupants at a given place. Examples of the activity information range from location of the occupants to the tasks a person is currently doing at his desktop. Such examples are found among the typical scenarios mentioned in the related investigations.
- ***Combination of both wearable and environmental sensors***
  There exists investigations that combine both of the above two categories for designated activity recognition. Sensor information from both type of sensors is aggregated and analysed to produce activity and other useful contexts for potential further usages and adaptations.

Similar to the abstraction levels for contexts, we group and define activities in three different categories, as shown in Figure 2.3 on page 17. The first category of activities consists of **gestures and movements**. It can be seen as the simplest form of activities that can be useful for potential context aware adaptation. Gestures and movements are usually short and possibly repetitive. The recognition of gestures and movements can be applied in areas such as recognizing sign languages [27] and intuitive human-computer interface input [28] [29]. It can also be the input information for the second category of activity - **basic activities**.

Basic activities can be seen as activities that involve different combinations of gestures and movements. For instance, a sequence of repeating steps can indicate that a person is walking or running, depending of the speed of the steps. This type of activities last longer than gestures and movements, ranging from seconds to minutes or hours. We view these basic activities as important information, because we perform them in our daily lives. Once basic activities are recognised, they can be used to derive **specific activities**.

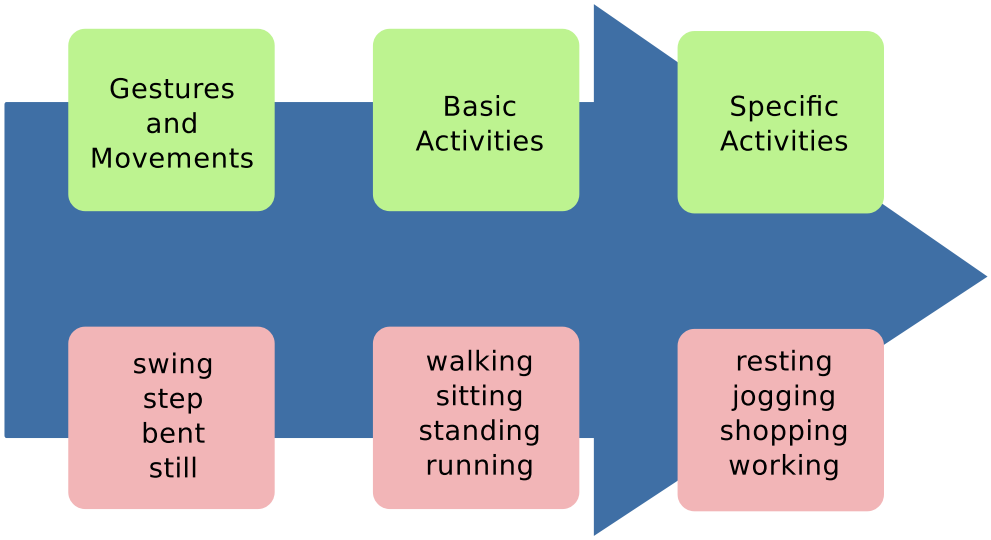


Figure 2.3: Different levels of activities.

We define specific activities as high level descriptions of task and event one performs at a given time and situation, i.e. working, jogging, shopping, sightseeing or relaxing. As an example, if a context aware system recognises that a person is walking, it may be possible to use additional information to derive whether the person is currently sightseeing or shopping. For the former, the system may need to obtain his calendar information to know he is on holiday and he is walking around a tourist location away from his home. For the latter, the person should be located in a place where people usually shop, such as a supermarket or a shopping centre.

The first two categories of activity are normally derivable via sensor data. Once distinct body motions are detectable, a system may be able to tell some gestures or basic activities apart. Depending on the available sensors and recognition approaches, basic activities can be recognised based on gestures and their transitions, or they can also be directly modeled and detected. For the last category one needs to model the possible relationships between the available context information that may help to reveal a user's specific activity at a given time.

This chapter reviewed and refined the definitions and concepts that were relevant to the thesis. Based on these definitions and concepts, the proposal of a user-centric context aware system was presented. As users' needs and acceptance were set as requirements, we highlighted the need to include unobtrusive approaches in the design and development of a user-centric context aware system. One particular area, selected as the focus in this thesis, was activity recognition. In the next chapter, we discuss the idea of activity recognition using unobtrusive sensor devices.

References

[1] B. N. Schilit, N. I. Adams, and R. Want, "Context-aware computing applications," in *Mobile Computing Systems and Applications*, Dec 1994, pp. 85–90.

[2] R. Hull, P. Neaves, and J. Bedford-Roberts, "Towards situated computing," in *ISWC '97: Proceedings of the 1st IEEE International Symposium on Wearable Computers*. Washington, DC, USA: IEEE Computer Society, 1997, p. 146.

[3] P. J. Brown, "The stick-e document: a framework for creating context-aware applications," in *Proceedings of EP'96, Palo Alto*. also published in it EP–odd, January 1996, pp. 259–272.

[4] N. S. Ryan, J. Pascoe, and D. R. Morse, "Enhanced reality fieldwork: the context-aware archaeological assistant," in *Computer Applications in Archaeology 1997*, ser. British Archaeological Reports, V. Gaffney, M. van Leusen, and S. Exxon, Eds. Oxford: Tempus Reparatum, October 1998.

[5] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*, 2000.

[6] K. Henricksen, "A framework for context-aware pervasive computing applications," Dissertation, School of Information Technology and Electrical Engineering, The University of Queensland, September 2003.

[7] R. Want, A. Hopper, V. Falc ao, and J. Gibbons, "The active badge location system," *ACM Trans. Inf. Syst.*, vol. 10, no. 1, pp. 91–102, 1992.

[8] S. Sigg, "Development of a novel context prediction algorithm and analysis of context prediction schemes," Dissertation, University of Kassel, 2008.

[9] R. Mayrhofer, H. Radi, and A. Ferscha, "Recognizing and predicting context by learning from user behavior," *Radiomatics: Journal of Communication Engineering, special issue on Advances in Mobile Multimedia*, vol. 1, no. 1, May 2004.

[10] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *First International Workshop on Advanced Context Modelling, Reasoning And Management, Nottingham, England*, September 2004.

[11] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: a mobile context-aware tour guide," *Wirel. Netw.*, vol. 3, no. 5, pp. 421–433, 1997.

[12] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, "Developing a context-aware electronic tourist guide: some issues and experiences," in *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2000, pp. 17–24.

[13] A. Schmidt, M. Beigl, and H.-W. Gellersen, "There is more to context than location," *Computers & Graphics*, vol. 23, no. 6, pp. 893 – 901, 1999.

[14] N. Roy, A. Roy, and S. K. Das, "Context-aware resource management in multi-inhabitant smart homes: A nash h-learning based approach," in *PERCOM '06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 148–158.

[15] M. Sutterer, O. Droegehorn, and K. David, "Upos: User profile ontology with situation-dependent preferences support," in *ACHI '08: Proceedings of the First International Conference on Advances in Computer-Human Interaction*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 230–235.

[16] J. E. Bardram and N. Nørskov, "A context-aware patient safety system for the operating room," in *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*. New York, NY, USA: ACM, 2008, pp. 272–281.

[17] A. Matic, P. Mehta, J. M. Rehg, V. Osmani, and O. Mayora, ""aid-me: Automatic identification of dressing failures through monitoring of patients and activity evaluation"„" in *4th International Conference on Pervasive Computing Technologies for Healthcare 2010 (Pervasive Health 2010)*, March 2010.

[18] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Hum.-Comput. Interact.*, vol. 16, no. 2, pp. 97–166, 2001.

[19] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Keränen, and E.-J. Malm, "Managing context information in mobile devices," *IEEE Pervasive Computing*, vol. 2, pp. 42–51, 2003.

[20] H. Chen, T. Finin, and A. Joshi, "An Intelligent Broker for Context-Aware Systems," *Adjunct Proceedings of Ubicomp 2003*, pp. 183–184, October 2003, (poster paper).

[21] P. Fahy and S. Clarke, "Cass – a middleware for mobile context-aware applications," in *Workshop on Context Awareness, MobiSys*, 2004.

[22] T. Gu, H. K. Pung, and D. Zhang, "A service-oriented middleware for building context-aware services," *J. Network and Computer Applications*, vol. 28, no. 1, pp. 1–18, 2005.

[23] L. Barkhuus and A. Dey, "Is Context-Aware Computing Taking Control Away from the User? Three Levels of Interactivity Examined," in *In Proceedings of Ubicomp 2003.* Springer, 2003, pp. 149–156.

[24] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Dartmouth College, Hanover, NH, USA, Tech. Rep., 2000.

[25] D. Siewiorek, A. Smailagic, J. Furukawa, N. Moraveji, K. Reiger, and J. Shaffer, "SenSay: A Context-Aware Mobile Phone," in *ISWC '03 Proceedings of the 7th IEEE International Symposium on Wearable Computers.* IEEE Computer Society, 2003, pp. 248–249.

[26] M. Weiser, "The computer for the 21st century," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 3, no. 3, pp. 3–11, 1999.

[27] H. Brashear, T. Starner, P. Lukowicz, and H. Junker, "Using multiple sensors for mobile sign language recognition," in *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, 18-21 2003, pp. 45 – 52.

[28] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uwave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive Computing and Communications, IEEE International Conference on*, vol. 0, pp. 1–9, 2009.

[29] A. Ferscha, S. Vogl, B. Emsenhuber, and B. Wally, "Physical shortcuts for media remote controls," in *Adjunct Proceedings of the 2nd International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN 2008)*, January 2008, p. 8.

3 Activity recognition system using unobtrusive sensor devices

In this chapter, the idea of activity recognition using unobtrusive sensor devices is introduced. As mentioned in the previous chapter, obtrusiveness of a system and its devices can be a hindrance for the system to be accepted and used by the designated users. For such cases, we need to identify approaches and devices that are less or unobtrusive and at the same time deliver equivalent performance like the obtrusive approaches. This chapter first presents an overview on the related work in activity recognition. This is followed by the elaboration of the proposal to use a smartphone as an unobtrusive device for activity recognition. Finally, a proof of concept application for activity recognition is introduced before the chapter is concluded.

3.1 Related work

There are numerous techniques to recognise user activities. Popular methods make use of different types of sensors such as microphone and camera, body-worn sensors as well as wireless sensor nodes in the environment. The different types of sensors require a different kind of techniques for the desired recognition. The following reviews elaborate the selected work that have investigated the use of different sensors and techniques in this area.

3.1.1 Sensor(s) used in activity recognition

The earlier work investigated the possibilities of using body-worn sensors for activity recognition. In the late 90s, researchers in Massachusetts Institute of Technology (MIT) [1] explored gestures and activity recognition techniques based on video [2] and audio data [3]. For example, in [2] one can use a small camera, installed on a cap, to track hand movements to interpret the performed sign language signals. With the help of a microphone and a Personal Digital Assistant (PDA), the authors in [3] processed audio streams to detect speeches or even proximity of several users. Video-based approaches are also found in premises monitoring. In the work of [4], video was used to detect Activities of daily living (ADL) for the care of elderly

people. The use of audio and video as sensor data has a problem, where installation and monitoring using microphones and video cameras often raise the issue of trust and privacy.

Besides audio and video, other sensors have been tested for potential activity recognition. Body-worn accelerometers are seen as a popular choice as the main sensor for activity recognition. Work such as [5], [6], [7], [8] and [9] demonstrated that the use of dedicated accelerometers can provide good results in the area of activity recognition. For example, in [5], the recognition accuracy was 84.26 % using five biaxial accelerometers on different body parts and a C4.5 [10] classifier. Similarly, Kern et al. [6] achieved accuracy up to 84 % using 12 triaxial accelerometers and a Naïve Bayes classifier. For example, Ravi et al. [7] obtained accuracy up to 99.82 % using a dedicated triaxial accelerometer mounted on a hoarder board placed near the pelvic region of a test person and meta-level classifiers.

The different investigations have one thing in common - a single accelerometer or multiple accelerometers were placed on different parts of the body, either wired or wireless, and users were required to perform designated movements. The recorded accelerometer values were processed and the resulted features were evaluated using classification algorithms for potential recognition. The accelerometer can be seen as an appropriate unobtrusive sensor device. It is small and uses relatively small amount of energy. However, these previous investigations focused more on offline recognitions and did not propose how suitable is the usage of accelerometer together with a mobile computing device in a real-time recognition system. The usage of five up to twelve accelerometers was also seen as a rather obtrusive approach, since users need to first wear all the accelerometers on the designated positions in order to make use of the proposed systems.

The ideas were expanded with the inclusion of additional sensor information. For example, in [11] a heart rate monitor was coupled with data, taken from five accelerometers, to detect physical activities. Researhers at the Intel Research in Seattle and the University of Washington used the Multi-modal sensor board (MSB) that had accelerometer, audio, temperature, Infrared (IR)/visible/high-frequency light, humidity, barometric pressure and digital compass [12]. They investigated activity recognition classification of physical activities with multiple MSBs. The group in [13] used a triaxial accelerometer together with a wearable camera to recognise human activity. The combination of these two sensors was used to recognise whether a user was walking forward, walking backward, standing, sitting, turning or taking the elevator. The inclusion of additional sensors can help improve the recognition of specific activities, particularly when the information from an accelerometer sensor was insufficient to correctly recognise them. However, one should only choose the sensors that are useful for the designated recognition tasks. Higher amount of sensor data will potentially increase the processing load and may utilise higher amount of energy.

Recently, the three-dimensional accelerometer integrated in smartphones was also

investigated as a potential sensor for movement recognition. In [14], the accelerometer of a Nokia N95 was used as a step counter. The results showed that such smartphones can provide accurate step-counts, comparable to some of the commercial and dedicated step counter products, provided the phone is firmly attached to the body. The DiaTrace project [15] uses a mobile phone with accelerometers for physical activity monitoring. The prototype obtained accuracy of >95 % for activity types of resting, walking, running, cycling and car driving. Brezmes et al. have also used the acceleration data collected with a Nokia N95 with K-nearest neighbour algorithm to detect common movements [16]. We regard this approach as suitable, since a smartphone is getting more and more common and can be used as an unobtrusive device for the purpose of activity recognition. However, the above investigations did not compare the respective applied methods with other classification algorithms, used in related investigations with one or multiple dedicated accelerometers. There was also no investigation to compare performance related issues, such as recognition speed and influence on the battery life of the smartphones.

There are also investigations in integrating sensors on garments for various activity recognition tasks. Mattmann et al. from the Eidgenössische Technische Hochschule Zürich (ETH Zürich) used tight-fitting clothing and strain sensors to measure body posture [17]. The SMArt SHirt (SMASH) [18] uses accelerometers integrated in the garment for potential rehabilitation applications, such as movement and posture rehabilitation for children. The Konnex unit has a Texas Instrucment's Mixed Signal Processor 430 (MSP430) microprocessor and performs the designated pattern recognition tasks. The SMASH Gateways is designed to acquire sensor data from the accelerometers and to perform feature extraction on these data before sending them to the Konnex unit. Cheng from the University of Passau used conductive textile based electrodes that are integrated in a garment to detect specific activities such as chewing, swallowing, speaking or sighing [19]. This capacitive-based sensing approach measures capacitance changes inside the human body that can give correlated information related to movements and shape changes of muscle, skin, and other tissues [19] . These investigations using smart garments are also seen as unobtrusive approaches. However, they are still prototypes and proof of concepts garments. It is not possible to obtain one in the commercial market for immediate usage and integration for everyday activities.

Apart from body-worn sensors, there are also approaches that utilise small sensors embedded in mobile devices and artifacts. The Technology Enabling Awareness (TEA) project investigated how multi sensor context awareness can be realised in a self-contained device [20]. A TEA device, consists of photo diodes, microphones, accelerometer, digital temperature sensor and touch sensor, as well as a microcontroller, can be attached to mobile devices for the use of activity recognition [21] [20]. Similar embedded sensor devices can also be integrated in normal daily life objects. For example, Active Badge [22] uses RFID for the recognition of the user's location. The MediaCup [23] is another example of a recognition technology

augmented non-computational artifact. The system can detect context changes such as temperature of the cup, keyboard activities (via keyboard clicking sound) and cup movements. The researchers in the Johannes Keppler University of Linz developed a cube with sensors such as accelerometer and gyroscope to be used as a remote control device with tangible user interface for set-top boxes [24]. These approaches are also considered as unobtrusive, as long as the sensor devices can be kept small and can be accepted by their potential users without requiring them to make big adjustments in their everyday lives.

3.1.2 Applications

A major area where activity recognition plays an important role is the health care. With the possibility to detect activities automatically and reliably, advance services such as remote patient monitoring or therapy can be offered. For instance, Jin et al. [25] investigated detection of human motion states using the SenseWear® PRO_2 sensor armband. They aim to use activity recognition for diagnosis of sleep disorder, screening of treatment effort as well as monitoring of motions and exercise prescription for patients with chronic disease. Jatobá et al. [26] from the University of Karlsruhe used a dedicated acceleration sensor to monitor physical activity of a patient. By combining Electrocardiogram (ECG), blood-pressure and physical activity, the developed system provides continuous monitoring of patients for the prevention or assisted therapy of cardiovascular diseases in their daily lives. In the investigation of Jehn et al. [27], the pedometer and accelerometers were used to assist the quantification of the 6-minute walk test (6MWT) performance for patients with Chronic heart failure (CHF).

The availability of activity contexts is also investigated and used in smart environment and assisted living areas. This specific area of interest intends to use the recognition to provide appropriate services and adaptations. The Gator Tech Smart House [28], located in Gainesville, Florida, is a project that aims to create assisting environments using sensors, actuators and services running on a middleware. Its goal is to realise homes with the ability to sense context information on the buildings and their residents, such as arrival of new mails in the mailbox, sleep pattern monitoring, contents in the refrigerator and floor that tracks location of the occupants and fall detection. Another smart home project is the Managing An Intelligent Versatile Home (MavHome) from the University of Texas of Arlington [29]. With the use of sensor information, the MavHome intelligent agent predicts next action of the inhabitant to automate the repetitive tasks for them. Activities were recorded and analysed to discover available patterns for the designated prediction tasks. The iDorm1 and iDorm2 projects from the University of Essex [30] used embedded agents to sense and recognise abnormal activities that take place in an environment such as a flat.

Other applications of activity recognition for smart environment and assisted liv-

ing include energy management, safety and elderly care. For example, Erickson et al. used wireless camera sensor nodes in a building to predict occupancy for the rooms [31]. With this occupancy information, the system automatic makes adjustments and controls the heating, ventilating and air conditioning (HVAC) systems installed in the rooms. Ferscha et al. proposed a system using a small sensor box with a built-in accelerometer to recognise three simple movements "sitting", "standing" and "walking" [32]. The different electrical appliances in the room are controlled via a management system by receiving different context information from the users and the environment. The researchers at University of Washington and Microsoft proposed a video surveillance system that analyses video images to detect new and unusual activities[33].

Another emerging application of activity recognition techniques focuses on how users interact with devices. The utilization of wireless controller that recognises human gestures as game controls have been popular since the last few years. Nintendo's Wii game console [1] as well as smartphones such as Apple IPhones, Nokia N-series and Android-based phones are some examples that took advantage of this possibility. The gesture-based game control has provided new experiences and new ways to interact with the supported games. Besides gaming, these applications can also provide more intuitive interaction between users and devices in a similar manner. As mentioned earlier, the research area of tangible user interfaces investigates how gestures and movements can be used for different purposes. For example, the uWave gesture recognition system supports devices such as Wii remote controller, custom-built micro-controller and smartphones. Applications tested with the uWave system include gesture-based authentication and Three Dimensional (3D) mobile user interface control. Agrawal et al. from the Duke University Durham presented the PhonePoint Pen system that turns a Nokia N95 smartphone into a hand-writing and drawing tool [34]. These investigations have demonstrated possibilities for users to interact with devices in a more natural and unobtrusive manner.

3.1.3 Recognition Techniques

The types of recognition techniques are usually dependent on the types of sensors and available data used in the respective investigations. Generally, the methods used can be categorised into two main groups. The first group is the supervised learning approach to train the given algorithm with labeled data to generate models that can be used for the designated recognition. The second group is known as unsupervised learning approach. It attempts to construct usable models without the need of having the data labeled. In other words, the unsupervised approach aims to discover available information within a set of unlabeled data that allows future recognition. Typically, both methods do not work on the raw data directly. The available measured data are first pre-processed to be transformed into features.

[1] `http://www.nintendo.com/wii`, last checked 1st August 2010.

Selected learning algorithms are then applied to analyse these features to draw useful observation and patterns from the transformed data. Further explanation on features and learning algorithms are presented in Chapter 4.

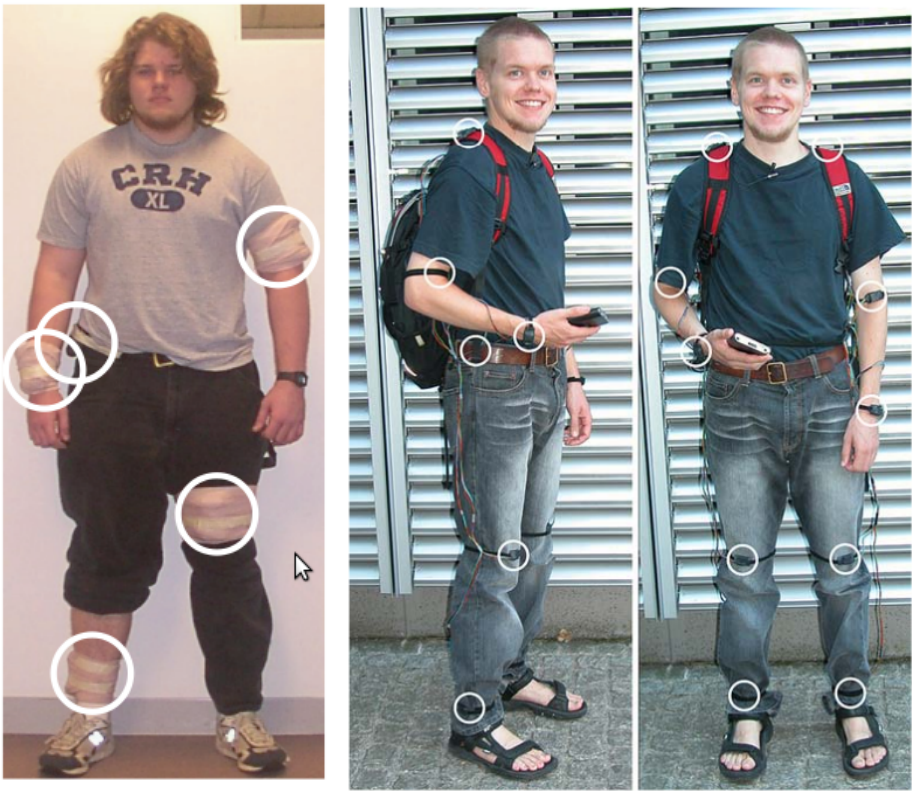
The supervised learning methods are considered as the most predominantly applied approach in the field of activity recognition [35] [36]. For example, simple but efficient base-level classification algorithms such as the k-nearest neighbour, decision tree, naïve Bayes and Bayesian Network were used in various investigations, for example [37], [5], [38], [7], [39], [40] and [41]. There are also investigations that used other base-level classification algorithms such as support vector machines (SVM) ([7], [42], [40] and [43]), Hidden Markov Model (HMM) ([12], [44] and [45]) and neural network based classifiers ([9], [46] and [47]).

It is also possible to combine more than one base-level classifier or to reuse the same base-level classifier in multiple iteration in order to improve the recognition accuracy. This approach, also known as meta-level classification, is also used in different previous activity recognition investigations. For example, the meta-level classifiers boosting and bagging techniques are used in investigations such as [48], [7] and [49]. It is shown that the meta-classifiers generally improve the recognition accuracies as compared to the respective single base-level classifiers. Lester et al. [12] has combined static classifiers with HMM to improve the intended recognition accuracies.

The unsupervised learning methods are not as common as the supervised approaches. A popular approach is to utilise clustering techniques to group similar patterns found in the data as possible activities. Clarkson and Pentland have used hierarchies of HMMs from audio and video data to perform unsupervised time series clustering on activities such as "walking down a sidewalk" and "at the video store". Huynh and Schiele have applied the use of multiple eigenspaces to enable unsupervised learning of basic activities [50] based on the measured triaxial accelerometer data. Minnen et al. [51] have shown that motifs (recurring patterns) found in data can be detected using unsupervised methods such as Symbolic Aggregate approXimation (SAX) technique developed by Lin et al.[52]. They have investigated efficient approaches for multivariate motif discovery applicable for different domains such as activity discovery using body-worn accelerometer and gyroscope data. In the investigation of Gu et al. [36], an unsupervised fingerprint-based algorithm has been proposed to recognise activities in a smart home environment using a wearable RFID system.

3.1.4 Summary

The above list of related work presented is by no means a complete list of all activity recognition investigations. Nevertheless, it gives us an overview of the various sensors, applications and techniques applied in this field. The common goal observed is evident - activity recognition plays an important role in the realisation of a context



(a) Bao and Intille (2004)[5] (b) Kern et al. (2007)[6]

Figure 3.1: Sensor placements in two activity recognition investigations using body-worn sensors.

aware system. As we have pointed out earlier, most of these investigations have not focused or researched on methods that are unobtrusive. On the contrary, some of the proposed approaches are not seen as practical and applicable for the everyday use. One of the solutions to this observation is to identify devices and techniques that are more likely to be accepted by a user. Obtrusiveness is one factor that we aim to address in this thesis.

3.2 Activity recognition using a smartphone

As the related work elaborated in the previous section, some of the popular activity recognition approaches have proposed the use of body-worn sensors. Different sensors are commonly placed at different parts of the body. We have selected the two following examples to show how the placement of sensors and corresponding devices was proposed. These examples are shown in Figure 3.1.

Bao and Intille (Figure 3.1a) used 5 sensor boards on different parts of the body such as arm, wrist, knee, ankle and waist [5]. Each sensor board consists of a biaxial accelerometer, four AAA batteries and a memory card for storage. As shown in

Figure 3.1b, Kern et al. investigated activity recognition using 12 body-worn triaxial accelerometers. Both investigations have shown that accelerometer-based activity recognition can give up to around 90 % accuracy. However, in order to enable the recognition of basic activities, the approaches suggest the use of a few sensors placed at fixed strategic positions depending on the targeted activities [6].

We contend that such approaches are obtrusive for a person [41]. In the investigation of Bao and Intille, the authors mentioned that some of the experiment participants have reported that they felt self conscious in public spaces. This was because the sensor devices used (see Figure 3.1a, though wireless and light (each weighed less than 120g), were visually noticeable. Choudhory et al. stated that the placement of sensors in multiple predefined locations can be quite obtrusive[53]. They contended this as a limitation for common activity recognition approaches using body-worn sensors [53].

There are also other people who have made similar observations on the same issue. Lester et al. suggested the use of a single sensor placement as a less obtrusive alternative [12]. Ferscha et al. discussed on this issue by saying the wearable multi-sensor solutions are very obtrusive since wired techniques are used and needed most of the time and users have to strap sensors with Velcro strips or even wear special suits for the intended recognitions [32]. We agree with these statements and observations because the chances of users to accept and use newer technologies are affected by how they perceive and accept the technology, including user friendliness of controls and interfaces as well as the obtrusiveness of the required devices and systems.

Instead of placing different sensors on the person's body for continuous monitoring, we propose to use unobtrusive and minimum number of devices. The person should not consciously feel intruded or disturb with the number of sensors and the placement of the sensors.

A smartphone can be seen as a potential unobtrusive sensor device. Currently, most smartphones in the market have multiple built-in sensors, such as accelerometer, proximity, Global Positioning System (GPS) and magnetometer. Particularly in the case where the person already owns a smartphone, it is not necessary for him to use additional device for sensor data collection. If there is such a demand in the near future, additional unobtrusive sensor devices can still be connected wirelessly to the smartphone for data collection, processing, transfer and even evaluation.

We propose the use of smartphones as an alternative to current body-worn sensor devices based on the following factors:

- The available sensors are built-in. As long as the desired context can be derived and recognised from the data of the built-in sensors, the users are not required to use external sensors in order to collect needed information. In cases where the need occurs, additional sensors and devices can be interfaced to smartphones to extend necessary sensors other than the ones built-in. The

flexibility and readiness of the smartphone as a sensor device are seen as advantages.

- The smartphones have many properties that enable context-aware related implementations. Most smartphones have relatively high processing power and sufficient memory for data processing tasks. They also contain more than adequate storage space for the storage of raw and computed data. The smartphones also provide communication possibilities that allow information exchange between user and external services. The smartphone itself can be seen as a small computing device with common connectivity integrated.
- A smartphone is likely to be with a user during his daily activities. It can be seen as a natural choice of an unobtrusive device. The chances of users feeling awkward or uncomfortable will be much lower as compared to approaches that affect the usage habits of the users.
- Most smartphones have also relatively long operation durations. For an average user, under normal usage patterns (some daily phone conversations and text messages), a smartphone should have at least a day's operation time before a recharge is required. With proper management for sensor data polling, whole day sensor data collection and processing may be achievable.

A smartphone plays five main roles for the tasks of activity recognition. It is a device that supports sensor, computing, storage, communication and user interaction. We can use the available built-in physical and virtual sensors to acquire information about the user and his environment. The acquired sensor data are stored for storage and further context processing steps. If necessary, the sensor information or the processed contexts can be sent to a remote server for further storage and processing. The communication channels offered by a smartphone range from short-range communication such as Bluetooth and Wireless Local Area Network (WLAN) to mobile broadband such as Universal Mobile Telecommunications System (UMTS).

The accelerometer of a smartphone is seen as a good candidate as sensor device for the desired activity recognition. It measures the acceleration of a person, which correlates to his movements. Many users usually carry a smartphone with them most of the time throughout a day [54]. If the measurable sensor information is able to be used to derive implicit information that reveal their contexts, such as their current activities and situations, the smartphone is an ideal all-in one device that complements the existing approaches.

3.3 Architecture for a user-centric context aware system

With the above ideas and concepts, we proposed an architecture that can be used to design a user-centric context aware system. The architecture uses a Service-oriented

Architecture (SoA) based design. The loosely coupling of components allows flexible deployment of context aware functions in the system. We elaborate the proposed user-centric context aware system using the CARMA, a prototype context aware application built for the German research project MATRIX [55] [49]. There are four main functional components defined in the proposed system, deployable and supported in a given Service Execution Environment (SEE). These components are illustrated in Figure 3.2:

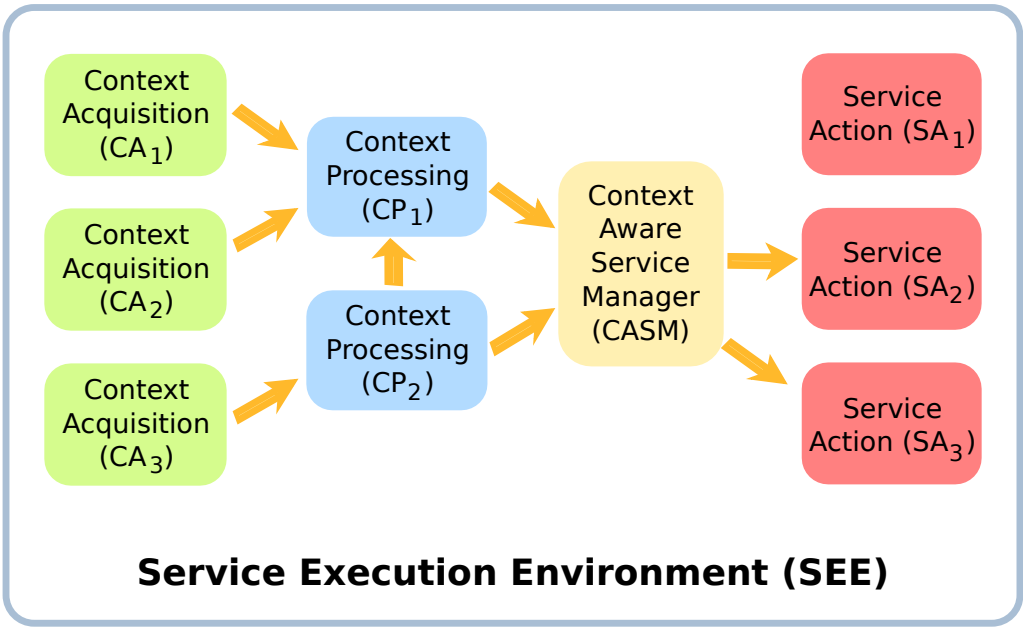


Figure 3.2: CARMA functional components.

The CARMA application is basically a combination of functional components with individual functions that complete the purpose of remote monitoring using activity recognition. The components have well-defined functions. There are four main functional categories of components, defined in CARMA. Each category is presented as follows:

- Context Acquisition (CA) Components
  The context acquisition components collect the sensor information from selected sensor sources. For the CARMA application, this acquisition process can take place on mobile devices, such as a smartphone and on stationary device such as a computer. The momentary sensor values are measured and sent to one or more designated destinations. Depending on the requirements for context processing, the sampling rate and method of data transmission may vary. At this stage, no manipulation of sensor data is expected.
- Context Processing (CP) Components
  We divide context processing into two categories. The first category is the

context interpreter. It is responsible to derive contexts from the available collected data. A context interpreter evaluates the collected data and produces contexts as results. In some literature, this interpretation process is also known as context reasoning or context inference. The output contexts can be low and high level contexts. Low level contexts are derived from the sensor data using simple techniques such as matching of key-value pairs. Context interpretation can also be performed by using more complicated approaches such as classification, clustering or ontology modelling. Classification and clustering based approaches usually require pre-processing techniques like filtering or feature extraction.
The second category of context processing is known as context prediction. It can be used to provide proactive adaptations by predicting future contexts. These contexts can help the context aware application to timely react. For example, a context aware application uses context prediction to obtain next location and activity of a user. The adaptation takes place as soon as he enters a new location. Pre-heating a room or a car automatically and timely is an example for context prediction. We group these two categories in context processing because they process given data based on available past knowledge to produce new contexts. The contexts produced by the CPs are essential in context awareness for adaptation.

- Service Action (SA) Components
  The third component in CARMA is known as the service action components. They are components designed to carry out tasks. For example, a user uses CARMA to keep watch of his grandmother at home while he is at work. He wishes to receive a notification when she stays at a place without movement for an unusual longer period of time. He can instruct CARMA to observe a number of contexts, such as her movement, location and duration of the same movements. At runtime, as soon as the interpreted contexts satisfy the above situation, a notification SA component is invoked.
  The SA components are not only limited to notification functions. An SA component can be an actuator for a certain device for a home automation system. A software component found on a normal desktop system can also be seen as an SA component. Therefore, a system can potentially have different types of SA components. They are only invoked when the obtained contexts satisfy a certain user- or system-defined condition.

- Context Aware Service Manager (CASM)
  This component is responsible to provide the desired adaptation according to the needs of the user. By observing the context changes provided by the corresponding context processing components, it triggers the designated service action components. A possible implementation of an CASM is to define context aware adaptations and services using rule-based policies. Using the scenario as an example, a doctor can define if-then rules to trigger a Short Mes-

saging Service (SMS) notification if the CP component recognises a specific movement of his patient that indicates potential risk for the latter.

The prototype application CARMA is designed to enable unobtrusive activity monitoring using a smartphone. Depending on the designated application scenario, the four types of components can either be solely deployed on the smartphone, or distributed on a smartphone and a remote server. For the latter option, we use the Context Server (CS) as a centralised infrastructure. It provides the storage and processing of sensor and context data obtained from one or more mobile devices. As shown in Figure 3.2 in page 30, the different CA components (CA_1 to CA_3) are responsible to collect sensor data. In our case, the CA component, deployed on a smartphone, delivers the acceleration data. This data are then sent to the CP component for further processing. The result of an CP component is sometimes used as an input for another CP component (see the relationship between CP_2 and CP_3 in Figure 3.2). In the same way, a SA component may also be used to trigger another CP component or even CA component(s) to obtain additional contexts. These possibilities and flexibilities are seen as attributes in the intended user-centric context aware system.

In this thesis, we will focus on creating suitable CA and CP components for the purpose of activity recognition in CARMA. A CA component is developed to provide the acquisition of acceleration data. The recognition tasks are implemented in an CP component, which processes the acceleration data from the CA component to recognise movements of a given user using the prototype application with a smartphone.

3.4 Questions to be answered

In this chapter, an introduction to activity recognition and the respective related work are elaborated. Based on the requirements and ideas proposed in Chapter 2 as well as the observations made based on the related work, we proposed to implement activity recognition using unobtrusive devices as a complement to the existing approaches. An architecture for a user-centric context aware system is introduced with the prototype application CARMA as an example how the designated system can be implemented.

Unlike previous investigations that have used multiple accelerometers to recognise both basic and specific activities (e.g. [9], [5], [6] and [11]), the CARMA application uses only a single triaxial built-in accelerometer in the smartphone. This raises some questions that this work investigates:

- What are the (potential) advantages and disadvantages of using the triaxial accelerometer as compared to dedicated accelerometers?

- Does a single triaxial accelerometer of a smartphone provide equivalent recognition accuracy as compared to the investigations with one or more dedicated accelerometers?
- What are the suitable recognition algorithms that can deliver the desired recognition?
- What are the suitable data pre-processing techniques that should be used for the designated recognition algorithm?
- How suitable is a smartphone as a sensor and a computing device for the purpose of activity recognition?

The following two chapters present the investigations performed within the scope of this thesis in applying smartphone as an unobtrusive sensor device for activity recognition. Chapter 4 describes the steps and techniques performed in order to prepare the data needed for the experiments. Chapter 5 and Chapter 6 present the experiments carried out and the evaluations of the respective results that provide answers to the above questions.

References

[1] A. Pentland, "Smart rooms, smart clothes," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 2, 16-20 1998, pp. 949 –953 vol.2.

[2] T. Starner, J. Weaver, and A. Pentland, "A wearable computer based american sign language recognizer," in *ISWC '97: Proceedings of the 1st IEEE International Symposium on Wearable Computers*. Washington, DC, USA: IEEE Computer Society, 1997, p. 130.

[3] N. Eagle and A. Pentland, "Wearables in the workplace: sensing interactions at the office," in *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, 18-21 2003, pp. 256 – 257.

[4] Z. Zhou, X. Chen, Y.-C. Chung, Z. He, T. Han, and J. Keller, "Activity analysis, summarization, and visualization for indoor human activity monitoring," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 11, pp. 1489 –1498, nov. 2008.

[5] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive 2004*, pp. 1–17, April 2004.

[6] N. Kern, B. Schiele, and A. Schmidt, "Recognizing context for annotating a live life recording," *Personal Ubiquitous Comput.*, vol. 11, no. 4, pp. 251–263, 2007.

[7] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," *American Association for Artificial Intelligence*, 2005.

[8] K. Van 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.

[9] J. Mäntyjärvi, J. Himberg, and T. Seppänen, "Recognizing human motion with multiple acceleration sensors," in *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, vol. 2, 2001, pp. 747–752 vol.2.

[10] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[11] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman, "Real-Time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in *Wearable Computers, 2007 11th IEEE International Symposium on*, 2007, pp. 37–40.

[12] J. Lester, T. Choudhury, and G. Borriello, "A practical approach to recognizing physical activities," in *Pervasive*, ser. Lecture Notes in Computer Science, K. P. Fishkin, B. Schiele, P. Nixon, and A. J. Quigley, Eds., vol. 3968. Springer, 2006, pp. 1–16.

[13] Y. Cho, Y. Nam, Y. Choi, and W. Cho, "SmartBuckle: human activity recognition using a 3-axis accelerometer and a wearable camera," in *Proceedings of the 2nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*. Breckenridge, Colorado: ACM, 2008, pp. 1–3.

[14] M. Mladenov and M. Mock, "A step counter service for java-enabled devices using a built-in accelerometer," in *CAMS '09: Proceedings of the 1st International Workshop on Context-Aware Middleware and Services*. New York, NY, USA: ACM, 2009, pp. 1–5.

[15] G. Bieber, J. Voskamp, and B. Urban, "Activity recognition for everyday life on mobile phones," in *HCI (6)*, 2009, pp. 289–296.

[16] T. Brezmes, J. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*. Salamanca, Spain: Springer-Verlag, 2009, pp. 796–799.

[17] C. Mattmann, O. Amft, H. Harms, F. Clemens, and G. Tröster, "Recognizing upper body postures using textile strain sensors," in *Proc. 11th International*

Symposium on Wearable Computers (ISWC07), 0 2007, pp. 29–36, recipient of the IEEE ISWC 2007 Best Paper Award.

[18] H. Harms, O. Amft, D. Favre, C. Liesen, D. Roggen, and G. Tröster, "Motion sensitive clothing," in *Adjunct Proceedings of Pervasive 2010 Conference*, 2010, paper with video.

[19] J. Cheng, O. Amft, and P. Lukowicz, "Active capacitive sensing: Exploring a new wearable sensing modality for activity recognition," in *Pervasive Computing*, ser. Lecture Notes in Computer Science, P. Floréen, A. Krüger, and M. Spasojevic, Eds. Springer Berlin / Heidelberg, 2010, vol. 6030, pp. 319–336.

[20] H. W. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in mobile devices and smart artifacts," *Mob. Netw. Appl.*, vol. 7, no. 5, pp. 341–351, 2002.

[21] K. V. Laerhoven, K. Aidoo, and S. Lowette, "Real-time analysis of data from many sensors with neural networks," in *In ISWC*. IEEE Press, 2001, pp. 115–123.

[22] R. Want, A. Hopper, V. Falc ao, and J. Gibbons, "The active badge location system," *ACM Trans. Inf. Syst.*, vol. 10, no. 1, pp. 91–102, 1992.

[23] M. Beigl, H.-W. Gellersen, and A. Schmidt, "Mediacups: experience with design and use of computer-augmented everyday artefacts," *Computer Networks*, vol. 35, no. 4, pp. 401 – 409, 2001.

[24] A. Ferscha, S. Vogl, B. Emsenhuber, and B. Wally, "Physical shortcuts for media remote controls," in *Adjunct Proceedings of the 2nd International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN 2008)*, January 2008, p. 8.

[25] G. H. Jin, S. B. Lee, and T. S. Lee, "Context awareness of human motion states using accelerometer," *J. Med. Syst.*, vol. 32, no. 2, pp. 93–100, 2008.

[26] L. C. Jatoba, U. Grossmann, C. Kunze, J. Ottenbacher, and W. Stork, "Context-aware mobile health monitoring: Evaluation of different pattern recognition methods for classification of physical activity," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, aug. 2008, pp. 5250 –5253.

[27] M. Jehn, A. Schmidt-Trucksäess, T. Schuster, H. Hanssen, M. Weis, M. Halle, and F. Koehler, "Accelerometer-based quantification of 6-minute walk test performance in patients with chronic heart failure: Applicability in telemedicine," *Journal of Cardiac Failure*, vol. 15, no. 4, pp. 334 – 340, 2009.

[28] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The Gator Tech Smart House: A programmable pervasive space," *Computer*, vol. 38, no. 3, pp. 50–60, 2005.

[29] D. J. Cook, M. Youngblood, E. O. Heierman, III, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome: An agent-based smart home," in *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA: IEEE Computer Society, 2003, p. 521.

[30] F. Rivera-Illingworth, V. Callaghan, and H. Hagras, "Automated discovery of human activities inside pervasive living spaces," in *Pervasive Computing and Applications, 2006 1st International Symposium on*, 3-5 2006, pp. 77 –82.

[31] V. L. Erickson, Y. Lin, A. Kamthe, R. Brahme, A. Surana, A. E. Cerpa, M. D. Sohn, and S. Narayanan, "Energy efficient building environment control strategies using real-time occupancy measurements," in *BuildSys '09: Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. New York, NY, USA: ACM, 2009, pp. 19–24.

[32] A. Ferscha, B. Emsenhuber, S. Gusenbauer, and B. Wally, "PowerSaver: Pocket-worn activity tracker for energy management," Innsbruck, Austria, pp. 321–324, September 2007.

[33] W. Lin, M.-T. Sun, R. Poovandran, and Z. Zhang, "Human activity recognition for video surveillance," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, 18-21 2008, pp. 2737 –2740.

[34] S. Agrawal, I. Constandache, S. Gaonkar, and R. R. Choudhury, "PhonePoint pen: using mobile phones to write in air," in *MobiHeld '09: Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*. New York, NY, USA: ACM, 2009, pp. 1–6.

[35] D. T. G. Huynh, "Human activity recognition with wearable sensors," Dissertation, Technische Universität Darmstadt, 2008.

[36] T. Gu, S. Chen, X. Tao, and J. Lu, "An unsupervised approach to activity recognition and segmentation based on object-use fingerprints," *Data Knowl. Eng.*, vol. 69, no. 6, pp. 533–544, 2010.

[37] K. Van Laerhoven, N. Kern, H.-W. Gellersen, and B. Schiele, "Towards a wearable inertial sensor network," in *Eurowearable, 2003. IEE*, 4-5 2003, pp. 125 –130.

[38] K. S. Kunze, P. Lukowicz, H. Junker, and G. Tröster, "Where am i: Recognizing on-body positions of wearable sensors," in *LoCA*, ser. Lecture Notes in Computer Science, T. Strang and C. Linnhoff-Popien, Eds., vol. 3479. Springer, 2005, pp. 264–275.

[39] 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 ICST 2nd international conference on Body area networks*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 1–6.

[40] J. Yang, "Toward physical activity diary: motion recognition using simple acceleration features with mobile phones," in *IMCE '09: Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*. New York, NY, USA: ACM, 2009, pp. 1–10.

[41] S. L. Lau and K. David, "Movement recognition using the accelerometer in smartphones," in *Future Network & Mobile Summit 2010*, 2010.

[42] C. Doukas and I. Maglogiannis, "Enabling human status awareness in assistive environments based on advanced sound and motion data classification," in *PETRA '08: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*. New York, NY, USA: ACM, 2008, pp. 1–8.

[43] S. Zhang, P. McCullagh, C. Nugent, and H. Zheng, "Activity monitoring using a smart phone's accelerometer with hierarchical classification," in *The 6th International Conference on Intelligent Environments*, Kualar Lumpar, Malaysia, 2010.

[44] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recogn.*, vol. 41, no. 6, pp. 2010–2024, 2008.

[45] E. Kim, S. Helal, and D. Cook, "Human activity recognition and pattern discovery," *IEEE Pervasive Computing*, vol. 9, no. 1, pp. 48–53, 2010.

[46] Pärkkä, Juha., M. Ermes, P. Korpipää, J. Mäntyjärvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 1, pp. 119 –128, jan. 2006.

[47] N. Wang, E. Ambikairajah, and B. G. Lovell, N.H.and Celler, "Accelerometry based classification of walking patterns using time-frequency analysis," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, 22-26 2007, pp. 4899 –4902.

[48] K. Van Laerhoven and H.-W. Gellersen, "Spine versus porcupine: a study in distributed wearable activity recognition," in *Wearable Computers, 2004. ISWC 2004. Eighth International Symposium on*, vol. 1, 31 2004, pp. 142 – 149.

[49] S. L. Lau, I. König, K. David, B. Parandian, C. Carius-Düssel, and M. Schultz, "Supporting patient monitoring using activity recognition with a smartphone," in *The 7th International Symposium on Wireless Communication Systems (ISWCS), 2010*, York, United Kingdom, September 19-22 2010, pp. 810 –814.

[50] T. Huynh and B. Schiele, "Unsupervised discovery of structure in activity data using multiple eigenspaces," in *2nd International Workshop on Location- and Context- Awareness (LoCA 2006)*, Dublin, Ireland, May 2006.

[51] D. Minnen, T. Starner, I. Essa, and C. Isbell, "Improving activity discovery with automatic neighborhood estimation," in *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2814–2819.

[52] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. New York, NY, USA: ACM, 2003, pp. 2–11.

[53] T. Choudhury, M. Philipose, D. Wyatt, and J. Lester, "Towards activity databases: Using sensors and statistical models to summarize people's lives," *IEEE Data Eng. Bull.*, vol. 29, no. 1, pp. 49–58, 2006.

[54] F. Ichikawa, J. Chipchase, and R. Grignani, "Where's the phone? a study of mobile phone location in public spaces," in *2nd International Conference on Mobile Technology, Applications and Systems 2005*, Guangzhou, nov. 2005, pp. 1 –8.

[55] MATRIX, "MATRIX - Middleware-plattform für die realisierung internetbasierter telemedizinischer dienste," available online at http://www.forschungsprojekt-matrix.de/, last checked on 1st July 2011.

4 Smartphone as an unobtrusive sensor device for activity recognition

This chapter presents an overview on an activity recognition system using a smartphone. The chapter begins with the elaboration of the rationale and motivation behind the idea. It also describes the research approach and the steps chosen to develop, investigate and evaluate the proposed activity recognition system. This includes the data collection and pre-processing tasks that prepare the needed data for the recognition algorithm evaluations.

4.1 Introduction

As described in the previous chapter, we want to utilise the accelerometer in a smartphone for the activity recognition system in CARMA. The smartphone is seen as a potential unobtrusive device that is more acceptable than the relatively more obtrusive multiple body-worn sensors approaches. We foresee the desired activity recognition system to have at least the following four major functions:

- It needs to collect the sensor data required for successful recognition.
- It has to process the sensor data that are going to be used as input information for the intended inference.
- The recognition process delivers the activity contexts as output.
- The activity contexts are then either stored in the system or be used for further steps, depending on the potential use of the recognised activities.

Based on the architecture, illustrated in Figure 3.2 on page 30, the first function is regarded as an CA component. The second and third functions are part of an CP component, though it is also possible to implement these two functions as two individual CP components. The final function is where one or more SA components are invoked.

A high-level way to explain the steps involved in designing the components for the desired activity recognition are illustrated in Figure 4.1. The first involved process is the modelling process (indicated using the green arrows). Available data

are prepared and processed by a selected learning algorithm. We defined these data as **training data**. The learning algorithm discovers useful information in the data. This information is then used to build models. The CP components utilise these models to recognise activities. The built models are known as classifiers.

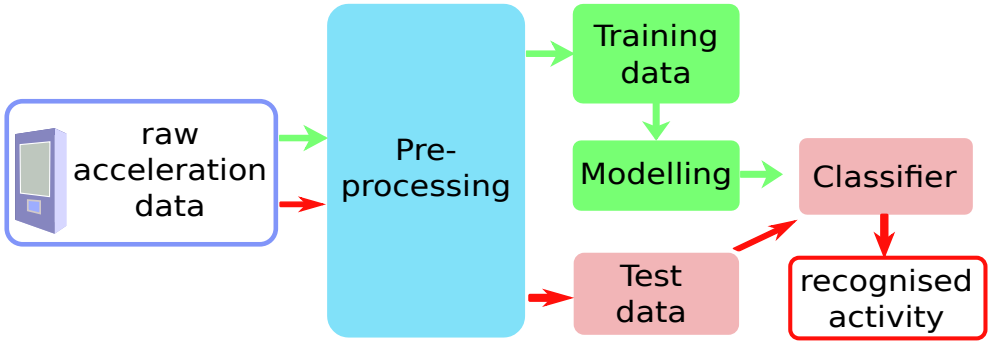


Figure 4.1: Steps involved in an activity recognition CP component.

In the second process (indicated by the red arrows), the acceleration data are measured and then pre-processed to produce the **test data**. The test data are then sent to the designated classifier to produce the recognised activity. In this thesis, we want to investigate and apply classification learning algorithm for the building of models, as the classification methods have shown good and promising results in previous investigations using multiple accelerometers. Theoretically, these classification methods should also provide similar recognition accuracies if we use the accelerometer of a smartphone as sensor data.

In the following sections, we present the three steps involved in the training data preparation process for the planned evaluations in Chapter 5. They are data collection, data pre-processing and feature extraction. These steps are also a component in the CARMA application.

4.2 Data collection

The first step in the activity recognition process is the data collection. The experiment data are collected using a selection of Nokia smartphones. The smartphones used for our experiments were a Nokia N95 8GB (Symbian 3rd Edition FP1), a Nokia N97 (Symbian 5th Edition) and a Nokia N900 (Maemo 5 Linux). A summary of the collected data is listed in Table 4.1. Total duration of collected data was 785.49 minutes.

We assumed that the recorded acceleration data had the same sensitivity because all these smartphones have the same LIS302DL accelerometer from the company STMicroelectronics. Nevertheless, it was observed that different smartphone operating system delivered different range of acceleration values. Our measurement showed that all three phones had different readings for the g values. When the

Table 4.1: Information on the collected data from the experiments.

Test user (TU)	Smartphone	Total minutes recorded
TU #1	N95 8GB	37.41
TU #2	N95 8GB	56.85
TU #3	N95 8GB	53.90
TU #4	N900	83.68
TU #5	N95 8GB	75.76
TU #6	N900	38.28
TU #7	N95 8GB	33.85
TU #8	N97	96.78
TU #9	N97	56.51
TU #10	N95 8GB	31.52
TU #11	N97	56.50
TU #12	N97	40.30
TU #13	N97	43.54
TU #14	N95 8GB	43.75
TU #15	N97	36.85

smartphones were placed on a table, the Nokia N95 8GB, N97 and N900 measured 300, 62 and 1000 as g respectively. This also indicated that if models for recognition are to be built from one smartphone and used in another smartphone for recognition, one needs to calibrate the acceleration data. For this purpose, the acceleration data can be normalised using the respective g values for each smartphone.

The accelerometer provides relative acceleration of three axes relative to the orientation of the smartphone. The LIS302DL accelerometer, according to the LIS302DL Datasheet [1], is designed for purposes such as free-fall detection, motion activated functions, gaming and virtual reality input devices, vibration monitoring and compensation. The Figure 4.2 depicts the three axes, measured by the accelerometer of a smartphone.

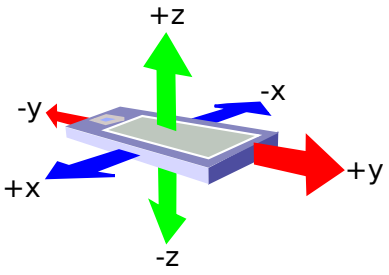


Figure 4.2: Three axes of the accelerometer relative to the orientation of a smartphone.

4.2.1 The accelerometer of a smartphone

The built-in accelerometer in smartphones is usually a Micro-electro-mechanical system (MEMS). The Figure 4.3 in page 42 represents a simplified illustration of how an accelerometer works. The structure in this figure is responsible for the acceleration measurement of an axis. The small box in the middle is the proof mass (the body of the sensor) that can only move in two directions on a plane. A movement with acceleration deflects the proof mass, and this creates the differential capacitance between the proof mass and the horizontal bar above it. The capacitance differences are measured and converted into values that represent the acceleration in the respective axes. If the accelerometer is on a moving object, it measures the acceleration of this object relative to gravitational force on all three axes. This mode of measurement is known as the dynamic acceleration.

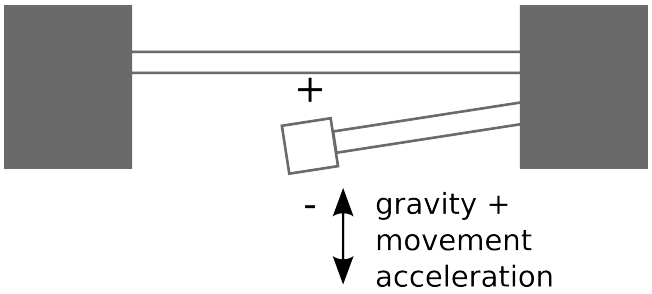


Figure 4.3: Simplified working principles of an MEMS accelerometer.

If the smartphone is in an idle state, for example, when a person carrying it is standing still or when the smartphone is placed on a table, the accelerometer measures the gravitational effect has on each axis. In other words, the proof mass of the accelerometer is still going to be deflected under the influence of gravitational force. This is known as static acceleration. If an axis is parallel to the gravity force, the accelerometer delivers a value of $1g$ (g is the gravitational force, where $1g = 9.81\ m/s^2$) as output. If an axis is perpendicular to the gravity force, it delivers $0g$ as the measured acceleration. Therefore, a tilted accelerometer gives different constant values on each axis, provided that the axes are not perpendicular to the gravity force. Commonly the static acceleration is applied on consumer products such as smartphones and digital cameras to detect tilt or gravity.

The usages of accelerometer in smartphones include automatic screen orientation change, gesture-based application control and stabilization function for image capture. The accelerometer measures the linear motion and gravity concurrently. Therefore, it is not originally designed to measure the actual human motion and movement directly. However, if the acceleration measurements consist of recognizable patterns for specific movements, it is possible to find these patterns in the acceleration data to recognise the corresponding movements.

According to the data sheet of the LIS302DL accelerometer [1], it is capable of detecting acceleration up to $\pm\ 2g/\pm\ 8g$. It has a sampling rate of 100 Hz or 400 Hz,

according to the data sheet. The current consumption is stated as $< 400\ \mu A$. We have performed a measurement on the actual current consumption using the Nokia Energy Profiler [1] on the Nokia N97. The obtain results are shown in Figure 4.4. The usage of accelerometer of a smartphone consumed averagely 8-10 mA more current than the consumption in an idle state (an average of 7 mA was observed). This may be due to the consumption involving the operating system and the acquisition of the sensor values via the provided Application Programming Interface (API). A further comparison on battery and current consumption will be presented in Chapter 6.

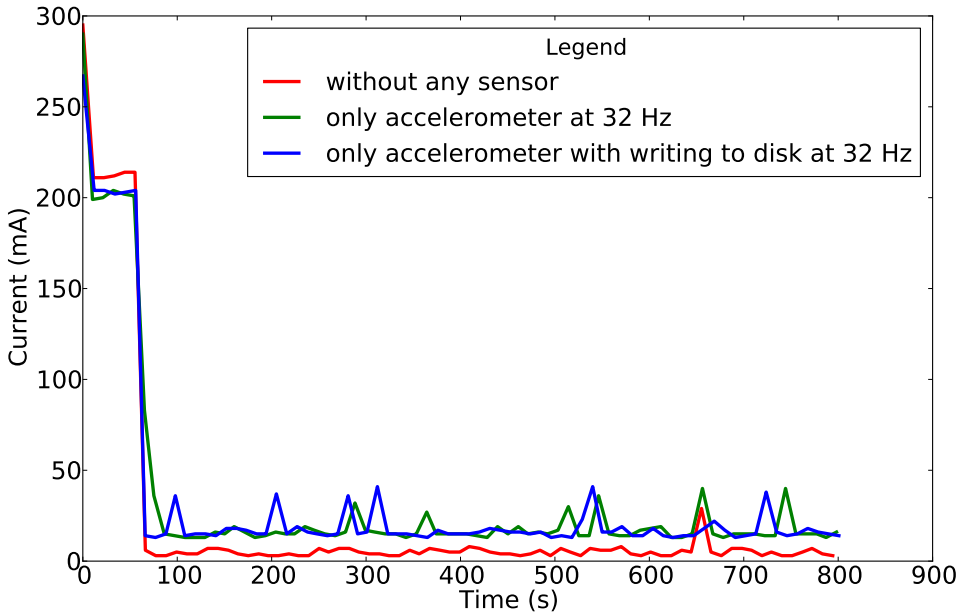


Figure 4.4: Current consumption measured on a Nokia N97 smartphone.

An important factor that can influence the outcome of the desired recognition is the sampling rate and the sensitivity of the accelerometer. We noticed in our measurements that while it is possible to obtain a recording of accelerometer data at sampling rates up to 190 Hz, with the certain programming environment such as Python for S60 (pyS60) [2] and m-Shell [3], there is a large number of repeated values in the collected data. For example, as shown in the Table 4.2, the recorded acceleration measurements were as fast as 190 Hz (time between each sample is around 0.0053 second). There are, however, only 4 unique acceleration data within the 0.1 second (highlighted in Table 4.2 as bold).

This observation suggested that only around 0.025 second is required to detect the acceleration changes. This translates to a sampling rate of 40 Hz. In the collected data from all the test users, the average sampling rate with no repetition was around 35-40 Hz. As compared to the investigations mentioned in Section 3.1, this value is

[1] Available online at `http://store.ovi.com/content/17374`, last checked 1st August 2010.

Table 4.2: A snippet of the obtained acceleration data from a test user.

Timestamp (s)	x	y	z
1269530170.0083	**106**	**324**	**-21**
1269530170.0143	106	324	-21
1269530170.0183	106	324	-21
1269530170.0223	106	324	-21
1269530170.0283	**159**	**356**	**-47**
1269530170.0343	159	356	-47
1269530170.0383	159	356	-47
1269530170.0423	159	356	-47
1269530170.0483	159	356	-47
1269530170.0542	**159**	**340**	**-69**
1269530170.0583	159	340	-69
1269530170.0633	159	340	-69
1269530170.0692	159	340	-69
1269530170.0763	**122**	**430**	**15**
1269530170.0793	122	430	15
1269530170.0832	122	430	15
1269530170.0893	122	430	15
1269530170.0953	122	430	15
1269530170.0993	122	430	15
1269530170.1033	**31**	**409**	**15**

lower than the investigations that used one or more dedicated accelerometers. The influence of sampling rate will be compared in the evaluations. For example, some of the previous investigations have used accelerometers that produced samples at higher sampling rates ([4] at 76.5 Hz, [5] at 50 Hz and [6] at 93 Hz). This raises a question: is it possible to obtain equivalent or better recognition accuracy with lower sampling rate?

From the physical movement's point of view, most of the common daily activities do not involve rapid movements. Activities such as sitting, standing and walking do not require any part of the body to move very fast. Bouton et al. [7] mentioned in their work that a sampling rate of 20 Hz is required to assess daily physical activity. Parkka et al. [8] observed that walking is seen as 2 Hz and running as 2.5-3 Hz oscillation in frequency. Bieber et al. [9] analysed different human movements and concluded that a human reflex reaction produces a movement within 0.06 seconds, which corresponds to 16 Hz. They concluded that a sampling rate of 32 Hz for acceleration would be sufficient for simple body movements based on Shannon theorem. Based on these observations, we investigate whether if a sampling rate of 32 Hz or below is usable for activity recognition using accelerometer of a smartphone.

Another reason to use lower sampling when possible is to reduce the load of the

Figure 4.5: Screenshot of the logging application running on the Nokia N800 Internet Tablet.

smartphone. The more data to be measured, the more resources will be needed for either storage, transmission or processing. Resources of a smartphone such as disk space, memory and battery life, should be kept as efficient as possible. We need to ensure that the use of a smartphone as an unobtrusive sensor device does not affect the other functions of the smartphone and end up bringing frustration to the users.

4.2.2 Experiment data collection

Five movements were selected as the basic activities to be recognised - *sitting*, *standing*, *walking*, *go upstairs* and *go downstairs*. They are the most frequently performed common movements for many people. Sleeping is excluded because usually one does not carry the smartphone with him to bed. 15 test users were involved in the experiments. Each test user was requested to perform all five movements in a natural manner (without fixed duration and sequence). A smartphone was placed in the trouser pocket for the recording of the acceleration data. This placement location is seen as strategic for two reasons. Firstly, the lower body is seen as a suitable area for active movements that involve leg movements [10] [6]. Secondly, in the study made by Cui et al. [11] the trouser pocket is the most common place (60 %) for male users to place their mobile phones. This value is a lot lower for female users (only 16 %) [2]. However, we foresee the users have higher chances to accept the choice of sensor placement with a smartphone somewhere near their thigh if they wish to use it for activity recognition.

For the preparation of the training data, the user has to annotate the different activities using the logging application on the Nokia N800 Internet Tablet. A screen shot of the application is shown in Figure 4.5. These annotations are used as class information for the model learning as well as the ground truth to be compared

[2]The study indicated bags as the most popular choice for female users (61 %)

with the recognition results. Both the data from the smartphone and the Internet tablet were combined with a script, after the data collection process to generate the training data for the activity recognition evaluations.

We let the users perform the annotations by themselves. This allows the collection of training data to be performed under naturalistic circumstances. Bao and Intille argued that data from the laboratory environments may have the tendencies to be artificially constricted, simplified, or influenced [4]. Therefore, all 15 test users were only given instructions on how they should use the two devices for data collection. They were requested to perform both acceleration data recording and annotation on their own without the researchers' direct help and intervention. It was also expected that this setup to be closer to a situation where users perform the modelling process on their own with the help of appropriate software tools. The obtained results are expected to reflect recognition accuracies based on non-laboratory controlled measurements.

4.2.3 The collected acceleration data

The triaxial accelerometer of a smartphone measures the acceleration of the user. It is observed from the obtained data that it is possible to set apart active movements from idle movements (*standing* and *sitting*). Figure 4.6 and Figure 4.7 display the recorded acceleration data of the triaxial accelerometer in a Nokia N95 8GB. Both movements display measurements of static acceleration, where the axis parallel to the gravitational force shows a value of $\pm 1g$ while two other axes perpendicular to the gravitational force show a value of $0g$. It was also observed that when a test user was sitting, the smartphone might be tilted. Therefore, the axes might not be completely parallel or perpendicular to the gravitational force (e.g. test user #3 and #4 in Figure 4.7). Nevertheless, even with slight movements in between *standing* and *sitting*, these two movements are expected to be distinguishable.

The other three movements are more difficult to tell apart than the movements *sitting* and standing. Basically, *go upstairs* (Figure 4.8) and *go downstairs* (Figure 4.9) are similar to *walking* (Figure 4.10). These three basic activities show repetitions of steps-related acceleration patterns. However, if we take a closer look at the magnitudes of the axes there are still noticeable differences between the three activities. Therefore, we investigate suitable techniques that can extract and *highlight* these differences. These techniques are known as feature extraction and are presented in section 4.4.

Besides the extraction of useful features, it is also possible to perform some pre-processing tasks on the recorded acceleration data. The purpose of pre-processing, when required in an activity recognition process, is to help improve the recognition process. The next section elaborates two data pre-processing proposals that have been tested in this work.

4.3 Data pre-processing

The raw acceleration data potentially has two issues that can affect the recognition accuracy. Firstly, the acceleration data display jitter noise. This can be well observed in the measurements of the activities of *standing* and *sitting*(see Figure 4.6 and Figure 4.7). Secondly, the orientation of the smartphone's placement in the trouser pocket influences the values of each axis. For example, by observing the acceleration data of the movement walking (Figure 4.10), only user #2 and #3 had the same orientation, while user #1 and user #4 had selected another two different orientations. If the orientation of the smartphone has to be fixed in order to enable a more consistent acceleration measurement, this also means that the user must constantly ensure the smartphone to remain at the same fixed-position throughout the usage of the movement recognition system.

Smoothing techniques are a possible method to solve the first issue. However, it is important to investigate whether a given chosen smoothing technique brings any advantage to the recognition accuracy. For the second issue, one has to find a technique that can provide orientation-independent acceleration data. It involves a conversion of the obtained acceleration measurements into values that are not affected by potential orientation change. The following sub-sections present the respective data pre-processing techniques that can potentially solve these two issues.

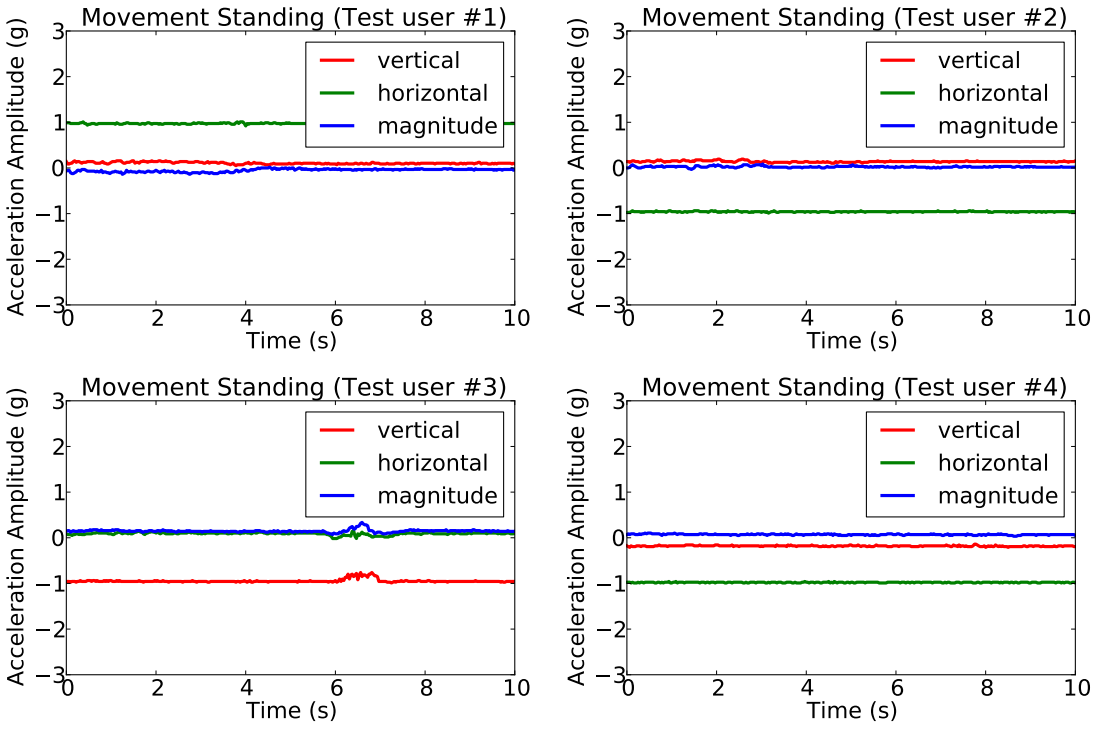


Figure 4.6: Acceleration of 4 test users for the movement *standing*.

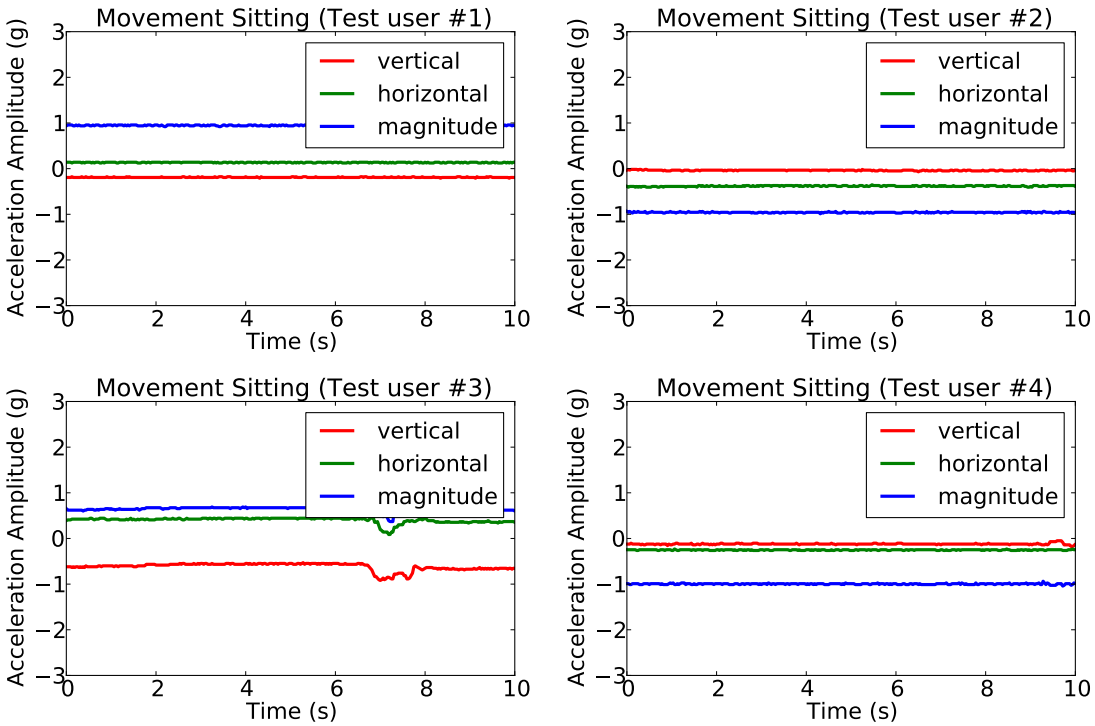


Figure 4.7: Acceleration of 4 test users for the movement *sitting*.

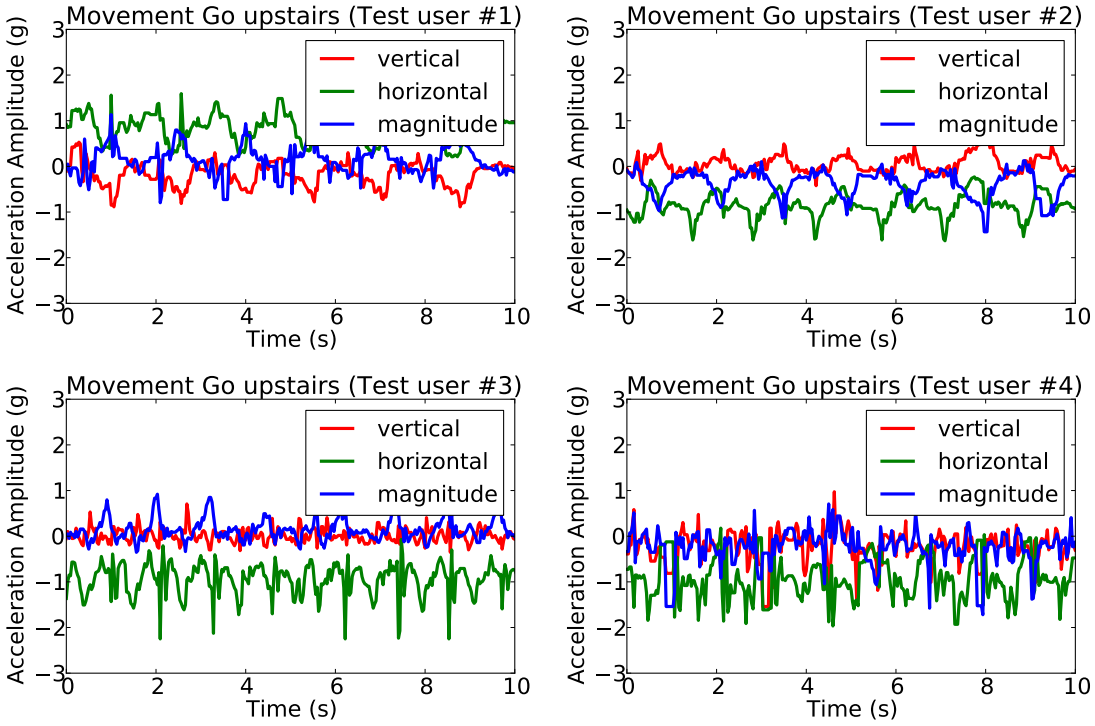


Figure 4.8: Acceleration of 4 test users for the movement *go upstairs*.

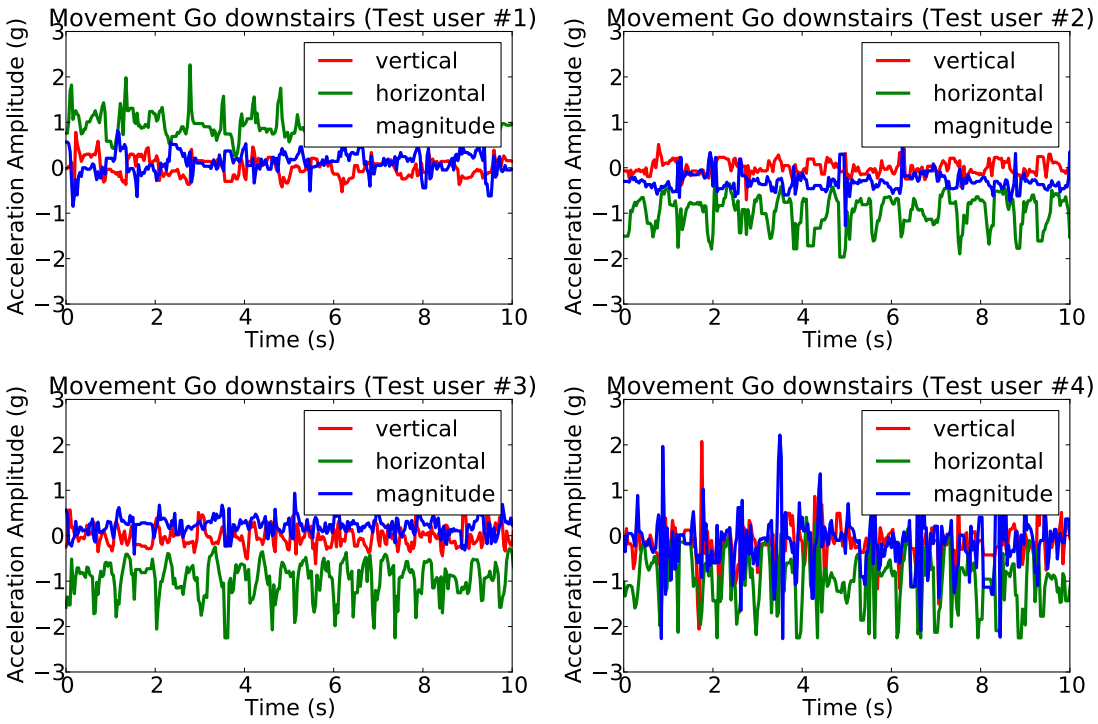


Figure 4.9: Acceleration of 4 test users for the movement *go downstairs*.

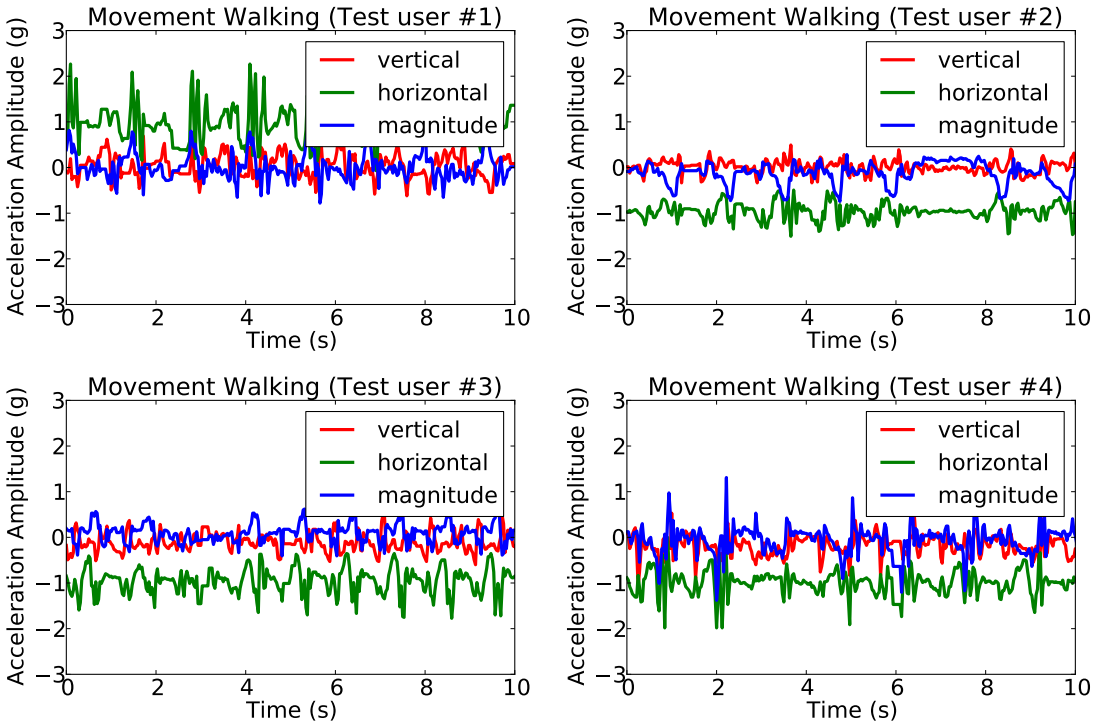
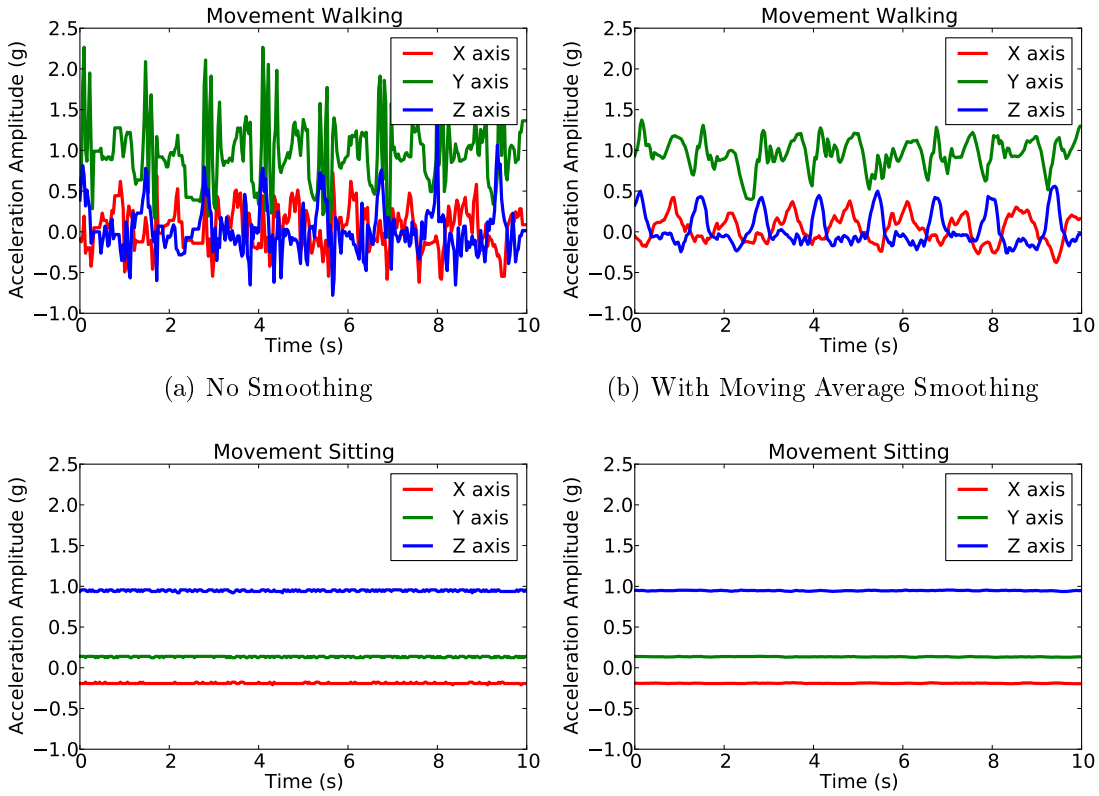


Figure 4.10: Acceleration of 4 test users for the movement *walking*.



(a) No Smoothing

(b) With Moving Average Smoothing

(c) No Smoothing

(d) With Moving Average Smoothing

Figure 4.11: Comparison between raw and smoothed acceleration data for movement *walking* and *sitting*.

4.3.1 Smoothing techniques

The jitter noise found in the obtained acceleration data can be removed using suitable smoothing techniques. Smoothing techniques have the purpose to reduce irregularities in time series data. Two examples of the irregularities are random fluctuations and noise from the environment. The use of smoothing techniques should provide a better and clearer view of the actual underlying behaviour of the measured data. These techniques are useful in improving visualization of the measured data for understanding and further analysis. Sometimes, the removal of jitter noise through smoothing might also be able to reduce the influence of irregularities in data that can affect the recognition outcome. The decision of whether or not to use smoothing techniques for the obtained data and the intended applications requires proper analysis to investigate the potential influence of the chosen smoothing techniques.

An example is shown in Figure 4.11, where the corresponding graphs demonstrate how the noisy acceleration data obtained from test user #1 is smoothed. The Figure

4.11b and Figure 4.11d are relative "cleaner" as compared to Figure 4.11a and Figure 4.11c. Nevertheless, one has to take two issues into consideration when he wishes to use smoothing techniques to filter the data. Firstly, he needs to ensure the selected smoothing technique does not remove useful information from the data. Over-filtering may result in the lost of usable acceleration information that helps to improve recognition accuracy.

Secondly, the smoothing technique also introduces the need of additional processing power and time. If an activity recognition system has to be implemented on a mobile device, the additional processing power will affect the battery life of the device, which is something seen as a disadvantage in certain usage scenario. Some of the smoothing techniques first require to collect a certain amount of samples in the time series before the smoothing calculation can be performed. For a real time recognition system, this requirement produces a potential delay for the subsequent processes and tasks. An offline recognition system, however, is not affected by this issue.

4.3.2 Orientation-independent acceleration

The placement and the orientation of the smartphone are two factors that affect the acceleration measurements. Since we intend to have a more natural and unobtrusive use of the smartphone, the test users have performed the measurements without the help and instruction of researchers. Also, they have placed the smartphone in the trouser pocket as how they usually do. This makes comparing the acceleration data between users difficult, as the smartphones do not have the same orientation [12]. This issue can also happen to a single user, because the smartphone may not maintain the same orientation during the measurement process.

One simple approach to remove the influence of orientation to acceleration measurement is to compute the magnitude of all three axes, as shown in Equation 4.1

$$mag = \sqrt{x^2 + y^2 + z^2} \tag{4.1}$$

A disadvantage of the value of magnitude is the lost of directional information. The 3D acceleration information is seen as important information that can help to distinguish different activities. Another useful technique that can solve the orientation issue is to estimate the gravity force components from the available acceleration data. This was suggested by Mizell [13] and has been applied in [14].

The algorithm suggested by Mizell works as follows: let $\vec{a} = (a_x, a_y, a_z)$ be the vector of the obtained acceleration data. A chosen sampling interval is selected to estimate the average of each axis in this interval. The produced values are the estimation of the vertical acceleration vector $\vec{v_i}$, where $i = 1, 2, \ldots, N$ and N are the total samples after the splitting of acceleration data using the sampling interval.

Therefore, $\vec{v}$ is presented as:

$$\vec{v} = (m_{a_x}, m_{a_y}, m_{a_z}) \tag{4.2}$$

where m_{a_x}, m_{a_y} and m_{a_z} are the means of the respective axes for each interval. The dynamic component of $\vec{a}$, $\vec{d}$ can be computed as shown in Equation 4.3:

$$\vec{d} = (a_x - m_{a_x}, a_y - m_{a_y}, a_z - m_{a_z}) \tag{4.3}$$

The dynamic component $\vec{d}$ is an estimation of the acceleration caused by the user's motion rather than gravity. Therefore, $\vec{d}$ is seen as the estimation of the user's orientation-independent acceleration. Let $\vec{p}$ be the vertical component and $\vec{h}$ the horizontal component of the vector $\vec{d}$, and one can compute the $\vec{p}$ as the projection of $\vec{d}$ upon the vector $\vec{v}$, therefore:

$$\vec{p} = \left(\frac{\vec{d} \cdot \vec{v}}{\vec{v} \cdot \vec{v}} \right) \vec{v} \tag{4.4}$$

In the same way, the horizontal component $\vec{h}$ can be computed by vector subtraction as shown in Equation 4.5

$$\vec{h} = \vec{d} - \vec{p} \tag{4.5}$$

The shortcoming of this algorithm is the problem to know the orientation of $\vec{h}$ relative to $\vec{f}$. The only compromise adapted by the previous investigations is to use the magnitude of $\vec{h}$. Since most movements show more influences and changes in the vertical component of the dynamic acceleration $\vec{p}$ than in the horizontal component $\vec{h}$, this compromise is the best what one can expect to do. The results of this algorithm are two waveforms of $\vec{p_i}, i = 1, 2, \ldots, N$ and $\left|\vec{h_i}\right|, i = 1, 2, \ldots, N$, where N is the total samples after the splitting of acceleration data using the sampling interval. The following figures (Figure 4.12 - 4.16) are the results computed using this algorithm based on the acceleration data showed in Figure 4.6 - 4.10:

From the obtained results, it was observed that the orientation-independent acceleration data for the four test users showed more similarity than the original raw acceleration data. For the activities *standing* and *sitting* all acceleration estimations of the four test users were close to 0 when the users were not moving. Though for activities *walking*, *go upstairs* and *go downstairs* it was not exactly identical among the test users, but this will be investigated in the evaluations which are elaborated in the Chapter 6.

4.4 Feature Extraction

Feature extraction is a technique that enables the reduction of dimensionality of the data and the discovery of useful patterns. A feature is defined as a new attribute

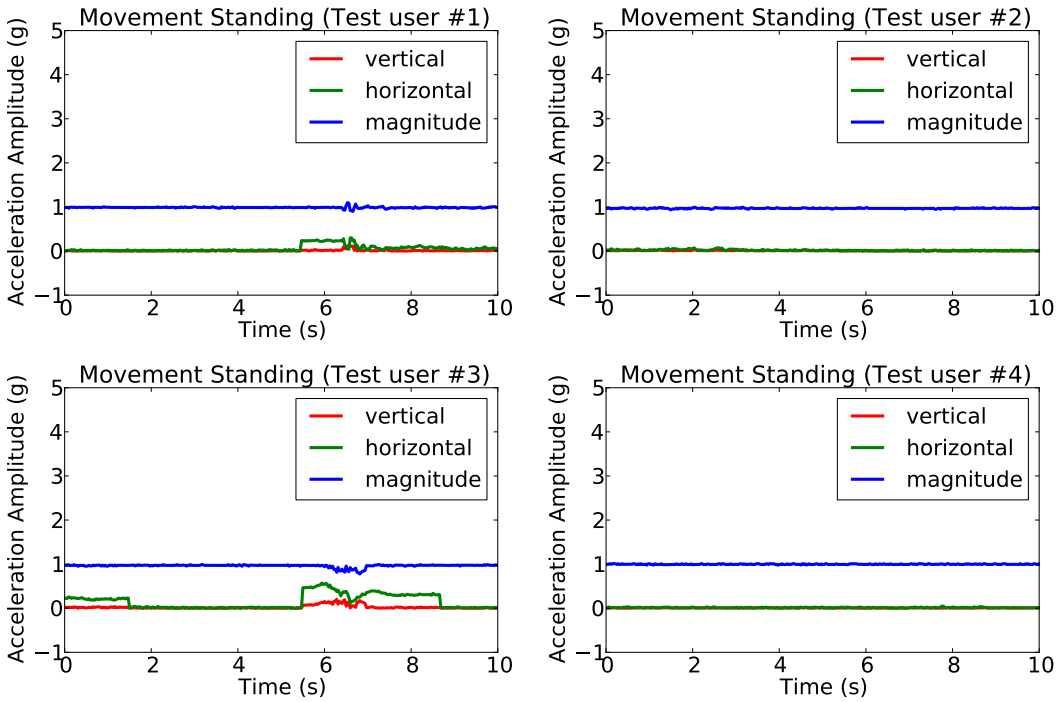


Figure 4.12: Orientation-independent values of 4 test users for the movement *standing*.

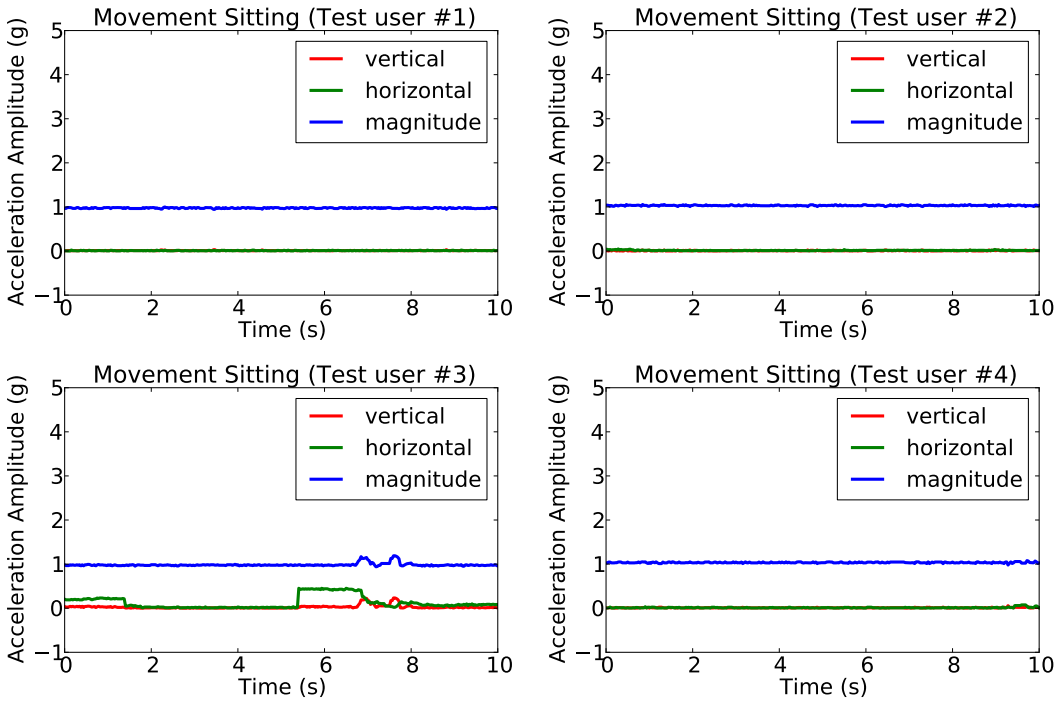


Figure 4.13: Orientation-independent values of 4 test users for the movement *sitting*.

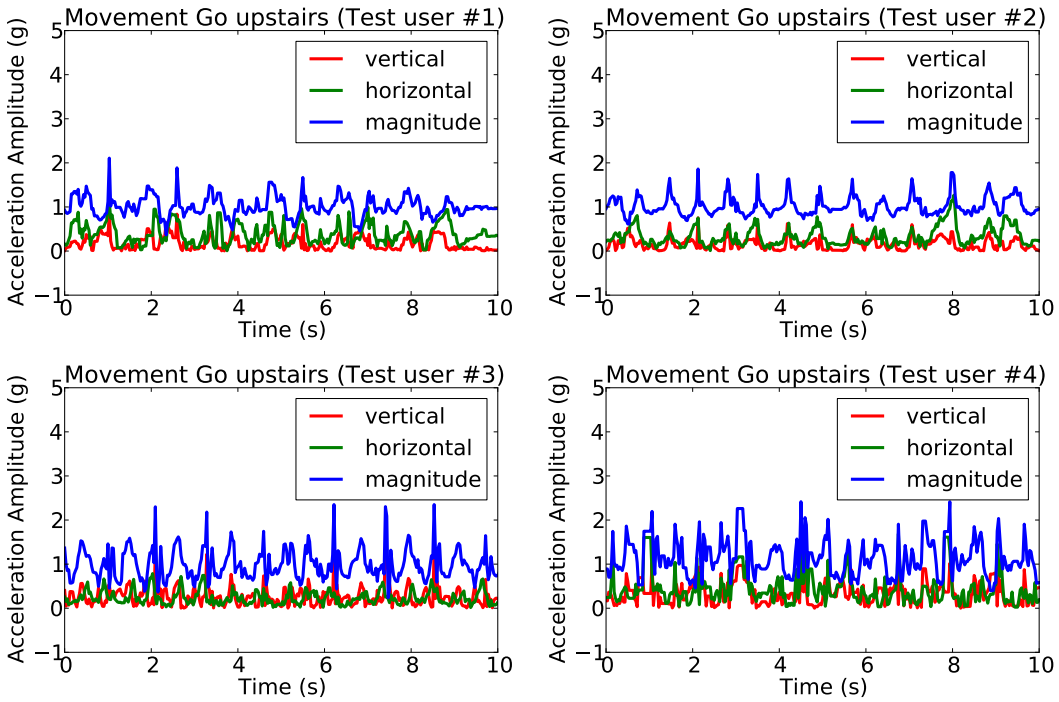


Figure 4.14: Orientation-independent values of 4 test users for the movement *go upstairs*.

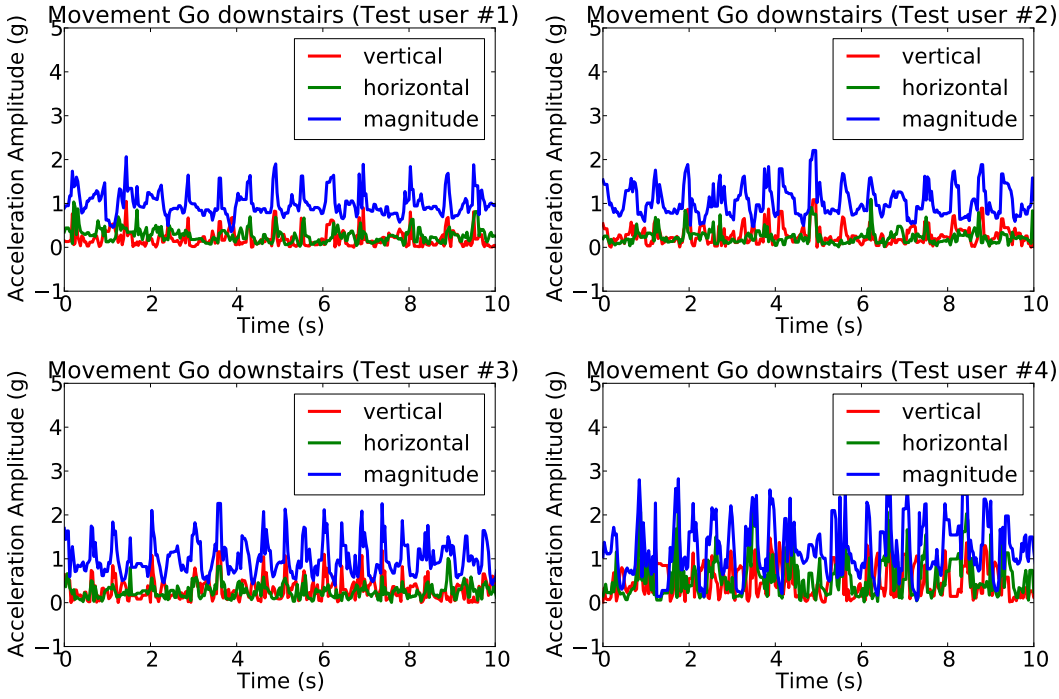


Figure 4.15: Orientation-independent values of 4 test users for the movement *go downstairs*.

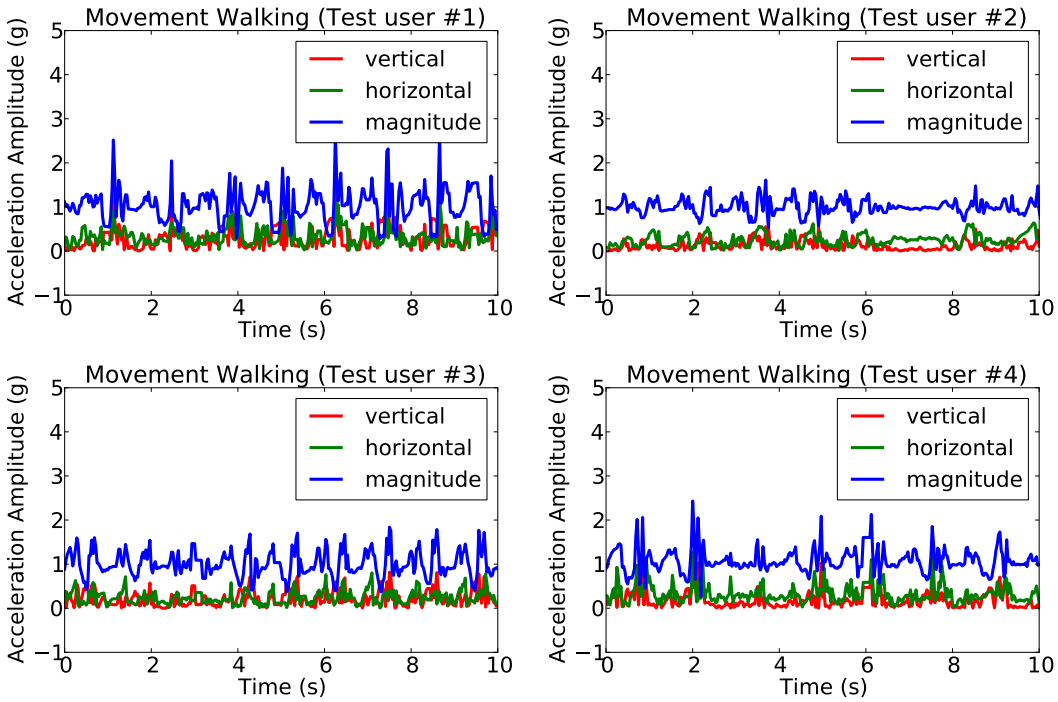


Figure 4.16: Orientation-independent values of 4 test users for the movement *walking*.

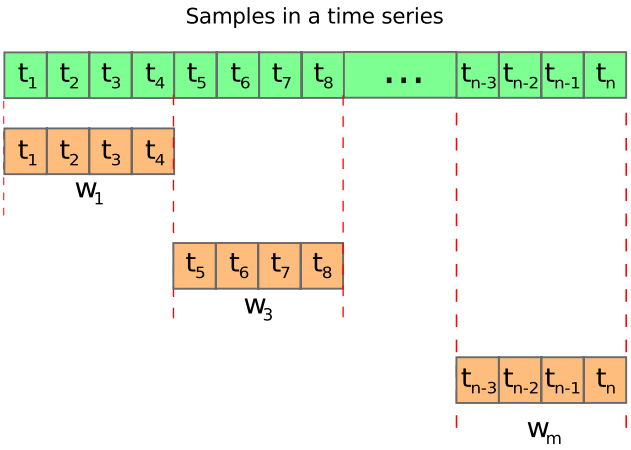
generated from the original raw data. Example of this technique can be seen in fields such as image processing and signal pattern matching. Feature extraction is useful especially when the original data are not directly usable for potential processing using algorithms such as classification or clustering. One specific type of such data is a time series data. Pyle [15] stated that a time series is not suitable to be directly analysed and processed by classification algorithms. Similarly, the accelerometer data, which are also time series, need to be transformed to obtain suitable features that can be applicable in the activity recognition processes.

This technique is usually highly domain-specific. In other words, features that are appropriate in a particular field are not equally appropriate for another field. The following sub-section presents the sliding window technique, which is a step needed for the computation of features. This is followed by an elaboration on feature extraction methods, with particular focus on features selected for the designated activity recognition system of this thesis.

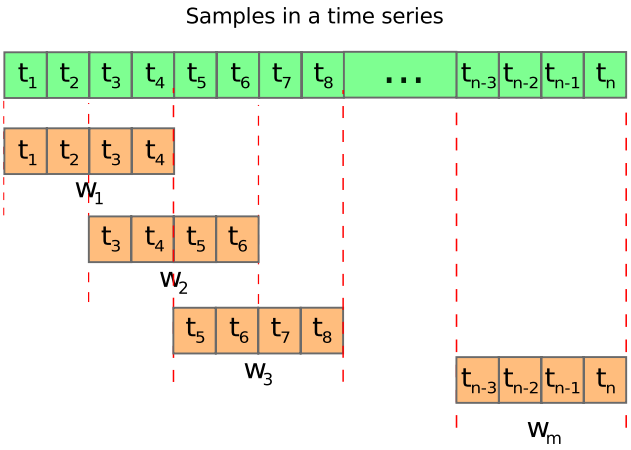
4.4.1 Sliding window data preparation

The first step in feature extraction process is to split the given accelerometer data into data segments with a fixed interval. This technique is usually known as the sliding window algorithm [16]. The algorithm is useful if one wishes to compare the segments to discover recurring patterns.

The Figure 4.17 in page 56 illustrates the technique.



(a) Sliding window with no overlapping



(b) Sliding window with 50 % overlapping

Figure 4.17: Data segmentation using sliding window.

A data segment is grown until it exceeds the given interval to form a so-called "window". If no overlapping is desired, the following window starts from where the previous window stops at. This can be illustrated as in the Figure 4.17a. In this figure, a time series sample is split into m windows. Each window has a window length of four samples. An overlapping sliding window has its following segments starting from a certain position (depending on the percentage of the overlap) in the newly created segment. An example is shown in Figure 4.17b where new data segments are generated using a sliding window of four samples with 50 % overlap.

The sliding window technique is commonly used in previous investigations. The accelerometer data are processed to produce three sets of data segments before the feature extraction step is performed. Most of the investigations using classification techniques selected 50 % as the overlap percentage, such as Laerhoven et al. [10], Bao and Intille [4], Ravi et al. [5]. Mäntyjävi et al. [17] used 75 % overlapping. The use of overlapping windows has the advantage of retaining the similarity of

data segments. The repeated samples in two subsequent windows may increase the similarity between them. It is also useful when the sample size of the training data is relative small [18]. A set of training data produces more instances with a higher percentage of overlapping than the same training data with a lower percentage of overlapping.

The next step to take after obtaining the sliding windows from the accelerometer data (both raw and orientation-independent data) is to perform the feature extraction steps. This is explained in the next sub-section.

4.4.2 Features extraction computation/transformation

For the classification-based activity recognition, feature extraction is an essential process. As a comparison, the selected features in the previous investigations are listed in Table 4.3 in page 58. Among them, most frequently selected features are mean, standard deviation and the fast Fourier transform (FFT) energy. The choice of simple statistic features is due to the simplicity and low computational cost. The FFT features are suitable to identify movements with distinguishable frequencies.

Feature extraction involves the computation and transformation of the sliding windows into newer values. A simple example is to compute the mean value for every window. The result of this computation is a new series of values that represent mean of the original time series at a given interval. An example illustration is shown in Figure 4.18 in page 59, where two features have been computed using mean and standard deviation for the given accelerometer data.

The corresponding features shown in the Figure 4.18 reflect the acceleration changes taken place in the top graph. For instance, the standard deviation values showed an increase and changes when the test user is in motion or when a transition takes place in his movements. The mean values have less significant differences between acceleration changes, but there are still noticeable patterns that can be used to recognise the movements. Transformations such as mean and standard deviation computations are relatively simpler features obtainable in the time domain. There are also transformations that attempt to extract information and patterns in another domain space such as frequency domain. This can be achieved by using the Fourier transform method to convert the time series to a representation in the frequency domain. Regardless of the transformation choices, the goal of feature extraction is to identify possibly enough different features that help the recognition system to better differentiate and detect the correct movements.

Five features have been selected to be evaluated. Mean, variance and standard deviation are the selected simple statistic features. Besides that, two frequency domain FFT-based features, energy and information entropy have also been selected. The formulas for the selected features are listed in Table 4.4. The mean of the acceleration value for each axis represents the DC component of the acceleration data. The variance and standard deviation values are used to represent the range

Table 4.3: Features selected and evaluated in previous investigations.

Investigation	Total features used	Features selected/evaluated
Laerhoven and Cakmaci [10]	3	mean of the sum of maximum (50 samples) and standard deviation (50 samples) in a 100 samples window, zero-crossings, mean of the standard deviation for 20 samples in a 100 samples window.
Mäntyjärvi et al. [17]	2	Principal Component Analysis (PCA), Independent Component Analysis (ICA)
Bao and Intille [4]	10	mean, variance or standard deviation, frequency domain energy, frequency domain entropy, correlation between axes
Van Laerhoven and Gellersen [19]	3	average, variance, peak set descriptors
Ravi et al. [5]	4	mean, standard deviation, frequency domain energy, correlation between axes
Huynh and Schiele [18]	2	mean, variance
Lester et al. [20]	651^{+}	including cepstral coefficients, log Fast Fourier transform (FFT), frequency bands, spectral entropy energy, mean, variance, linear FFT frequency bands, correlation coefficient integration
Pärkkä et al. [8]	5*	peak frequency, median, peak power, variance, sum of variances
Kern et al. [6]	2	mean, variance

Note: + - 651 was the total number of features computed from 8 different sensors. The exact number of features for accelerometer was not mentioned in the literature.

* - the sixth feature stated in the literature is computed from a magnetometer

of acceleration differences from the mean, which may be representative if different movements demonstrate distinguishable ranges. The FFT-based features (energy and information entropy) are chosen because the frequency domain characteristics in the acceleration may be usable to discriminate movements with different frequencies. The energy is calculated using the sum of the squared FFT component magnitudes of the acceleration of each axis and divided by the number of samples for normalization. Similarly, the information entropy of the discrete FFT component magnitudes of the acceleration values is also normalised. The information entropy may be used to support discrimination of movements with similar energy values [4].

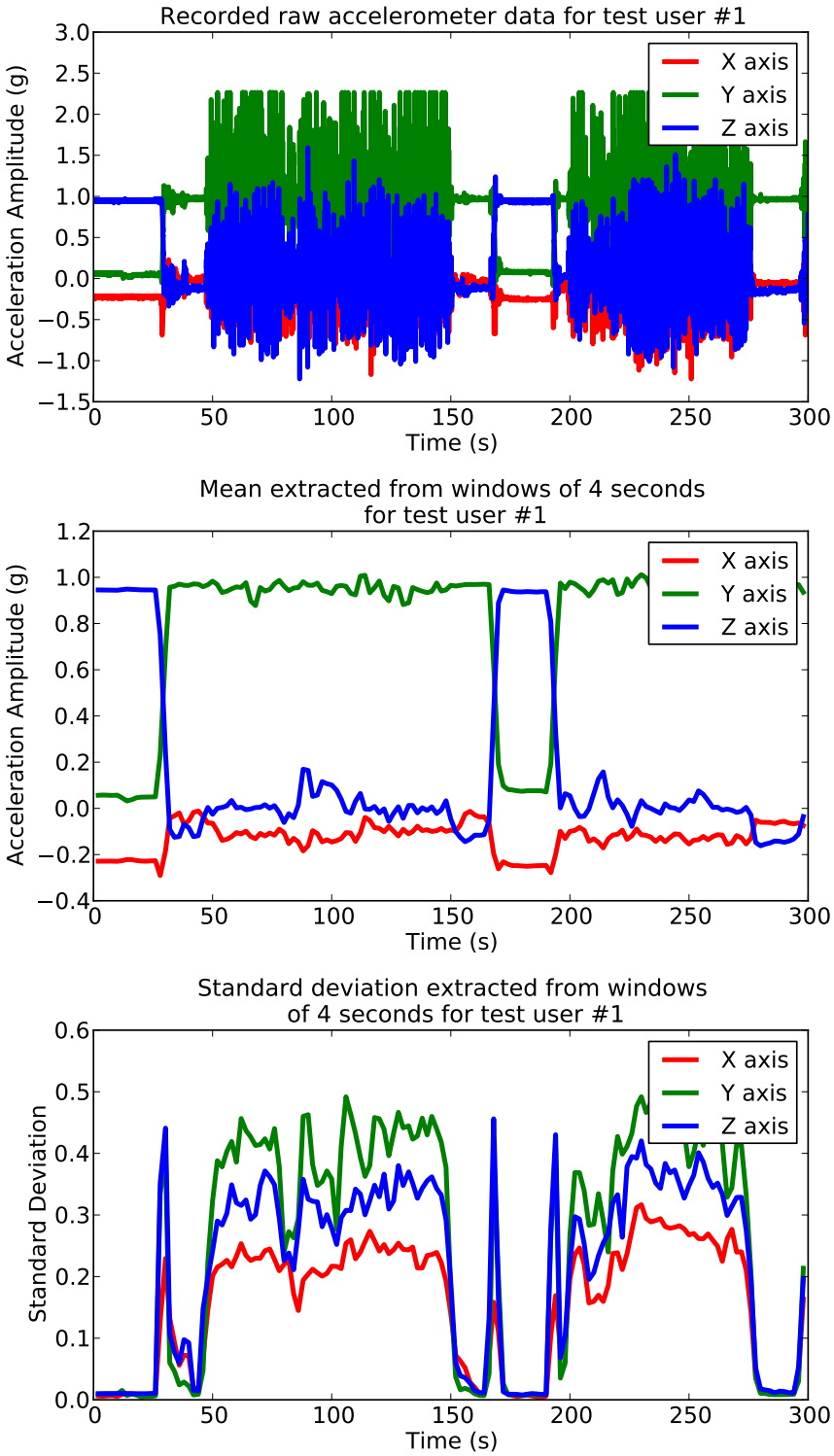


Figure 4.18: Comparison between accelerometer raw data, extracted mean values and extracted standard deviation values for a test user.

Table 4.4: Formulas for the computation of the selected features.

Feature	Formula
mean	$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
variance	$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} x_i - \bar{x}$
standard deviation	$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i - \bar{x}}$
energy	$E = \frac{1}{n}\sum_{i=0}^{n} \lvert FFT_i\rvert^2$
information entropy	$I = -\frac{1}{\ln n}\sum_{i=0}^{n} \lvert FFT_i\rvert \cdot \ln(\lvert FFT_i\rvert)$

Legend: x = acceleration of a single axis, $|FFT|$ = FFT magnitude, n = number of samples

The total number of samples for each feature is relatively smaller than the original accelerometer data. For example, if the original data has 1000 samples and a window interval of 10 is selected, the total number of samples for each feature is 100 (with no overlapping) and 199 (with 50 % overlapping). Therefore, if the feature extraction process is not computational complex, this reduction of the total number of samples may help improve recognition speed, provided the number of selected features are also kept relatively low, and the features are suitable and applicable for the designated recognition.

4.5 Training data preparation for the recognition evaluations

Following the requirements stated in Chapter 3, we wanted to select only the simpler and common features, which were the simple statistic features and FFT features. As compared to the former, the FFT features are considered as more computational complex features. The decision to use these features has two main reasons: Firstly, we want to compare the recognition accuracies of the approach using a smartphone with those using dedicated accelerometers. The selected features are commonly found in many previous investigations. This allows us to compare our results with theirs. Secondly, we wish to investigate whether if a smaller set of features (preferably features with simpler computation requirements) will be sufficient for good recognition. This is particularly important and interesting for an implementation on a smartphone.

For the experiments, the obtained accelerometer data have been resampled to produce samples at 8 Hz, 16 Hz, 32 Hz and 64 Hz. We regarded 32 Hz as the maximum meaningful frequency obtainable using a smartphone, since distinguishable acceleration values are only measurable up to 40 Hz. Furthermore, it is also mentioned by some investigations that a sampling rate at 32 Hz should be sufficient [7] [8] [9]. The sampling rates 8 Hz and 16 Hz are chosen to investigate whether lower sampling rates can provide equivalent recognition accuracies as compared to previous work and higher sampling rates. The sampling rate 64 Hz is used as a comparison, since we aim to use lower sampling rate when possible.

Next, the resampled training data were split into selected window lengths with different overlap percentages. The window lengths were 0.5, 1, 2 and 4 seconds. Total samples in a window were the product of the sampling rate multiply by the window length. We selected four different overlap percentages, which are 0 %(no overlapping), 25 %, 50 % and 75 %. Many previous investigations only used 50 % as default overlap percentages. We wanted to find out whether if other overlap percentages could produce equally good or better recognition accuracy. The resulted training data with different combinations of sampling rate, window lengths and overlap percentages were computed to produce the desired features for evaluations. These features were the input data, needed for the generation of the desired classifiers, which will be used for the recognition of movements.

In the next chapter, the evaluations and the obtained results are presented.

References

[1] STMicroelectronics, “LIS302DL - MEMS Motion Sensor,” available online at http://http://www.st.com/stonline/products/literature/ds/12726.pdf, last checked on 1st August 2011.

[2] PyS60, "Python for S60," Available online at http://wiki.opensource.nokia.com/projects/PyS60/, last checked 1st August 2010.

[3] A. AG, "mShell Makes Smart Phones Smarter," Available online at http://www.m-shell.net/, last checked 1st August 2010.

[4] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive 2004*, pp. 1–17, April 2004.

[5] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," *American Association for Artificial Intelligence*, 2005.

[6] N. Kern, B. Schiele, and A. Schmidt, "Recognizing context for annotating a live life recording," *Personal Ubiquitous Comput.*, vol. 11, no. 4, pp. 251–263, 2007.

[7] C. Bouten, K. Koekkoek, M. Verduin, R. Kodde, and J. Janssen, "A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity," *Biomedical Engineering, IEEE Transactions on*, vol. 44, no. 3, pp. 136 –147, mar. 1997.

[8] Pärkkä, Juha., M. Ermes, P. Korpipää, J. Mäntyjärvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 1, pp. 119 –128, jan. 2006.

[9] G. Bieber, J. Voskamp, and B. Urban, "Activity recognition for everyday life on mobile phones," in *HCI (6)*, 2009, pp. 289–296.

[10] K. Van 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.

[11] Y. Cui, J. Chipchase, and F. Ichikawa, "A cross culture study on phone carrying and physical personalization," in *HCI (10)*, ser. Lecture Notes in Computer Science, N. M. Aykin, Ed., vol. 4559. Springer, 2007, pp. 483–492.

[12] T. Brezmes, J. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*. Salamanca, Spain: Springer-Verlag, 2009, pp. 796–799.

[13] D. W. Mizell, "Using gravity to estimate accelerometer orientation," in *7th International Symposium on Wearable Computers (ISWC 2003), 21-23 October 2003, White Plains, NY, USA*. IEEE Computer Society, 2003, pp. 252–253.

[14] J. Yang, "Toward physical activity diary: motion recognition using simple acceleration features with mobile phones," in *IMCE '09: Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*. New York, NY, USA: ACM, 2009, pp. 1–10.

[15] D. Pyle, *Data preparation for data mining*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[16] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani, "An online algorithm for segmenting time series," in *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 289–296.

[17] J. Mäntyjärvi, J. Himberg, and T. Seppänen, "Recognizing human motion with multiple acceleration sensors," in *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, vol. 2, 2001, pp. 747–752 vol.2.

[18] T. Huynh and B. Schiele, "Towards less supervision in activity recognition from wearable sensors," in *Proceedings of the 10th IEEE International Symposium on Wearable Computing (ISWC 2006)*, Montreux, Switzerland, October 2006, pp. 3 –10.

[19] K. Van Laerhoven and H.-W. Gellersen, "Spine versus porcupine: a study in distributed wearable activity recognition," in *Wearable Computers, 2004. ISWC 2004. Eighth International Symposium on*, vol. 1, 31 2004, pp. 142 – 149.

[20] J. Lester, T. Choudhury, and G. Borriello, "A practical approach to recognizing physical activities," in *Pervasive*, ser. Lecture Notes in Computer Science, K. P. Fishkin, B. Schiele, P. Nixon, and A. J. Quigley, Eds., vol. 3968. Springer, 2006, pp. 1–16.

5 Classifying activities using accelerometer of a smartphone

In order to investigate how an activity recognition system can be developed using smartphones as the sole sensor and computing device, experiments have been carried out to compare suitable data pre-processing and recognition techniques. This chapter focuses on using classification algorithms to perform activity recognition. In the next section, a brief introduction to the classifiers selected is presented. The following section elaborates the experiments carried out. This is followed by the results and the respective discussions.

5.1 Activity recognition using supervised classifiers

The evaluations carried out in this chapter intend to investigate the following questions:

Q1 - Can we use classification algorithms to perform activity recognition using a smartphone?

Q2 - Which classifier gives the best recognition accuracy?

Q3 - Do we need to pre-process the raw accelerometer data with smoothing techniques?

Q4 - Which sampling rate should be used with the selected classifier(s)?

Q5 - Which combination of data preparations and features work best with the selected classifier(s)?

For the comparison of the features' impact on the classification accuracy, we prepared the data as mentioned in Chapter 4. Altogether 15 sets of annotated acceleration data had been pre-processed as 15 sets of training data. Classifiers were built using the respective classification learning algorithm and the training data, which were produced using different combinations of features, sampling rates and sliding window conditions. The accuracy produced by each of the classifiers was compared to provide answers to the above questions. We compared the classification accuracies with different combinations of features and sampling conditions.

Additionally, we also investigated if the combination of one or more base-level classifiers helps improve classification accuracy for the case of activity recognition using the accelerometer of a smartphone. Therefore, we included three meta-level classifiers and compared them along with the base-level classifiers.

The choices of both base- and meta-level classifiers were by no means an exhaustive selection. We wanted to compare the recognition accuracy and performance of the classifiers that were used in related work in order to understand how one can implement activity recognition using sensors (particularly the accelerometer) of a smartphone. However, it was not the goal of this thesis to analyse the best performing classifiers for all potential recognition tasks. The obtained results can only be used to best describe the recognition accuracies based on the data sets we used in this work. Wolpert mentioned in [1], what he called as "No Free Lunch Theorem", that a classification algorithm is as good as how well the algorithm's inductive bias matches the properties of the data. As long as a system is capable of reproducing similar data and obtaining patterns that are consistent with the built models using the selected classifiers, it is safe to generalise that the best classifiers, observed in the performed evaluations, should also perform equally well in a real implementation.

In the next sub-section the selected base-level and meta-level classifiers are first presented. This is followed by the evaluation results and the discussions.

5.1.1 Base-level classifiers

Decision Tree

Decision Tree (DT) is known as a simple yet widely used classification technique [2]. It uses a tree-like model of decisions and their outcomes provide the desired classifications. The resulting model has a hierarchical tree structure with a root node, the subsequent intermediate nodes and the leaf nodes. An example is shown in Figure 5.1. The root node and the intermediate nodes represent the decisions in the built tree that lead to the respective leaf nodes. The leaf nodes are the labels assigned to the object being classified.

As shown in Figure 5.1, taken from [3], the simple DT is constructed to classify the vertebrates into mammals and non-mammals. The root node *Body Temperature* separates the cold-blooded from the warm-blooded vertebrates. Since all in the former category are non-mammals, a leaf node labeled *Non-mammals* is given to all vertebrates that are cold-blooded. For the warm-blooded vertebrates, an intermediate node with the feature *Gives Birth* is created to differentiate non-mammals from mammals, due to the fact that most birds do not give birth but lay eggs. The decisions of *Gives Birth* result in the two final leaf nodes, *Mammals* and *Non-mammals*. If an animal is warm blooded and gives birth, the DT model classifies this animal as a mammal.

The Weka's implementation of Quinlan's C4.5 revision 8 in Java (J48) decision tree building algorithm is selected for the evaluation in this work. It is a Java re-

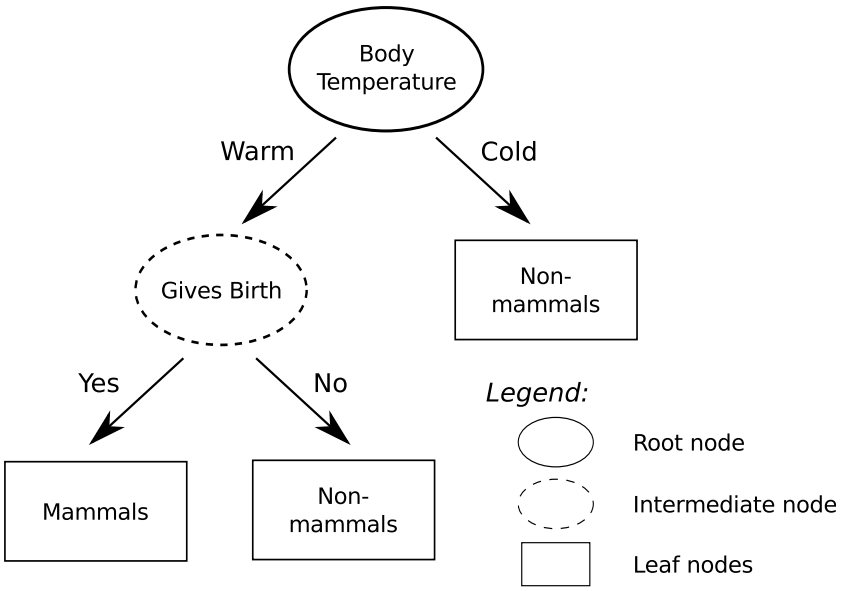


Figure 5.1: An example of a decision tree (taken from [3]).

implementation of Decision tree learning algorithm by Ross Quinlan (C4.5) version 8 algorithm by Quinlan [4] provided in the Waikato Environment for Knowledge Analysis (WEKA) workbench [5]. The building of decision trees using J48 is presented as pseudo-code in Figure 5.2:

J48 based on C4.5 by Quinlan [4]

1. Check for base cases.
2. For each attribute a:
   (a) Find the feature that best divides the training data such as information gain from splitting on a.
3. Let a_{best} be the attribute with the highest normalised information gain.
   (a) Create a decision node that splits on a_{best}
4. Recurs on the sub-lists obtained by splitting on a_{best} and add those nodes as children of node. Stop when the stopping condition is met.

Figure 5.2: The pseudo-code for the J48 Algorithm.

The J48 decision tree learner first examines the training set for the possible base cases. A base case is a stopping point where the recursion in building the decision nodes should stop, such as the number of instances or attributes left to be processed. Once base cases are selected, the algorithm starts searching for features that best divide the training data. In J48, the normalised information gain is used to select

the best attributes. The algorithm then recurs on each best attribute identified until the stopping base cases are met.

Rule-based Classifier

As the name suggests, a rule-based classifier uses a collection of "if *conditions* then *action*" rules to classify the given training data set. In a rule, the Left-hand side (LHS) represents the conditions derived from the attributes, and the Right-hand side (RHS) denotes the corresponding actions that satisfy the conditions in the LHS. The produced model contains a set of rules that generalise the training data. The test data will be examined by an inference engine using the model. In a classification, the actions are the classes of the training data.

A rule-based learner produces descriptive models as classifiers. The models are easy to interpret as the rules are comprehensible for human readers. Performance wise, the rule-based classifier is comparable to the decision tree classifier. Rule-based classifiers using ordered rules are suitable for data sets with imbalanced class distributions. It uses a separate-and-conquer algorithm to construct rules. A rule that explains a portion of the training instances separates them from the remaining instances. The algorithm recursively "conquers" these remaining instances by constructing more rules until no instances are left. One rule-based learning algorithm is the Repeated Incremental Pruning to Produce Error Reduction (RIPPER) algorithm by Cohen [6]. It applies a repeating process of growing and pruning to generate rules from a given training set. Figure 5.3 on page 69 illustrates the pseudo-code of JRip, which is a Java implementation of RIPPER provided by the Weka tool.

The description length (DL) used in Cohen's RIPPER is based on the Minimum Description Length (MDL) [7], which is also used in Quinlan's C4.5Rules algorithm [8]. The DL of a rule set is the sum of the description lengths of all the rules in the set, plus the description of instances that are not covered by the rule set. The RIPPER algorithm stops adding new rules into the rule set when its DL is 64 bits greater than the smallest DL obtained so far.

According to [9], rule learning approaches only evaluate the quality of the set of examples that is covered by the candidate rule. The decision tree learning techniques evaluate the average quality of a number of disjoint sets (one for each value of the attribute that is tested). Both techniques utilise heuristics-based techniques to determine several basic properties of a candidate rule, like the number of positigve and negative examples that it covers.

k-Nearest Neighbour

The k-nearest neighbour (KNN) classifier is an instance-based classification algorithm. It is also known as a *lazy learner* because there is no prior generalization process to create a model as compared to other classification algorithms such as decision tree or rule-based classifiers. The training data are stored and only processed

JRip - based on RIPPER by Cohen [6]

1. Initialise an empty rule set, R.
2. For each class, starting from the less prevalent one to the more frequent one:
   (a) Building stage - repeat the following steps until the Description length (DL) of R and the examples is 64 bits greater than the smallest DL obtained so far:
      i. Incrementally prune each rule and allow the pruning of any final sequences of the conditions with a given pruning metric.
      ii. Grow one rule by adding conditions to the rule until the rule is 100 % accurate.
      iii. Add rules to the rule set R.
   (b) Optimization stage:
      i. For each R_i in R, construct two alternative rules. One rule variant is generated from an empty rule while the other by greedily adding conditions to the original rule R_i. The rule variant with the smaller DL is selected as the final rule for each R_i.
      ii. If there are still residual positives, more rules are built using Building Stage (step 2a) with these residual positives.
      iii. Remove rules from the rule set that would increase the DL of the rule set.
      iv. Add the resultant rule set to R.

Figure 5.3: The pseudo-code for the JRip Algorithm in the Weka Tool.

when a new instance needs to be classified. Instead of using all training data, only a selection of instances are used to create a local model for the classification. The KNN algorithm is illustrated as pseudo-code in Figure 5.4 on page 70.

The KNN algorithm computes the distance between the test data and all available training data to obtain a list of nearest neighbours (with the total of k). Common distance metric used in KNN are such as Euclidean, Manhattan or Minkowsky. The choice of the number of neighbours (k) and the distance metric depends on data set to be classified. Idealistically, the choices should produce the highest accuracy for the given test data set. For a discrete-valued target function, the algorithm classifies the test data to the majority class. For a real-value target function, the mean of the nearest neighbours will be computed as output. This approach is suitable for data sets with a large number of training data. The training phase of KNN algorithm is

k-nearest neighbour (KNN)

1. Select the number of neighbours to be searched, k and the distance metric.
2. For each testing instance:
   (a) Find the k nearest instances in the training set according to the selected distance metric.
   (b) Return the most frequent class label in the found k nearest instances as the classified class.

Figure 5.4: The pseudo-code for the KNN algorithm.

considered as very fast because it merely stores the training data in memory without performing any further step (hence the name lazy learner). Since all training will be used to be compared with the test data, there is also no loss of information.

One of the drawbacks of the KNN algorithm is the costly computation during query time, especially when the training data is large. Since no generalised model is available, the classification of test data requires the processing of all training data. Another disadvantage of the KNN algorithm is the inclusion of all attributes during classification. Large number of irrelevant attributes will affect the computed distance between the test data and the training data. Nevertheless, there exist techniques to overcome these drawbacks. The indexing of training data can potentially speed up the classification process. One of such method is the k-Dimensional tree (KD-Tree), where the search for nearest instances is optimised by limiting it to a certain partition in the training set. For the case of irrelevant attributes, one can weight each attribute differently during the distance calculation between instances. In this way, less relevant attributes will not affect the resulted total distance and thus potentially provide a more accurate classification result.

Probabilistic classifiers - Naïve Bayes and Bayesian Network

In cases where the relationship between the attribute set and the class variable are non-deterministic, probabilistic-based approach such as Bayes theorem based classifiers can be applied to make predictions for a given data set. Bayes theorem can be expressed in Equation 5.1:

$$P[H|E] = \frac{P[E|H]P[H]}{P[E]} \tag{5.1}$$

In this equation, H is the hypothesis and E is the evidence. $P[H]$ is the prior probability of the hypothesis H and $P[E]$ is the prior probability that the data E will be observed. $P[E|H]$ is the conditional probability of E given H (also known as the

likelihood) and $P[H|E]$ is the posterior probability, which denotes the probability of H given E.

A Naïve Bayes (NB) classifier computes the class-conditional probability with an assumption where the attributes for a given class label in a data set are conditionally independent. Based on the Bayes Theorem (Eq. 5.1), the NB classifier computes the posterior probability for each class H:

$$P[H|E] = \frac{\prod_{i=1}^{d} P[E_i|H]P[H]}{P[E]} \tag{5.2}$$

The NB classifier is considered as a simple yet efficient algorithm [10]. It is suitable for a moderate or large set of training data. In training data where redundant and non-conditionally independent attributes are found, the NB classifier may not perform well [10] [3]. In such cases, one can consider the use of the Bayesian Network (BN) classifier.

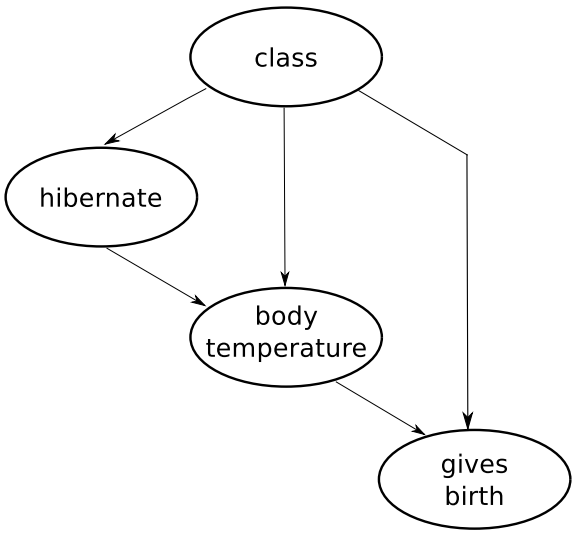


Figure 5.5: An example of a Bayesian network graph using the mammals data set example.

A BN describes a probability-based system by specifying relationships of conditional dependencies between its variables. The conditional dependencies are represented by a directed acyclic graph (DAG), in which, each node represents a variable and the arcs represent relationships between variables. This graph creates a model that can be used for the classification of the test data. Figure 5.5 is an example of a BN graph model built from the simple mammals example used for DT on page 66. As an example, if given the an unknown animal that gives birth and is warm-blooded, we can calculate the probability of mammals and non-mammals based on the conditional probability relationships between the classes and the two nodes *body temperature* and *gives birth*. The class with the highest probability distribution is

selected as the output class. The learning of a BN graph is presented in Figure 5.6 as pseudo-code.

Bayesian Network (BN)

1. Develop a DAG G where X is believed to satisfy the local Markov property with respect to G.
2. Ascertain the conditional probability distributions of each variable given its parents in G.

Figure 5.6: The pseudo-code for the BN algorithm.

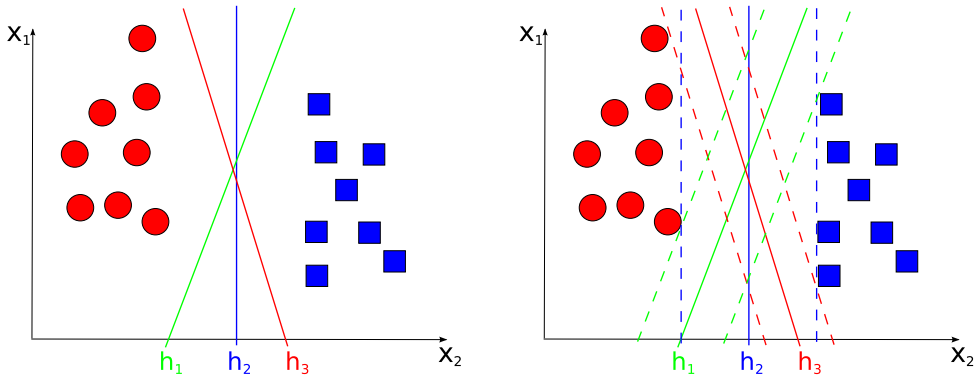
The major advantage of the NB classifier is the speed for model building. It is robust to isolated noise instances in the data set because the estimation of conditional probabilities from data averages out these instances. The class-conditional probability of the irrelevant features has no influence on the overall computation of the posterior probability. This makes NB classifiers to be robust to the irrelevant features. However, if the features in a given data set are correlated, the performance of NB can be negatively affected.

The BN classifier provides a graphical representation of the built model. It is able to handle incomplete data sets and provides an efficient method for preventing over-fitting of data. For data set that are correlated, BN can be used instead of NB since the former does not require the features in a data set to be independent. BN may also be more attractive than NB for the DAG representation of the model.

Support Vector Machine and Sequential Minimal Optimization

Support vector machine (SVM) [11] uses statistical learning to perform classification. The basic idea behind SVM is to find hyperplanes in linearly separable data. A hyperplane can be seen as a linear function that separates the data into two groups in space. A depiction of this basic idea is shown in Figure 5.7a. The hyperplane is able to tell apart that the circles are on the left-hand side, and the squares are on the right-hand side.

In the training phase, the SVM algorithm attempts to find the best hyperplane called the maximal margin hyperplane. As shown in Figure 5.7a, there are many possible hyperplanes that will separate the data into the corresponding classes. The maximal margin hyperplane represents the best hyperplane that gives the largest margin between the two classes. In other words, the maximal margin hyperplane gives the largest separation between the two classes. This is shown in Figure 5.7b. For the given example, among the three possible hyperplanes h_1, h_2 and h_3, the hyperplane h_2 is seen as the maximal margin hyperplane because h_2 has the widest margin for the linear separation of the two classes of data.



(a) Three possible hyperplanes (h_1,h_2 and h_3) for a given data set with two groups - circles and squares

(b) The margins for each hyperplane

Figure 5.7: An illustration of the basic idea of SVM and maximal margin hyperplane.

There are many varieties of SVM implementations, and we have chosen the Sequential Minimal Optimization (SMO) algorithm [12] to be used in our experiments as a SVM-based classification algorithm. The SMO algorithm divides the general Quadratic programming (QP) problem into QP sub-problems using Osuna's theorem [13] to ensure convergence. In this way, it solves the QP problem in SVM quickly without the need of any additional matrix storage and without using numerical QP optimization steps. Hence, very large SVM training problems are solvable with relative smaller memory space using SMO and the speed is improved. The SMO algorithm is also known to be less susceptible to numerical precision problems [12].

Discussion

The DT classifiers are suitable for data instances that are describable with attribute-value pairs. It uses a divide-and-conquer strategy to discover the relationship between the attributes and the respective class. The produced tree can also be viewed as rules. DT is seen as an attractive classification algorithm because it can produce (relatively) easy to understand models. This enables users or domain experts to view and review the produced models. It is also an unstable classifier, because a small change in the training data set may cause a big change in the produced tree model. It tends to perform better with categorical features, though it supports both categorical and continuous features.

Similar to decision tree, rule-based classifiers are comprehensible models. This is commonly seen as an advantage over the approaches that produce complex and incomprehensible models. It is also a relative fast classifier, though the training process for rule learning may take longer time as compared to the time needed for decision tree learning. A rule-based classifier consists of rules that are more compact than trees built from the same training set using decision tree. The ability to add

new rules to an existing rule set can be seen as an advantage over decision tree, since, for the latter, a reconstruction of tree models may be required when new trees are introduced.

The KNN classifier is also often investigated in the area of activity recognition using accelerometer data. It is seen as an attractive classifier due to the accuracies achieved in these investigations. The classification speed is a potential issue especially if it is executed on mobile devices. Another classifier that is commonly tested together with other classification algorithms is the NB classifier. The BN classifier, on the other hand, was not investigated. In this thesis, both the NB and BN classifiers were included to investigate whether they are also suitable for the intended unobtrusive activity recognition. As observed in some of the investigations ([14], [15], [16]) the NB classifier had produced lower recognition accuracies as compared to KNN and DT. The addition of BN allows us to exclude the possible influence of dependency between features in the designated classification.

In the recent years SVM-based classification algorithms have gained popularity in solving problems in different domains, where they have performed better than other classification algorithms. However, when compared to algorithms such as decision tree or rule-based learner, the SVM-based algorithms are slow, especially when applied in a nonlinear setting [10]. The selection of SMO may be a solution in terms of speed, and we will compare the performance of SMO with other classifiers for the intended activity recognition in the evaluations.

Besides the comparison of the above base-level classifiers, it is also possible to combine more than one classifier to obtain potential improvement in accuracy. The next sub-section describes three meta-level classifiers that enable such an ensemble of classifiers, which we have selected for the planned evaluations and comparisons.

5.1.2 Meta-level classifiers

Besides the comparison with the base-level classifiers, we have also investigated the recognition accuracy using meta-level classifiers. Meta-level classifiers are techniques that combine multiple base-level classifiers to perform the designated classification. A meta-level classifier is sometimes also known as an ensemble method. The general idea of the technique is to improve the classification accuracy by aggregating the prediction of the multiple classifiers [3].

A meta-level classifier can be in an ensemble of more than one of the same base-level level classifier. It is also possible to select more than two different base-level classifiers and build a meta-level classifier based on these base-level classifiers. There are generally four different approaches to construct an ensemble of classifiers [3]:

- The manipulation of the training set
  In this approach, the training set is re-sampled to create multiple training sets according to selected sampling distribution. A particular learning algorithm is then used to build classifiers from all the newly created training sets.

- The manipulation of the input features
  This approach creates multiple training sets from the original training set by selecting a subset of the input features/attributes. The selection can be made randomly or be decided base on the recommendation of the domain experts.
- The manipulation of the class labels
  In this approach, the class label of the training data is randomly partitioned into two disjoint subsets. The training data is then transformed into a binary class problem where class 0 and 1 are assigned to the training instances according to the two subsets respectively and used to train a base-level classifier. This step is repeated multiple times to create an ensemble of base-level classifiers. The prediction of a test instance is carried out for all the base-level classifiers and is used to add a vote to all the classes that belong to the corresponding subset. The votes are later tallied to assign the class with the highest vote to the test instance.
- The manipulation of the learning algorithm
  It is also possible to manipulate the learning algorithm so that, in the repetition of the learning process, different classifiers can be obtained from the same set of training data. An ensemble of classifiers is built using these classifiers and is used for the designated prediction by aggregating the predictions of each underlying classifier.

Theoretically, the accuracies obtained by using meta-level classifiers are expected to be better than by using just base-level classifiers. For example, by giving higher weight to a better performing classifier in an ensemble may improve the prediction accuracy. Errors that occur in the training process can also be reduced by aggregating the base-level classifiers built using different re-sampled training sets, especially when the selected base-level classifiers are sensitive to minor variations in the training set.

Selected classifiers are bagging, boosting and voting. These classifiers were used in the investigations of [17] and gave high accuracies. Both bagging and boosting classifiers use one base-level classifier at a time. Voting combines two or more different base-level classifiers for the intended classification. The brief overview on each meta-level classifier is as follows:

Bagging

Bagging is short for **b**ootstrap **agg**regat**ing**. It is proposed by Leo Breiman of University of California in 1994 [18]. The basic idea of bagging is to create an ensemble of classifiers using bootstrap replicates of the given training set. The bootstrap process generates a certain number of smaller samples randomly re-sampled from the given training set. The output of a bagging classifier is the combined voting result of these classifiers. Bagging is able to improve classification accuracy of unstable base-

Bagging

Input:

Training set S consisting a sequence of N instances
$< (x_1, y_1), \ldots, (x_N, y_N) >$ with labels $y_i \in Y = \{1, \ldots, k\}$
Number of iteration, T

Training phase:

1. For $t = 1, 2, \ldots, T$:
   - Take a bootstrap sample S'_t from the training set S.
   - Build a classifier C_t using S'_t as training set.
   - Add the classifier C_t to the ensemble, $C = C \cup C_t$
2. Return C for the designated classification.

Classification phase:

1. Given the test data x, run the classifiers $C_1, \ldots, C_T$
2. For numerical outputs, an average is computed from the output of the classifiers in C. For categorical outputs, the highest number of output classes obtained from all classifiers in C is selected as the output class.

Figure 5.8: The Bagging algorithm.

level classifiers, such as decision trees and neural networks [18], [2]. These base-level classifiers are considered unstable because they may have a large variance in the probability of misclassification due to small training set [19]. Small changes in the training set lead to large changes in unstable classifiers. In cases where the majority of the classifiers built from the bootstrap replicates give accurate classification, the bagging meta-level classifier may perform better than the use of corresponding base-level classifier alone.

Figure 5.8 shows the bagging algorithm in both training and classification phases. During the training phase, Given a training set, S, bagging generates n new bootstrap replicates of S'. From the bootstrap samples S'_1 to S'_n, the classifiers C_1 to C_n are built. In the classification phase, the classifiers C_1 to C_n are used to classify an input x. The output of these classifiers is then averaged (for regression) or voted (for classification).

The bagging meta-classifier may be useful for the classification of activities using acceleration data if it improves classification accuracies with a relatively small training set. Another potential advantage of using the bagging meta-classifier is the

possibility to train and run the classifier ensembles on different processors or devices [2]. It may also reduce classification error due to misleading data in the training set and avoid overfitting [20].

Boosting

Boosting applies a base-level classifier repeatedly and attempts to improve the classification accuracies by changing the weights assigned to the objects in the training set in every repetition. Initially, all objects in the training set have equal weights. A classifier is built using this training set. The result of this first classifier is used to adjust the weights of these objects. Objects in the training set that are wrongly classified get higher weights. The next classifier is built using the "boosted" training set. With the re-weighting on the training set in each repetition, subsequent classifiers are then built. In the classification phase, these classifiers are used to classify the test set where each output of the classifiers is weighted. These weighted outputs are combined to produce the classification output.

A popular boosting algorithm is the AdaBoost (short for **Ada**ptive **Boost**ing) algorithm. It is introduced by Freund and Schapire [21] [22]. AdaBoost adjusts adaptively to the error of the hypotheses returned by the selected base-level classifier. In this work, the AdaBoost.M1 algorithm is used for comparison. Figure 5.9 shows the AdaBoost.M1 algorithm as presented by Freund and Schapire in [22]. Given a training set with x_i as the attributes that lead to the corresponding outcomes y_i, the first weight vector w^1 is initialised. A distribution p is computed by normalizing these weights. This distribution is then fed to the learner L to obtain the hypothesis h. The requirement in AdaBoost.M1 is that each hypothesis needs to have a prediction error less than $1/2$ with the respective distribution. Based on this hypothesis, the weight vector of w^{t+1} for the subsequent iteration is computed. The same process is repeated until the final iteration T. The final hypothesis h_f is the label y that has the maximum sum of the weights of the hypotheses predicting that label.

AdaBoost is sensitive to noisy data and outliers. Due to the iteration and the recalculation of weights or re-sampling, AdaBoost may be slow during the training phase. However, in certain cases it can be less susceptible to the overfitting problem than most learning algorithms [22]. It has been widely investigated in various domains. It is seen as a good technique to improve classification accuracy of a given base-level classifier.

Voting

The meta-level classifier *voting* combines multiple base-level classifiers to perform the intended classification. These base-level classifiers $(C_1, C_2, \ldots, C_N)$ are generated on a given training data set using selected learning algorithms. A voting scheme is selected to produce the classification. The simplest voting scheme is the majority

AdaBoost.M1

Input:

training set, S consisting a sequence of N instances
$< (x_1, y_1), \ldots, (x_N, y_N) >$ with labels $y_i \in Y = \{1, \ldots, k\}$
distribution D over the N instances, number of iteration, T

Training phase:

1. initialise the weight vector: $w_i^1 = D(i)$ for $i = 1, \ldots, N$.
2. For $t = 1, 2, \ldots, T$:
   (a) Compute the distribution, $p^t = \dfrac{w^t}{\sum_{i=1}^{N} w_i^t}$
   (b) Call the classifier C_t with the distribution p^t; Obtain a hypothesis $h_t : X \to Y$. Adds C_t to the ensemble C.
   (c) Calculate the error of $h_t : \varepsilon_t = \sum_{i=1}^{N} p_i^t [\![h_i(x_i) \neq y_i]\!]$
   if $\varepsilon > 1/2$, then set $T = t - 1$ and abort loop.
   (d) Set $\beta_t = \dfrac{\varepsilon_t}{1 - \varepsilon_t}$.
   (e) Set the new weights vector to be: $w_i^{t+1} = w_i^t \beta_t^{1-[\![h_t(x_i) \neq y_i]\!]}$
3. return C and $\beta_1, \ldots, \beta_T$.

Classification phase:

1. Given the test data x, run the classifiers $C_1, \ldots, C_T$
2. The final hypothesis is computed:

$$h_f(x) = \arg \max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) [\![h_t(x_i) = y]\!]$$

The output class is the label with maximum sum of the weights of the hypotheses predicting that label.

Figure 5.9: The AdaBoost.M1 algorithm, adapted from [22].

vote, where the output class with the highest frequency or votes is selected as the output for a given test instance [23]. The base-level classifiers are either generated using different learning algorithms or a selected learning algorithm with different

parameters. The goal of using the meta-level classifier *voting* is to improve the classification accuracy of a single base-level classifier by combining the outputs of the multiple base-level classifiers.

There are several techniques to vote for the desired output class. Among them are majority vote, average of probability, product of probability and maximum/minimum probability. The *voting* classifier provided in Weka tool uses the computation of the class probability distributions predicted by the underlying base-level classifiers for the intended classification. The average probability distribution method is selected in this thesis. For a given test instance, each base-level classifier returns the respective probability distribution vectors. The class with the highest average probability is selected as the output class for the respective test instance.

Discussion

The classification methods are commonly used in the past investigations using acceleration data for activity recognition. However, to the best of our knowledge, base-level classifiers are mostly used and compared in these investigations. Investigations such as [17], [24] and [25] have investigated the use of meta-level classifiers to recognise activities based on acceleration data. Their results have shown that meta-level classifiers generally perform better than base-level classifiers.

All three selected meta-level classifiers have shown classification improvements in various investigations. Both bagging and boosting improve classification performance of a single base-level classifier by iterating through classifiers built using respective techniques. However, this may also increase the classification duration depending on the selected base-level classifier and number of iterations. Voting provides classifications by combining different base-level classifiers. It is expected to be faster than bagging and boosting and has shown advantages over the other two selected meta-level classifier in [17]. In this chapter, evaluations are carried out to compare the classification performances of the selected base- and meta-level classifiers.

As we look into the use of meta-level classifiers for activity recognition using a smartphone, we want to investigate whether the recognition accuracy of meta-level classifiers is comparable or better than the accuracies of the respective base-level classifiers. Additionally, these meta-level classifiers should provide equivalent performance in execution, since the ensemble of multiple base-level classifiers may also means longer execution time and higher resources (such as memory and processing power) as compared to a single base-level classifier.

5.2 Evaluations

As mentioned in Chapter 4, the acceleration data with the respective movement annotations are processed to produce the training data needed for the planned ex-

periments.

There were altogether two main categories of evaluation. Firstly, classification evaluations were performed to compare performances of different classifiers and the possible selection of the pre-processing parameters. The annotated acceleration data was used as the input data for the evaluations. Based on the results, a short-list of recommended classifiers were chosen for further evaluations. The selected evaluation methods are elaborated in the sub-section. Secondly, evaluations were also carried out to investigate the performance of the recommended classifiers. Performance factors included durations of pre-processing and classification processes as well as the influence on battery life. The second categories of evaluation are presented in Chapter 6.

Following are the list of evaluations performed in this chapter:

- A comparison of classification accuracy using the selected base- and meta-level classifiers based on acceleration data obtained from smartphones.
- A comparison of classification accuracy using selected classifiers based on acceleration data from a smartphone and a dedicated accelerometer sensor device.
- A comparison of the influence of data smoothing has on the classification accuracy using selected classifiers.
- A comparison of the influence of different pre-processing parameters has on the classification accuracy using selected classifiers.
- A comparison of the influence of selected features has on the classification accuracy using selected classifiers.

5.2.1 Evaluation methodology

In order to compare the classification accuracy of the selected algorithms, the following evaluation techniques have been selected and applied. They are commonly used in classification evaluations and comparisons [26] [27] [2] [5].

Classifier accuracy evaluation using the 10-fold cross-validation method

The 10-fold cross-validation method [28] is used for comparisons of the classification accuracy. This evaluation method is commonly used to provide an estimation on how well the models will perform generally. The training data are randomly partitioned into 10 sub-samples. For every round of evaluation, one sub-sample is used as the test data while the other nine sub-samples are used as training data. The cross-validation process is repeated 10 times and the produced evaluation results are averaged as the model's performance estimation. The accuracy value used for comparison is the percentage of the number of correctly classified instances as compared to the total

number of instances (see Equation 5.3).

$$\text{accuracy} = \frac{\text{number of correctly classified instances}}{\text{total number of instances}} \times 100\% \tag{5.3}$$

There were 15 sets of acceleration data recorded by the test users. Comparisons were made based on the average classification accuracy from the selected evaluations, denoted by $\bar{x}$. Besides the 10-fold cross-validation evaluation, there are also comparisons made using the hold out method. Certain portion of the acceleration data were extracted as training data, which were used to generate the designated classifiers. The rest of the acceleration data were used as test data to evaluate the performance of the generated classifiers. Similar to the 10-fold cross-validation method, the hold out evaluations carried out in this thesis were repeated 10 times. In each repetition, the acceleration data was reshuffled and then separated into training and test data. This was done to estimate how well a classifier, generated from a given proportion of the accelerated data, may perform in a real situation, provided that the input acceleration data are similar to the given test data used in the hold out evaluation.

Both evaluations are nevertheless regarded as estimations of performance for the selected classifiers. The estimations were valid for the acceleration data collected from the 15 test users. In cases where the estimations were very close to one another, the following statistic method is applied to analyse the observed values and their differences.

Accuracy comparison using the statistic significant test

The paired Wilcoxon signed-rank test [29] was applied in the evaluations. This technique helps to investigate whether the differences between comparison factors are significant. The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test. It is used to test two related samples or repeated measurements of a given sample by calculating the differences of the pairs. The absolute differences obtained are ranked after discarding pairs with the difference of zero. These ranks are then sorted in an ascending order. Each of these several pairs with absolute differences that are equal to each other is assigned as the average of ranks. The hypothesis is that the differences have the mean of 0. We have selected the Wilcoxon signed-rank test instead of paired t-test [30] because the nonparametric procedures in the Wilcoxon signed-rank test make less stringent demands than the paired t-test on the data. It does not have the requirement where the population needs to be normally distributed.

A statistical hypothesis test helps to estimate whether the mean values to be compared are the same. We apply the Wilcoxon signed-rank test in our comparisons in order to test the differences in the obtained average accuracies of the classification performed on acceleration data of all the 15 test users. In the previous investigations in activity recognition, only direct comparisons of the accuracies are made to arrive at the conclusions. The use of statistical hypothesis test, such as the Wilcoxon

signed-rank test, can be used to provide a better estimation and comparison to draw the conclusion, whether a classifier performs better than the others. Due to the relative small number of test users, the conclusions drawn using the Wilcoxon signed-rank tests in this thesis are regarded as estimations, valid for the data used in this thesis. These estimations should reflect how well certain classifiers performed as compared to one another, based on the annotated acceleration data collected from all 15 test users.

In this thesis, the null hypothesis Ho used is that the two accuracies are not significantly different. The alpha level, α, is set at 0.05. If the calculated p-value is greater than 0.05, we accept the hypothesis H_o. Otherwise, we reject the H_o and conclude that the differences of the two accuracies, denoted by Δ, are statistically significant. The evaluations and the respective results are presented in the following sub-sections.

5.2.2 Classification of movement using with base- and meta-level classifiers and raw acceleration data

First and foremost, we wanted to investigate how well could the accelerometer in a smartphone be used for movement recognition similar to the dedicated accelerometer approaches. The accelerometer data, collected from all test users, were pre-processed using standard conditions - sampling rate of 32 Hz, window length of 4 seconds and 50 % of overlap. In some previous investigations, such as [14], [17], [31], [32] and [33], the combination of 4 seconds window length and 50 % overlap was selected. Therefore, we had selected the same window length and overlap percentage in order to compare the results with some of the previous investigations. The only difference in the acceleration samples was the sampling rate. From the measurements performed, it is observed that the selected smartphones did not deliver acceleration changes higher than 35 Hz to 40 Hz. Therefore, 32 Hz was taken as the best possible sampling rate for the smartphone used in the experiments. All these investigations except [33] had used higher sampling rates (from 50 Hz to 93 Hz). Therefore, we also wanted to investigate whether the sampling rate of 32 Hz was sufficient and could provide equivalent accuracies as compared to the dedicated accelerometers approaches in these investigations.

The features used in this comparison were both simple and FFT-based features. These were mean, standard deviation and variance of the accelerometer data as well as energy and information entropy of the FFT transformation of the accelerometer data. Both base- and meta-level classifiers were built using all five features similar to the previous investigations using single or multiple accelerometers for activity recognition (see Table 5.1 on page 83).

Table 5.2 on page 83 shows the average accuracy of the 10-fold cross-validation evaluation for all 15 test users. The KNN classifier had the highest average accuracy with 91.68 %, followed by the DT and SMO classifiers with 91.18 % and 91.12 %

Table 5.1: List of base- and meta-level classifiers used in the evaluations.

	Meta-level classifiers		
Base-level classifiers	Bagging	Boosting	Voting
K-nearest neighbour (KNN)	with KNN	with KNN	with KNN + SMO
Decision Tree (DT)	with DT	with DT	with KNN + DT
Bayesian Network (BN)	with BN	with BN	with KNN + NB
Naive Bayes (NB)	with NB	with NB	with DT + NB
Ripper - Rule Learner (JRip)	with JRip	with JRip	with DT + SMO
Sequential Minimal Optimization (SMO)	with SMO	with SMO	with SMO + NB

Table 5.2: Evaluation result of movement recognition using base-level classifiers, with $N = 15$, and average instances total = 1569.60.

Base-level Classifier	Average Accuracy (%)	Std. Deviation (%)
BN	87.68	3.44
KNN	91.68	2.46
DT	91.18	2.77
JRip	90.92	2.91
NB	85.15	6.19
SMO	91.12	2.68

Table 5.3: The Wilcoxon signed-rank test for all base-level classifiers.

		Base level					
		BN	KNN	DT	JRip	NB	SMO
Base level	BN	-	**0.00**	**0.00**	**0.00**	0.21	**0.00**
	KNN	**0.00**	-	**0.02**	**0.00**	**0.00**	0.52
	DT	**0.00**	**0.02**	-	0.27	**0.00**	0.52
	JRip	**0.00**	**0.00**	0.27	-	**0.00**	0.42
	NB	0.21	**0.00**	**0.00**	**0.00**	-	**0.00**
	SMO	**0.00**	0.52	0.52	0.42	**0.00**	-

respectively. Bayesian-based classifiers had among the lowest accuracies. We used the two-tail paired Wilcoxon signed-rank test to compare the accuracies of the base-level classifiers obtained from the 10-fold cross-validation evaluations. The p-values from the Wilcoxon signed-rank test at $\alpha = 0.05$ are listed in Table 5.3 on page 83. A **bold p-value** indicates that it is smaller than or equal to 0.05. Among the

base-level classifiers, the classifier SMO had shown no significant differences with the classifiers KNN, DT and JRip. Similarly, the classifier DT had also no significant difference with the classifier JRip. The same observation applied to the classifiers BN and NB too. The differences between two classifiers that are marked **bold** in Table 5.3 were seen as significant according to the Wilcoxon signed-rank test at $\alpha = 0.05$.

In other words, the cross-validation results of KNN was better than the other base-level classifiers except SMO. The differences between KNN and DT or JRip were significant, but were within a relatively small range (up to 0.76 %). Therefore, we considered these four classifiers as the best among the selected base-level classifiers. The BN and NB classifiers had significant differences in accuracy when compared to the other four best classifiers. The differences were up to 6.53 % when compared to the KNN classifier.

Table 5.4: Confusion matrix of the cross-validation evaluation for test user #3, classifier = DT, sampling rate = 32 Hz, window length = 4 seconds, overlap = 50 %.

		Classified as				
		a	b	c	d	e
Ground truth	a = Go upstairs	137	42	0	6	4
	b = Walking	31	562	1	6	15
	c = Sitting	0	1	279	17	0
	d = Standing	5	4	15	309	4
	e = Go downstairs	7	20	1	2	148

As we took a closer look at the classification results, not all activities had equally high recognition accuracy. As an example, Table 5.4 on page 84 shows the confusion matrix of the 10-fold cross-validation evaluation results for the user #3. The recognition accuracy for activity *standing*, *sitting* and *walking* using base-level classifiers was higher than 91 %. The confusion matrix could be used to understand how well a certain classifier had performed. For example, by comparing the ground truth and the classification outcomes for each movement, 309 instances of *standing* were correctly classified, while others were misclassified as *go upstairs* (5 instances), *walking* (4 instances), *sitting* (15 instances) and *go downstairs* (4 instances). Generally, the movements *go upstairs* and *go downstairs* were often misclassified as *walking*. There was also a tendency where a small number of *sitting* and *standing* were misclassified as each other. Some of these misclassifications took place because of the transition between movements and possible human annotation mistakes.

The 10-fold cross-validation evaluation for the meta-level classifiers are listed in Table 5.5 on page 85 and Table 5.6 on page 85. The meta-level classifiers Boosting and Bagging produced higher accuracies than the corresponding base-level classifiers. As highlighted **bold** in Table 5.5, meta-level classifiers Boosting with BN, DT

Table 5.5: Evaluation result of movement recognition using meta-level classifiers, with total test users, $N = 15$, and average total instances = 1569.60.

Meta-level Classifier	Base-level Classifier	$\bar{x}_{meta}$	$\Delta\bar{x}_{meta-base}$ (%)	p-value
Boosting	BN	**89.71**	**2.03**	**0.00**
	KNN	91.81	0.35	0.08
	DT	**92.53**	**1.36**	**0.00**
	JRip	**92.46**	**1.54**	**0.00**
	NB	85.51	6.56	0.13
	SMO	93.55	2.88	0.50
Bagging	BN	**88.14**	**0.69**	**0.00**
	KNN	**92.56**	**0.88**	**0.00**
	DT	**92.54**	**1.36**	**0.00**
	JRip	**92.53**	**1.61**	**0.00**
	NB	85.55	0.05	0.52
	SMO	91.37	0.08	0.34

Table 5.6: Evaluation result of movement recognition using meta-level classifiers, with total test users, $N = 15$, and average total instances = 1569.60.

Meta-level Classifier	Base-level Classifier	$\bar{x}_{meta}$	$\Delta\bar{x}_{meta-base_1}$ (%)	p_{base1}	$\Delta\bar{x}_{meta-base_2}$ (%)	p_{base2}
Vote	KNN + NB	**89.97**	**-1.70**	**0.00**	**4.83**	**0.00**
	KNN + SMO	**92.41**	**0.91**	**0.00**	**1.51**	**0.00**
	DT + KNN	**91.89**	**0.71**	**0.00**	0.21	0.25
	DT + NB	**89.84**	**-1.33**	**0.00**	**5.03**	**0.00**
	DT + SMO	90.80	0.10	0.25	0.09	0.52
	SMO + NB	**85.50**	**-5.62**	**0.00**	**0.41**	**0.00**

Legend: $\bar{x}$ = average accuracy, Δ = difference,
p_{base1} = p-values from the Wilcoxon signed-rank test comparing a meta-level classifier with its first base-level classifier,
p_{base2} = p-values from the Wilcoxon signed-rank test comparing a meta-level classifier with its second base-level classifier

and JRip as well as Bagging with BN, KNN, DT and JRip had improvements in recognition accuracies that were significant according to the two-tail paired Wilcoxon signed-rank test at $\alpha = 0.05$. Both Boosting and Bagging with DT and JRip performed better than classifiers built using KNN and SMO. The rest of the Boosting and Bagging meta-level classifiers were considered equally good as the respective base-level classifiers.

For the meta-level classifiers Vote, only Vote with KNN + SMO had produced higher accuracies than the base-level classifiers KNN and SMO respectively. Vote

with DT + KNN also produced significant improvement of 0.71 % as compared to the base-level classifier DT. The use of Vote with KNN, DT or SMO together with NB improved the recognition accuracies as compared to the classifier NB alone, but they were worse than the base-level classifiers KNN, DT or SMO respectively. In other words, there was no advantage to combine the classifier NB with other base-level classifiers selected in this thesis to build the respective meta-level classifier Vote.

Discussion

The results had shown that the acceleration data together with the chosen sampling and pre-processing choices (sampling rate 32 Hz, window length 4 seconds and 50 % overlap) provided accuracy from 85.15 % (NB) up to 92.56 % (Bagging with KNN). Most of the meta-level classifiers had shown improvements that were statistically significant, when compared with the accuracies of the respective base-level classifier.

From the above analysis, we concluded that the recommended classifier choices for the recognition of basic activities as listed in Table 5.7.

Table 5.7: Recommended Classifiers.

Base-level Classifiers	**Meta-Level Classifiers**
KNN, DT, JRip and SMO	Boosting - with DT and JRip
	Bagging - with KNN, DT and JRip
	Vote - with KNN + SMO and DT + KNN

The recommended base- and meta-level classifiers had provided accuracy averagely higher than 91 %, for all 15 training data sets.

The results have shown that with only an accelerometer of a smartphone, these classifiers provided comparable accuracies achieved in related investigations.

5.2.3 Comparison between classification of acceleration data from a smartphone and a dedicated sensor

From the previous sub section, it was observed that the average accuracy of the 10-fold cross-validation evaluation was around 90 %. In order to justify whether the accuracies are acceptable and comparable to an approach using dedicated accelerometer, an experiment was carried out with the test person performing the basic activities using a Sun SPOT device and a smartphone. The Sun Sun Small Programmable Object Technology (SPOT) [34] is a compact wireless sensor network device developed by Sun Microsystems. It has a 180 MHz 32bit ARM920T core with 512K RAM. The sensors available on a Sun SPOT device are a tri-axial accelerometer, temperature sensor and light sensor.

Table 5.8: Comparison between the accuracies obtain from acceleration data of a smartphone and a Sun SPOT.

Meta-level Classifier	Base-level Classifier	Average Accuracy,$\bar{x}$ (%)			
		$\bar{x}_{Smartphone}$	$\bar{x}_{SunSPOT}$	$\bar{x}_{SunSPOT} - \bar{x}_{Smartphone}$	p-value
-	BN	82.83	82.86	0.03	0.99
	KNN	86.96	88.15	1.20	0.77
	DT	86.30	87.27	0.97	0.63
	JRip	86.90	87.48	0.59	0.44
	NB	82.49	83.81	1.32	0.73
	SMO	89.63	89.24	-0.39	0.77
Boosting	BN	86.96	88.12	1.16	0.68
	KNN	87.97	89.37	1.40	0.72
	DT	88.15	89.05	0.90	0.80
	JRip	89.02	89.80	0.78	0.74
	NB	82.49	83.81	1.32	0.73
	SMO	89.63	89.24	-0.39	0.77
Bagging	BN	82.61	83.77	1.16	0.75
	KNN	88.01	89.56	1.55	0.60
	DT	88.48	89.09	0.61	0.77
	JRip	88.65	89.25	0.60	0.64
	NB	82.58	83.87	1.29	0.74
	SMO	89.48	89.26	-0.22	0.88
Vote	KNN + NB	85.71	86.96	1.25	0.67
	KNN + SMO	88.26	90.01	1.74	0.62
	DT + KNN	87.34	88.39	1.05	0.67
	DT + NB	85.39	87.14	1.75	0.63
	DT + SMO	86.34	87.32	0.99	0.64
	SMO + NB	82.99	84.19	1.20	0.76

For this comparison, the Sun SPOT was placed together with the smartphone in the trouser pocket during the experiment phase. This ensured the accelerometers on both devices measured the same movement during the experiment. Similarly, the N800 Internet Tablet was used for the annotations of the performed movements. The recorded acceleration data from both devices were processed using the same procedures as mentioned in the sub section 5.2.2. Five features (mean, variance and standard deviation of the raw acceleration data, as well as the energy and information entropy of the FFT coefficients) were extracted from the acceleration

data and then used in the 10-fold cross-validation evaluations. The results for the evaluations using base- and meta-level classifiers, built with a sampling rate of 32 Hz, window size of 4 seconds and overlap percentage of 50 %, are listed in Table 5.8 on page 87.

In order to compare the differences between the accuracies obtained using acceleration data from both devices, the paired Wilcoxon signed-rank test method was applied. The column *p-value* of the Table 5.8 represents the two-tailed p-value at $\alpha = 0.05$ for the corresponding base- and meta-level classifiers. As shown in this table, the p-values were all greater than 0.05. Therefore, the differences between the accuracy produced by both devices were considered to be not statistically significant. In other words, the accelerometers on the smartphones that were used in the experiments produce recognition results similar to the accelerometer found on a Sun SPOT.

5.2.4 Smoothing of acceleration data

The experiment is repeated by applying three common signal smoothing techniques. These techniques are moving average [35], Bartlett window [36] and Hamming window [37]. These three techniques remove the jitter noise from the obtained acceleration data. We want to investigate whether the smoothing process is needed in the pre-processing step for the activity recognition task.

The raw acceleration data from each test user was first re-sampled at 32 Hz then processed with all three smoothing techniques to produce three sets of smoothed acceleration data. The same feature extraction process, used in the previous sections, was then carried out to create the training data. Next, the 10-fold cross-validation evaluation was performed.

The average accuracies for all 24 classifiers for both raw and the smoothed acceleration data were 90.27 % and 89.90 % respectively. All four average accuracies were close (average differences between the raw and smoothed data were -0.37 %). The Wilcoxon signed-rank test was performed to check if the accuracies between the raw and each smoothed acceleration data had significant differences. The results are listed in Table 5.9 on page 90. The results of the Wilcoxon signed-rank test (at $\alpha = 0.05$) had shown that the differences between the accuracies for all three smoothed acceleration data and the accuracies for the raw acceleration data were not significant. All of the obtained p-values were higher than 0.05.

This comparison had shown that even though unnecessary jitter noise was removed and the smoothed acceleration data may be visually easier to interpret, the accuracy results of the 10-fold cross-validation evaluations had not shown significant improvement for the acceleration data smoothed using all three selected signal smoothing techniques. Since the smoothing process is an additional computational task in the pre-processing phase of the activity recognition process, it would also potentially take up additional resources for the smoothing process. A slight delay of

recognition is also anticipated, because these smoothing techniques require a certain amount of instances before a smoothed value is calculated. Since the outcome of this comparison does not show significant influence towards the recognition accuracy, it is acceptable to use only acceleration data without any filtering and noise removal for the designated activity recognition.

5.2.5 Classification using different sampling rates, window lengths and overlap percentages

One factor that can influence the classifiers' performances and accuracy is the combination of pre-processing conditions for the feature extraction, which include the selection of appropriate sampling rate, window length and the overlap percentage. To the best of our knowledge, there are no investigation that compares the recognition accuracies by using different combinations of these parameters except for [16]. The changes to the pre-processing conditions may produce better accuracies. As observed in [16], the researchers have selected sampling rate of 32 Hz with 2.5 seconds window length and an overlap percentage of 70 % and obtained accuracies from 86 % up to 98 % using the classifiers KNN and DT. According to [16], the sampling rate of 32 Hz, shorter window length and higher overlap percentage have been chosen as a good compromise between processing needs and recognition performance.

We would like to also compare different combinations of sampling rate, window length and overlap percentage in order to find the best combinations that work with the acceleration data obtained from a smartphone. The rationale behind this comparison was to investigate whether the most commonly used combinations are the best choices. In investigations such as [14] [17] and [31], the sampling rates selected were higher than 50 Hz and most used a window length of four times the selected sampling rate. These investigations also used 50 % of window overlap percentages. In our opinion, the decision on the most appropriate combination can depend on three factors. Firstly, it has to be a combination that gives the highest accuracy for a given classifier in the 10-fold cross-validation and the hold out evaluations. Secondly, for an implementation using a smartphone, for both sensor data acquisition and activity recognition, the pre-processing tasks should not be computationally intensive. In this way, context processing components running on a smartphone do not affect the normal operation of a smartphone, such as occupying the available resources or increasingly draining the battery. Thirdly, the higher each parameter is, the more data the smartphone generates. If one doubles the sampling rate, the total acceleration data acquired will also be doubled. Besides an increase in the amount of data to be processed, it may also create issues in the storage and the transfer of data, particularly if the data should be sent to a remote device. In this sub-section, the first factor was investigated. The investigations on the second and third factors will be present in the next chapter.

Table 5.9: Original and Smoothed Acceleration Data as Training Data.

Meta-level Classifier	Base-level Classifier	Accuracy (%)									
		$\bar{Raw}$	$\bar{MV}$	$\Delta\bar{MV} - \bar{Raw}$	p-value	$\bar{B}$	$\Delta\bar{B} - \bar{Raw}$	p-value	$\bar{H}$	$\Delta\bar{H} - \bar{Raw}$	p-value
-	BN	88.30	89.73	1.43	0.06	90.17	1.88	0.19	90.34	2.04	0.06
	KNN	91.66	91.23	-0.43	0.44	91.68	0.02	1.00	91.35	-0.31	0.63
	DT	91.01	91.00	-0.01	1.00	90.88	-0.14	0.81	91.04	0.03	1.00
	JRip	90.49	90.58	0.09	0.81	90.80	0.31	0.63	91.00	0.51	0.44
	NB	86.94	85.74	-1.21	0.31	83.67	-3.28	0.13	84.77	-2.17	0.19
	SMO	90.07	89.02	-1.06	0.06	88.75	-1.32	0.06	88.79	-1.29	0.06
Boosting	BN	90.07	90.02	-0.05	0.88	90.22	0.15	0.81	90.16	0.09	1.00
	KNN	92.15	91.31	-0.83	0.06	91.83	-0.32	0.63	91.75	-0.40	0.38
	DT	92.46	92.49	0.03	1.00	92.43	-0.02	1.00	91.82	-0.64	0.25
	JRip	92.23	92.31	0.08	1.00	91.99	-0.23	0.63	92.14	-0.08	1.00
	NB	86.94	85.74	-1.21	0.31	83.67	-3.28	0.13	84.77	-2.17	0.19
	SMO	91.16	90.25	-0.91	0.10	89.93	-1.23	0.19	89.98	-1.18	0.13
Bagging	BN	88.98	90.30	1.32	0.38	90.53	1.55	0.38	90.48	1.50	0.31
	KNN	92.47	92.00	-0.47	0.13	92.15	-0.32	0.19	92.15	-0.31	0.63
	DT	92.41	92.48	0.07	1.00	92.26	-0.15	0.81	92.32	-0.08	1.00
	JRip	92.02	92.17	0.15	0.31	92.12	0.11	0.63	92.13	0.11	0.63
	NB	86.95	86.22	-0.73	0.31	84.40	-2.54	0.19	85.37	-1.58	0.13
	SMO	90.16	88.97	-1.19	0.06	88.65	-1.50	0.06	88.85	-1.30	0.06
Vote	KNN + NB	89.02	89.77	0.74	0.44	89.65	0.62	0.81	89.56	0.54	0.81
	KNN + SMO	92.41	92.12	-0.29	0.13	92.21	-0.20	0.25	92.21	-0.20	0.31
	DT + KNN	91.66	91.62	-0.05	0.63	91.51	-0.16	1.00	91.78	0.12	0.88
	DT + NB	88.71	89.11	0.41	1.00	88.74	0.03	1.00	89.22	0.51	1.00
	DT + SMO	91.05	91.04	-0.01	1.00	90.93	-0.12	0.81	91.09	0.04	1.00
	SMO + NB	87.09	86.04	-1.05	0.31	84.05	-3.04	0.13	85.10	-1.99	0.19

Legend: $\bar{Raw}$ = average of the raw acceleration data, Δ = difference,
$\bar{MV}$ = average of the acceleration data filtered using moving average, $\bar{B}$ = average of the acceleration data filtered using Bartlett window,
$\bar{H}$ = average of the acceleration data filtered using Hamming window

Table 5.10: The variations of the different parameters used for sampling and preprocessing.

Parameter	Variations
Sampling rates (Hz)	8, 16, 32, 64
Window lengths (seconds)	0.5, 1, 2, 4
Overlapping percentages (%)	0, 25, 50, 75

The selected choices of parameters for this comparison is shown in Table 5.10. Though it was observed that the smartphones used in the experiments delivered acceleration values and changes below 40 Hz, the sampling rate 64 Hz was taken as a comparison reference condition. As mentioned in Chapter 4, acceleration data recorded at 64 Hz have some repeating acceleration values since the underlying system on a smartphone is only able to provide accelerometer value changes up to 40 Hz. Though we wished to find out whether 32 Hz is equally suitable for high recognition accuracy, the lower sampling rates 8 Hz and 16 Hz were also included in the evaluations to investigate the performance of lower sampling rates.

The choices of window lengths and overlap percentages might influence the potential recognition accuracy. The longer the window length is, the more information a given window can provide. The overlap of the subsequent windows may help to increase similarities for the same movement, provided that the selected features with higher overlap percentages display distinctions among different movements. Therefore, the comparison were carried out to investigate which combination of window length and overlap percentages apart from 50 % could provide better recognition accuracy.

Influence of different sampling rates

The cross-validation evaluation performed in Section 5.2.5 was repeated with four different sampling rates, a fixed window length at 4 seconds and an overlap percentage of 50 % . The obtained evaluation results are shown in Table 5.11 on page 92. It was observed that higher sampling rates produced higher recognition accuracy in the cross-validation evaluations.

Among the recommended base-level classifiers, the SMO and DT classifiers performed best for the sampling rate of 8 Hz (88.23 % and 88.00 % respectively) and the DT and JRip classifiers for 16 Hz (90.16 % and 90.14 % respectively). For the higher sampling rate at 64 Hz, only the KNN and SMO classifiers produced slightly higher average accuracies that are statistically significant. The evaluations for the meta-level classifiers displayed similar results. The meta-level classifiers produced accuracy at a sampling rate of 16 Hz that was similar to the accuracy of the base-level classifiers at the sampling rate of 32 Hz. At 8 Hz, the recommended meta-level

Table 5.11: Evaluation result of movement recognition using the recommended base- and meta-level classifiers, with sampling rates 8 Hz, 16 Hz, 32 Hz and 64 Hz, window length = 4 seconds, overlap percentage = 50 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	Accuracy using different sampling rates (%)						
		$\bar{x}_{32}$	$\bar{x}_8$	p	$\bar{x}_{16}$	p	$\bar{x}_{64}$	p
-	KNN	91.68	**87.70**	**0.00**	**90.07**	**0.00**	**92.10**	**0.00**
	DT	91.18	**88.00**	**0.00**	**90.16**	**0.00**	91.38	0.64
	JRip	90.92	**87.97**	**0.00**	**90.14**	**0.00**	91.08	0.76
	SMO	91.12	**88.23**	**0.00**	**89.91**	**0.00**	**91.56**	**0.00**
Boosting	DT	92.53	**89.54**	**0.00**	**91.57**	**0.00**	**92.93**	**0.03**
	JRip	92.46	**89.60**	**0.00**	**91.54**	**0.00**	92.80	0.19
Bagging	KNN	92.56	**89.36**	**0.00**	**91.07**	**0.00**	**93.03**	**0.00**
	DT	92.54	**89.84**	**0.00**	**91.73**	**0.00**	**92.94**	**0.01**
	JRip	92.53	**89.93**	**0.00**	**91.69**	**0.00**	92.77	0.10
Vote	KNN + SMO	92.53	**89.56**	**0.00**	**91.21**	**0.00**	**92.98**	**0.01**
	DT + KNN	91.89	**88.80**	**0.00**	**90.87**	**0.00**	**92.23**	**0.05**

Legend: $\bar{x}_f$: average accuracy, where f represents the sampling rate, p: p-value for the two-tail Wilcoxon signed rank test

classifiers had improved the respective base-level classifiers up to accuracies that were close to 90 %.

As mentioned earlier, the sampling rate 64 Hz was taken as a comparison training data set. The smartphones used in our experiment was only able to record acceleration changes up to 40 Hz. Therefore, the data sets with the sampling rate of 64 Hz might have contained up to 30 % of repetitions. This had contributed to the slight increase of accuracies (averagely 0.35 %) for all evaluations between sampling rates at 32 Hz and 64 Hz (both base- and meta-level classifiers except for DT, JRip, Bagging with DT and Vote with DT+KNN, which the accuracy improvements were not statistically significant). Based on this observation, the sampling rate of 32 Hz could be taken as a suitable and usable sampling rate for an implementation of a movement recognition application using the accelerometer of a smartphone.

The 10-fold cross-validation evaluation for the different sampling rates gave some insights for the desired implementation. The sampling rate of 32 Hz was seen as the most ideal selection among all these choices. The increase in the sampling rate to 64 Hz did not bring much improvement to the accuracy (average improvement of 0.40 % was observed), but it doubled the amount of data to be recorded and transmitted as well as for the computation of features. For a resource limited implementation, the sampling rate 8 Hz and 16 Hz can be taken into consideration, though the accuracies with 8 Hz obtained in the evaluation were averagely 3.04 % lower than the results

Table 5.12: Evaluation result of movement recognition using the recommended base- and meta-level classifiers, with window lengths 0.5 second, 1 second, 2 seconds and 4 seconds, sampling rate = 32 Hz, overlap percentage = 50 %, total test users N = 15.

Meta-level Classifier	Base-level Classifier	Accuracy using different window lengths (%)						
		4 seconds	0.5 second		1 second		2 seconds	
		$\bar{x}_{4s}$	$\bar{x}_{0.5s}$	p	$\bar{x}_{1s}$	p	$\bar{x}_{2s}$	p
-	KNN	91.68	**86.51**	**0.00**	**88.85**	**0.00**	**89.81**	**0.00**
	DT	91.18	**86.62**	**0.00**	**88.36**	**0.00**	**89.60**	**0.00**
	JRip	90.92	**87.00**	**0.00**	**88.59**	**0.00**	**89.71**	**0.00**
	SMO	91.12	**84.57**	**0.00**	**88.73**	**0.00**	**90.14**	**0.00**
Boosting	DT	92.53	**88.27**	**0.00**	**89.96**	**0.00**	**91.09**	**0.00**
	JRip	92.46	**87.34**	**0.00**	**89.23**	**0.00**	**90.74**	**0.00**
Bagging	KNN	92.56	**88.94**	**0.00**	**90.27**	**0.00**	**91.23**	**0.00**
	DT	92.54	**88.80**	**0.00**	**90.11**	**0.00**	**91.12**	**0.00**
	JRip	92.53	**88.29**	**0.00**	**89.96**	**0.00**	**91.20**	**0.00**
Vote	KNN + SMO	92.53	**88.92**	**0.00**	**90.45**	**0.00**	**91.33**	**0.00**
	DT + KNN	91.89	**87.53**	**0.00**	**89.15**	**0.00**	**90.31**	**0.00**

Legend: $\bar{x}_w$: average accuracy, where w represents the window length

obtained using 32 Hz as the sampling rate.

Influence of different window lengths

Next, we compared the influence of different window lengths towards the recognition accuracy using the same sampling rate 32 Hz and overlap percentage 50 %. The window lengths selected for comparison are 0.5 second, 1 second, 2 seconds and 4 seconds. At a sampling rate of 32 Hz, they were equivalent to 16, 32, 64 and 128 samples per window respectively. The logical expectation here was, based on the previous investigations [38] [39], that longer window lengths should produce better accuracies. The dynamic movements in our experiments (walking, go upstairs and downstairs) were movements that produced a pattern around a second. Therefore, it was expected that a window length higher than 1 second will produce meaningful results. The window length 0.5 second was selected as a control data for the case of a short window length.

The results of the evaluation using the recommended base- and meta-level classifiers are shown in Table 5.12. Among the base-level classifiers, the best accuracies for each window length were 87.00 % with JRip (0.5 second), 88.85 % with KNN (1 second), 90.14 % with SMO (2 seconds) and 91.68 % with KNN (4 seconds). Similar

Table 5.13: Evaluation result of movement recognition using the recommended base- and meta-level classifiers, with overlap percentages of 0 %, 25 %, 50 % and 75 %, and the respective p-values from the Wilcoxon signed-rank test with 50 % overlap as reference, sampling rate = 32 Hz, window length = 4 seconds, total test users N = 15.

Meta-level Classifier	Base-level Classifier	Average accuracies with different overlap percentages (%)						
		$\bar{x}_{50\%}$	$\bar{x}_{0\%}$	p	$\bar{x}_{25\%}$	p	$\bar{x}_{75\%}$	p
-	KNN	91.68	**90.34**	**0.00**	**90.75**	**0.01**	**93.53**	**0.00**
	DT	91.18	**90.29**	**0.03**	**90.45**	**0.02**	**92.43**	**0.00**
	JRip	90.92	**89.81**	**0.00**	90.22	0.11	**92.11**	**0.00**
	SMO	91.12	**88.95**	**0.00**	**90.06**	**0.00**	**92.17**	**0.00**
Boosting	DT	92.53	**91.77**	**0.00**	**91.88**	**0.03**	**94.14**	**0.00**
	JRip	92.46	**91.23**	**0.00**	91.98	0.11	**93.68**	**0.00**
Bagging	KNN	92.56	**91.22**	**0.00**	**91.78**	**0.01**	**93.72**	**0.00**
	DT	92.54	**91.64**	**0.00**	**91.81**	**0.04**	**93.73**	**0.00**
	JRip	92.53	**91.48**	**0.00**	**91.96**	**0.03**	**93.66**	**0.00**
Vote	KNN + SMO	92.53	**91.51**	**0.00**	92.08	0.25	**93.80**	**0.00**
	DT + KNN	91.89	**91.18**	**0.02**	91.28	0.12	**93.05**	**0.00**

Legend: $\bar{x}_o$: average accuracy, where o represents the overlap percentages

to previous investigations, the increase of window length also gave higher accuracies. As we compared the average accuracies for each window length, the results of 2 seconds were averagely 1.42 % lower than the accuracies using 4 seconds, while for the window length 0.5 second the results were averagely 4.47 % lower.

The highest accuracies achieved using meta-level classifiers were 92.56 % using Bagging with KNN (4 seconds). As compared to the other window length, the highest accuracies for window lengths 0.5 second, 1 second and 2 seconds were 88.94 % (Bagging with KNN), 90.45 % (Vote with KNN + SMO) and 91.33 % (Vote with KNN+SMO) respectively. As mentioned earlier, the larger window lengths produce window segments with potentially more distinguishable details for the designated classifiers. Therefore, the accuracies using window lengths 1, 2 and 4 seconds were comparatively higher than 0.5 second.

Influence of different overlap percentages

We noticed that most of the previous investigations using dedicated accelerometer and accelerometer on smartphones had mainly selected window overlap percentage of 50 % (such as [14], [17]). In this work, we also wanted to take other overlap percentages to compare the recognition accuracy. As seen in [16], 70 % of overlap

percentage gave also relatively high accuracies. The selected overlap percentages were 0 % (no overlap), 25 %, 50 % (the most popular choice) and 75 %.

As shown in Table 5.13, higher accuracies were obtained with higher overlap percentages for the recommended base-level classifiers. Among these base-level classifiers, KNN performed best and was followed by DT. Best results were obtained with the overlap percentage of 75 %, where KNN and DT gave average accuracies of 93.53 % and 92.43 % respectively. For the recommended meta-level classifiers, the accuracies also increased with the overlap percentages. The best accuracy for the meta-level classifiers were 91.77 % using Boosting with KNN (0 %), 92.08 % using Vote with KNN + SMO (25 %), 92.56 % using Bagging with KNN (50 %) and 94.14 % using Boosting with DT (75 %). As compared to the reference percentage 50 %, the overlap percentage of 75 % provided an increase in accuracy up to 1.85 % (average increase = 1.28 %). For the same number of recorded acceleration data, the 75 % overlap percentage produced higher number of training data for the generated of the respective classifiers. This may contribute to the improvement in accuracy, since higher number of training data might produce classifiers that perform better.

We observed from the evaluation results that 75 % can be an attractive overlap percentage to be applied in movement recognition. Besides the possibilities to achieve higher recognition accuracies, the use of the higher overlap percentage also shorten the time between recognitions. A 75 % only needs an additional 25 % of acceleration data to build the next window for the designated feature extraction. This also increases the number of recognitions and shorten the classification period between classified movements.

Discussion

From the above four groups of evaluations, we compared the 10-fold cross validation evaluations using different combinations of sampling rates, window lengths and overlap percentages. The same sets of base- and meta-level classifiers from the evaluation in Section 5.2.2 were selected to evaluate these combinations. In terms of sampling rate, 32 Hz was observed as an optimal value for movement recognition on smartphones with similar accelerometer found on smartphones we used in our experiments (Nokia N95 8GB, N97 and N900). This is recommended as long as there is no problem with the amount of data. In cases where data volume should be kept low, the sampling rate of 8 Hz can still provide satisfactory accuracies, which were around 3.04 % lower than the classifiers built using a sampling rate of 32 Hz.

Besides the choice of sampling rates, the suitable window length and overlap percentage were also evaluated. Similar to previous investigations, longer window lengths produced higher recognition accuracies. Therefore, the overlap percentage of 4 seconds was seen as a suitable parameter for the desired feature extraction. However, a longer window length also means a longer gap between every two subsequent recognitions. If an overlap percentage of 50 % is selected, time between two recognition is expected to be 2 seconds.

Regarding the choice of overlap percentages, we observed that an overlap percentage of 75 % was seen as a better selection over other values. The classifiers, which used 75 % overlap percentage, produced the highest overall accuracy in the evaluations. In a real-time recognition, every subsequent recognition needs only additional 25 % of new data to take place. For example, classifiers built using a window length of 4 seconds and an overlap percentage of 75 % produce a recognised movement after every second.

Both the accuracy and the output frequency of the recognition are important factors to be considered in an application, depending on its requirements and needs. For example, a medical application that keeps track of the patients' activities or movements will require high accuracy as well as frequent and consistent updates. However, if one wishes to have automatic record of his daily life as a logging tool, the requirements on update frequency and perhaps accuracy will not be as rigid as the first example. The evaluation results in this section can be used as a reference for various potential implementations. Meta-level classifiers can be seen as suitable candidates especially if enough computing and storage resources are available. In a resource-limited environment, such as a movement recognition application running on a smartphone, one may want to select base-level classifiers such as KNN or DT with lower parameters such as a sampling rate of 8 Hz with 4 seconds window length and 75 % overlap percentage.

Based on the results and observations above, the subsequent evaluations were carried out mainly using the combinations based on 32 Hz with a window length of 4 seconds as well as overlap percentages of 50 % and 75 %. These parameters gave the best accuracies in the evaluations of this sub-section.

5.2.6 Classification using different combinations of features

The evaluations in Section 5.2.2 and 5.2.5 were carried out using all five types of features computed from the acceleration data. These features were mean, variance, standard deviation as well as the energy and information entropy of the FFT. This section elaborates the repetition of the 10-fold cross-validation evaluations on the recommended classifiers with one of the selected features removed. This is followed by the next evaluation, where the recommended classifiers were generated using two different groups of features and evaluated. As recommended in the previous section, the selected parameters for the evaluations were the sampling rate 32 Hz, the window length 4 seconds and the overlap percentages 50 % and 75 %.

Evaluation of recognition accuracy with the removal of a single feature

The results and comparisons of the 10-fold cross-validation evaluation of the classifiers with a certain feature removed from the training data are listed in Table 5.14 - 5.17. In this comparison, the accuracy obtained using classifiers with all five features were used as the reference accuracy. Table 5.14 and Table 5.16 list the results of

Table 5.14: Evaluation result of movement recognition using the recommended base- and meta-level classifiers with different combinations of features, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 50 %, total test users N = 15.

Meta-level Classifier	Base-level Classifier	Average Accuracy with One Feature Removed (%)					
		None	Mean	Variance	Standard Deviation	FFT Energy	FFT Info. Entropy
-	KNN	91.68	90.67	91.88	91.77	91.71	91.09
	DT	91.18	89.85	91.16	91.17	91.22	91.22
	JRip	90.92	89.91	90.85	91.17	90.77	90.82
	SMO	91.12	88.25	90.55	90.62	90.78	90.05
Boosting	DT	92.53	91.68	92.73	92.72	92.82	92.15
	JRip	92.46	91.40	92.60	92.47	92.43	91.94
Bagging	KNN	92.56	91.53	92.62	92.61	92.61	92.11
	DT	92.54	91.62	92.56	92.56	92.55	92.51
	JRip	92.53	91.43	92.53	92.57	92.55	92.21
Vote	KNN + SMO	92.53	91.60	92.58	92.60	92.59	92.13
	DT + KNN	91.89	90.61	91.84	91.83	92.00	91.98

the 10-fold cross-validation of these classifiers, built with 50 % and 75 % overlap percentages respectively. The Wilcoxon signed-rank test was carried out to compare the accuracies of these classifiers with the reference accuracies. The average accuracy differences and the p-values of the Wilcoxon signed-rank test are summarised in Table 5.15 and Table 5.17 for overlap percentages of 50 % and 75 % respectively. The results marked **bold** are the results that were significantly different from the reference accuracies (at $p < 0.05$).

From Tables 5.14 and 5.15, the removal of one feature, from a total of five, did not produce accuracy that was statistically different with the reference accuracy except for the classifiers highlighted in **bold** in Table 5.15. The removal of the feature *mean* had caused the accuracy for all recommended classifiers to drop up to 2.87 % (Average = 1.22 %). For the evaluation using overlap percentage of 75 %, as shown in Tables 5.16 and 5.17, similar results were also obtained. However, for the overlap percentage of 75 %, the average decrease in accuracy was lower than for the case of 50 % (75 % : -1.01 % and 50 % : - 1.22 %). Similarly, the decrease in accuracy was slightly higher for the overlap percentage of 50 %.

The removal of the feature FFT information entropy gave the second highest decrease in accuracy that were statistically significant for some of the classifiers. For both overlap percentages, the KNN- and SMO-based classifiers as well as the

Table 5.15: The differences between all features and a feature removed as well as the Wilcoxon signed-rank test, 50 % ovelapping.

Meta-level Classifier	Base-level Classifier	Mean		Variance		Standard Deviation		FFT Energy		FFT Info. Entropy	
		$\Delta\bar{x}$	p	$\Delta\bar{x}$	p	$\Delta\bar{x}$	p	$\Delta\bar{x}$	p	$\Delta\bar{x}$	p
-	KNN	**-1.01**	**0.00**	**0.21**	**0.05**	0.09	0.28	0.03	0.80	**-0.59**	**0.00**
	DT	**-1.32**	**0.00**	-0.01	0.55	-0.01	0.77	0.04	0.74	0.04	0.25
	JRip	**-1.01**	**0.00**	-0.07	0.33	0.24	0.12	-0.15	0.44	-0.10	0.64
	SMO	**-2.87**	**0.00**	**-0.57**	**0.00**	**-0.49**	**0.00**	**-0.33**	**0.02**	**-1.06**	**0.00**
Boosting	DT	**-0.85**	**0.00**	0.20	0.29	0.19	0.28	0.29	0.16	**-0.39**	**0.03**
	JRip	**-1.06**	**0.00**	0.15	0.44	0.01	0.91	-0.02	0.74	**-0.52**	**0.00**
Bagging	KNN	**-1.03**	**0.00**	0.06	0.26	0.06	0.25	0.05	0.98	**-0.45**	**0.00**
	DT	**-0.92**	**0.00**	0.02	0.68	0.02	0.56	0.02	0.74	-0.03	0.78
	JRip	**-1.10**	**0.00**	-0.00	0.74	0.04	0.66	0.02	0.83	**-0.32**	**0.01**
Vote	KNN + SMO	**-0.93**	**0.00**	0.06	0.49	0.08	0.51	0.06	0.27	**-0.40**	**0.02**
	DT + KNN	**-1.28**	**0.00**	-0.04	0.32	-0.06	0.24	0.11	0.16	0.10	0.33

Legend: $\Delta\bar{x}$ = difference of accuracy averages for the two classifiers,
p = the obtained p-value from the Wilcoxon signed-rank test

recommended meta-level classifiers Boosting with DT, Boosting with JRip, Bagging with KNN and Bagging with JRip had shown a decrease in accuracy. Apart from these two features, the single removal of other features had not showed differences that were significant according to the Wilcoxon signed-rank test, or significant differences that were lower than around 0.50 %. The SMO base-level classifier had shown decrease in accuracy in every case.

From the above observations, the results suggested that, the features mean and FFT information entropy should be included in the feature combination. If one wishes to use the classifier SMO, all five features should be included. It was also observed that the decreases of accuracies, due to the removal of a given feature, were either not statistically significant. For the decreases that were statistically significant, most of them were lower than 0.5 % except for the case of the base-level classifier SMO and the removal of the feature standard deviation. In other words, the removal of a single feature from the feature combination did not greatly affect the recognition accuracies for the chosen feature extraction parameters.

Evaluation of recognition accuracy using only simple or FFT features

In the previous sub-section, it was observed that the removal of a given feature may not always affect the accuracies significantly. In this sub-section, we wanted to investigate whether it is possible to remove the FFT-based features and yet obtain equivalent recognition accuracies. The evaluation was repeated with the

Table 5.16: Evaluation result of movement recognition using meta-level classifiers with different combinations of features, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 75 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	Average Accuracy with One Feature Removed (%)					
		None	Mean	Variance	Standard Deviation	FFT Energy	FFT Info. Entropy
-	KNN	93.53	92.48	93.74	93.65	93.56	92.82
	DT	92.43	91.27	92.42	92.43	92.28	92.34
	JRip	92.11	91.28	92.22	92.24	92.02	92.07
	SMO	92.17	90.21	91.87	91.86	91.92	91.24
Boosting	DT	94.14	93.26	94.15	94.14	93.99	93.60
	JRip	93.68	92.91	93.61	93.79	93.60	93.14
Bagging	KNN	93.72	92.86	93.90	93.81	93.71	93.24
	DT	93.73	92.82	93.72	93.72	93.56	93.53
	JRip	93.66	92.78	93.54	93.60	93.53	93.16
Vote	KNN + SMO	93.80	92.94	94.00	93.85	93.77	93.35
	DT + KNN	93.05	92.08	93.07	93.05	92.93	92.99

recommended classifier built using only mean, variance and standard deviation. This combination of features is named simple features. The computation of the simple features is expected to be faster than the FFT-based features. Hence, the exclusion of FFT-based features may help to speed up the feature extraction process.

The average accuracies for both beta- and meta-level classifiers using training data with 32 Hz sampling rate, 4 seconds window length and both 50 % as well as 75 % overlap are listed in Table 5.18 on page 101 and Table 5.19 on page 102 respectively. The accuracy differences between the classifiers built using the given feature group and all five features were tested with the Wilcoxon signed-rank test (with $\alpha < 0.05$). The differences and p-values marked in **bold** indicate accuracy differences that are considered as significantly different.

For both overlap percentages, the accuracies of the classifiers built using only FFT-based features were lower than the classifiers built using all five features and simple features. For the classifiers built with the simple features and the overlap percentage of 50 %, only the base-level classifiers KNN and SMO, the meta-level classifiers Bagging with KNN and Bagging with JRip as well as the meta-level classifier Vote with KNN+SMO showed a significant decrease in accuracies (-0.75 %, 1.83 %, 0.43 %, 0.48 % and 0.51 % respectively). As for the case of the overlap percentage of 75 %, all classifiers built with only the simple or FFT features produced lower accuracies that were statistically significant except the base-level classifier

Table 5.17: The differences between all features and a feature removed as well as the Wilcoxon signed-rank test with 75 % overlap.

Meta-level Classifier	Base-level Classifier	Mean		Variance		Standard Deviation		FFT Energy		FFT Info. Entropy	
		$\Delta\bar{x}$	p	$\Delta\bar{x}$	p	$\Delta\bar{x}$	p	$\Delta\bar{x}$	p	$\Delta\bar{x}$	p
-	KNN	**-1.04**	**0.00**	**0.22**	**0.00**	**0.13**	**0.03**	0.04	0.49	**-0.70**	**0.00**
	DT	**-1.16**	**0.00**	-0.00	0.52	0.00	0.52	-0.15	0.20	-0.09	0.23
	JRip	**-0.82**	**0.00**	0.11	0.47	0.13	0.33	-0.09	0.34	-0.03	0.80
	SMO	**-1.96**	**0.00**	**-0.30**	**0.01**	**-0.31**	**0.00**	**-0.24**	**0.00**	**-0.93**	**0.00**
Boosting	DT	**-0.89**	**0.00**	0.00	0.98	-0.00	0.98	-0.15	0.14	**-0.55**	**0.00**
	JRip	**-0.77**	**0.00**	-0.07	0.45	0.11	0.21	-0.07	0.44	**-0.53**	**0.00**
Bagging	KNN	**-0.85**	**0.00**	**0.18**	**0.01**	0.10	0.11	-0.01	0.72	**-0.48**	**0.00**
	DT	**-0.91**	**0.00**	-0.01	0.66	-0.01	0.74	-0.16	0.39	-0.19	0.07
	JRip	**-0.88**	**0.00**	-0.12	0.11	-0.06	0.29	-0.13	0.12	**-0.50**	**0.00**
Vote	KNN + SMO	**-0.87**	**0.00**	**0.19**	**0.00**	0.04	0.37	-0.03	0.37	**-0.45**	**0.00**
	DT + KNN	**-0.98**	**0.00**	0.02	0.64	-0.01	0.80	-0.12	0.23	-0.06	0.39

Legend: $\Delta\bar{x}$ = difference of accuracy averages for the two classifiers,
p = the obtained p-value from the Wilcoxon signed-rank test

JRip. However, the decreased accuracies for the classifiers built using only simple features and 75 % overlap percentage were still higher than the respective classifiers built using all features and 50 % overlap percentages.

Discussion

In this sub-section, we investigated how the different combinations of features selected in a classification may influence the recognition accuracy. Based on the findings, the use of the simple features can produce comparable recognition accuracies with the use of all five features. The advantage of using only simple features for classification is the reduction of needed pre-processing tasks and resources. Furthermore, the inclusion of FFT-based features not only increase the number of features, but they are also computationally more intensive than the simple features.

Based on the results and observations, one can optimise the designated activity recognition system on a resource restricted device, such as a smartphone, by selecting only the simple features. The reduction of the number of features, from five to three for each accelerometer axis, reduces the total features from 15 to 9. The computation of the simple features is expected to be fast. Hence, the choice of selecting fewer features may result in a smaller and potentially simpler classifier (the model built from the training data), shorter pre-processing time, lesser computation of features and possible saving on battery consumption. The issues regarding these factors will

Table 5.18: Evaluation result of movement recognition using meta-level classifiers with different combinations of features, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 50 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	Selected Features						
		All	Simple only			FFT only		
		$\bar{x}_{all}$	$\bar{x}_s$	$\bar{x}_s - \bar{x}_{all}$	p	$\bar{x}_f$	$\bar{x}_f - \bar{x}_{all}$	p
-	KNN	91.68	**90.92**	**-0.75**	**0.00**	**88.24**	**-3.43**	**0.00**
	DT	91.18	91.25	0.08	0.55	**88.61**	**-2.57**	**0.00**
	JRip	90.92	90.96	0.04	0.60	**88.29**	**-2.63**	**0.00**
	SMO	91.12	**89.28**	**-1.83**	**0.00**	**84.95**	**-6.17**	**0.00**
Boosting	DT	92.53	92.26	-0.27	0.11	**89.63**	**-2.91**	**0.00**
	JRip	92.46	92.15	-0.30	0.14	**89.63**	**-2.82**	**0.00**
Bagging	KNN	92.56	**92.13**	**-0.43**	**0.01**	**90.15**	**-2.41**	**0.00**
	DT	92.54	92.35	-0.19	0.22	**89.96**	**-2.58**	**0.00**
	JRip	92.53	**92.06**	**-0.48**	**0.00**	**89.84**	**-2.69**	**0.00**
Vote	KNN + SMO	92.53	**92.01**	**-0.51**	**0.00**	**90.16**	**-2.37**	**0.00**
	DT + KNN	91.89	92.01	0.12	0.49	**89.39**	**-2.49**	**0.00**

Legend: $\bar{x}_{all}$ = average accuracies for all features, $\bar{x}_s$ = average accuracies for only simple features, $\bar{x}_f$ = average accuracies for only FFT features, p = the obtained p-value from the Wilcoxon signed-rank test

be presented and discussed in the next chapter.

5.3 Summary

The evaluations presented in this chapter demonstrated that the accelerometer on a smartphone can be used for the intended activity recognition. The comparison using a smartphone and a Sun SPOT device showed that the selected classifiers were able to provide equivalent recognition accuracy by using only the accelerometer of a smartphone as compared to a single dedicated sensor device. Classifiers such as KNN, DT and JRip were among the classifiers with higher accuracies. The meta-level classifiers built using KNN, DT, JRip had shown better recognition accuracies than the respective base-level classifiers. The SMO classifier and meta-level classifiers built with SMO had shown equally good accuracy. Hence, based on the obtained results, we propose the base- and meta-level classifiers based on KNN, DT, JRip and SMO as the recommended classifiers. The probability based classifiers, NB and BN, were comparatively less accurate than the recommended classifiers.

Table 5.19: Evaluation result of movement recognition using meta-level classifiers with different combinations of features, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 75 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	Selected Features						
		All	Simple only			FFT only		
		$\bar{x_{all}}$	$\bar{x_s}$	$\bar{x_s} - \bar{x_{all}}$	p	$\bar{x_f}$	$\bar{x}_f - \bar{x}_{all}$	p
-	KNN	93.53	**92.68**	**-0.85**	**0.00**	**90.32**	**-3.21**	**0.00**
	DT	92.43	**92.19**	**-0.23**	**0.02**	**89.66**	**-2.77**	**0.00**
	JRip	92.11	91.84	-0.27	0.22	**89.55**	**-2.55**	**0.00**
	SMO	92.17	**90.78**	**-1.39**	**0.00**	**86.18**	**-5.99**	**0.00**
Boosting	DT	94.14	**93.57**	**-0.57**	**0.00**	**91.21**	**-2.93**	**0.00**
	JRip	93.68	**93.10**	**-0.58**	**0.00**	**90.78**	**-2.90**	**0.00**
Bagging	KNN	93.72	**93.12**	**-0.59**	**0.00**	**91.23**	**-2.49**	**0.00**
	DT	93.73	**93.33**	**-0.39**	**0.02**	**91.26**	**-2.47**	**0.00**
	JRip	93.66	**93.05**	**-0.61**	**0.00**	**90.96**	**-2.70**	**0.00**
Vote	KNN + SMO	93.80	**93.15**	**-0.66**	**0.00**	**91.31**	**-2.49**	**0.00**
	DT + KNN	93.05	**92.78**	**-0.28**	**0.01**	**90.47**	**-2.58**	**0.00**

Legend: $\bar{x}_{all}$ = average accuracies for all features, $\bar{x}_s$ = average accuracies for only simple features, $\bar{x}_f$ = average accuracies for only FFT features, p = the obtained p-value from the Wilcoxon signed-rank test

The evaluations were repeated with different pre-processing parameters. The recommended classifiers built using the sampling rates of 32 Hz and 64 Hz provided highest accuracies. The accuracies obtained from the recommended classifiers using the selected lower sampling rates were considered as acceptable. For best results, the window length should be set to 4 seconds. This value was also commonly selected in related investigations. The results of the evaluations also showed that, besides the overlap percentage of 50 %, which was commonly observed in related investigations, the overlap percentage of 75 % for the sliding windows had provided among the highest accuray. The average recognition accuracy was increased up to around 1.28 % for the recommended classifiers.

The evaluations had also shown that the classifiers built with only simple features provided comparable and acceptable accuracies as the classifier built with all five features. Based on this observation, in cases where computing resources are limited, the FFT-based features can be excluded and it is still possible to obtain equivalent recognition accuracies. We may improve efficiency in terms of speed and computing resources of the recommended classifiers in the designated activity recognition system by only using computationally simpler features and a smaller number

of features, while maintaining the desired recognition accuracies.

Based on the results of the evaluations, we identified the classifiers and pre-processing parameters that can be suitable for activity recognition using the accelerometer of a smartphone. Meta-classifiers can be used together with base-level classifiers such as KNN, DT and JRip for the designated recognition. Sampling rate of acceleration data obtained from a smartphone, though lower than dedicated sensor devices as observed in previous investigations, were also applicable using the recommended classifiers. It was also possible to improve the recognition accuracies slightly by increasing the overlap percentage to 75 %. The choice of features can also be reduced to only simple features without sacrificing the recognition accuracies.

The above observations were used as the basis for the following evaluations in the next chapter. These evaluations investigate performance related issues of the classifiers such as classification using separate sets of acceleration data, pre-processing, classification and recognition speed.

References

[1] D. H. Wolpert and R. Waters, "The relationship between PAC, the statistical physics framework, the bayesian framework, and the vc framework," in *In.* Addison-Wesley, 1994, pp. 117–214.

[2] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms.* Wiley-Interscience, 2004.

[3] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition).* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[4] J. R. Quinlan, *C4.5: programs for machine learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[5] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, 2009.

[6] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, 1995, pp. 115–123.

[7] J. Rissanen, "Modeling By Shortest Data Description," *Automatica*, vol. 14, pp. 465–471, 1978.

[8] J. R. Quinlan, "MDL and categorial theories (continued)," in *ICML*, 1995, pp. 464–470.

[9] J. Fürnkranz, "Separate-and-conquer rule learning," *Artificial Intelligence Review*, vol. 13, pp. 3–54, 1999.

[10] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 2005.

[11] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[12] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods: Support Vector Learning*, pp. 185–208, 1999.

[13] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, Sep. 1997, pp. 276 –285.

[14] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive 2004*, pp. 1–17, April 2004.

[15] K. S. Kunze, P. Lukowicz, H. Junker, and G. Tröster, "Where am i: Recognizing on-body positions of wearable sensors," in *LoCA*, ser. Lecture Notes in Computer Science, T. Strang and C. Linnhoff-Popien, Eds., vol. 3479. Springer, 2005, pp. 264–275.

[16] 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 ICST 2nd international conference on Body area networks*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 1–6.

[17] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," *American Association for Artificial Intelligence*, 2005.

[18] L. Breiman, "Bagging predictors," Department of Statistics, University of California, Berkely, Tech. Rep. 421, September 1994.

[19] S. Raudys and A. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252 –264, Mar. 1991.

[20] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[21] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*. London, UK: Springer-Verlag, 1995, pp. 23–37.

[22] ——, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, August 1997.

[23] S. Dzeroski and B. Zenko, "Is combining classifiers better than selecting the best one," in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 123–130.

[24] N. Krishnan and S. Panchanathan, "Analysis of low resolution accelerometer data for continuous human activity recognition," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, 31 March 2008 - 4 April 2008 2008, pp. 3337 –3340.

[25] A. Dalton and G. O'Laighin, "Identifying activities of daily living using wireless kinematic sensors and data mining algorithms," in *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, june 2009, pp. 87 –91.

[26] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, Eds., *Machine learning, neural and statistical classification.* Upper Saddle River, NJ, USA: Ellis Horwood, 1994.

[27] T. M. Mitchell, *Machine Learning.* New York: McGraw-Hill, 1997.

[28] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1143.

[29] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[30] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, December 2006.

[31] N. Kern, B. Schiele, and A. Schmidt, "Recognizing context for annotating a live life recording," *Personal Ubiquitous Comput.*, vol. 11, no. 4, pp. 251–263, 2007.

[32] T. Brezmes, J. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living.* Salamanca, Spain: Springer-Verlag, 2009, pp. 796–799.

[33] G. Bieber, J. Voskamp, and B. Urban, "Activity recognition for everyday life on mobile phones," in *HCI (6)*, 2009, pp. 289–296.

[34] S. Microsystems, "Sun spot world," Available online at http://www.sunspotworld.com/, last checked 1st November 2010.

[35] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994.

[36] M. S. Bartlett, "Periodogram analysis and continuous spectra," *Biometrika*, vol. 37, no. 1-2, pp. 1–16, 1950.

[37] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes – The Art of Scientific Computing*. Cambridge: Cambridge University Press, 1986.

[38] S. L. Lau and K. David, "Movement recognition using the accelerometer in smartphones," in *Future Network & Mobile Summit 2010*, 2010.

[39] S. L. Lau, I. König, K. David, B. Parandian, C. Carius-Düssel, and M. Schultz, "Supporting patient monitoring using activity recognition with a smartphone," in *The 7th International Symposium on Wireless Communication Systems (ISWCS), 2010*, York, United Kingdom, September 19-22 2010, pp. 810 –814.

6 Performance of the classifiers for activity recognition using a smartphone

In this chapter, further evaluations performed based on the results in the Chapter 5 are presented. These results from the 10-fold cross validation evaluations have provided insights in the selection of suitable classification algorithms, acceleration data sampling and the pre-processing of the acceleration data. The further evaluations include the evaluation of the classifiers using separate test data, evaluation of the classifiers only using lower sampling rates as well as classification performance in terms of speed on different possible platforms. The goal of these evaluations is to investigate whether the introduced techniques and classifiers are applicable in a real implementation.

6.1 Influence of training sample sizes on the recognition accuracy

One factor that may affect the recognition accuracy of a classifier is the amount of available training data. Theoretically, higher amount of training data produces better classifier, provided the training data has a negligible amount of noise information. However, in reality it is not always possible to obtain a large sample of training data of high quality. For our designated application, it is necessary to know how much training data is desired for the generation of the classifiers. Idealistically, an unobtrusive system should not require long duration of the training phase. If this duration can be kept short, it may be more possible and acceptable to allow users to train the recognition system on their own, without professional assistance.

For this purpose, three sets of acceleration data from three test users were selected for this evaluation. From each set of the acceleration data, training data was generated with incremental sample sizes. The sizes were 150, 300, 450, 600, 900, 1200, 1500, 1800. Each training data set consisted of equal amount of samples for all five selected movements (walking, go upstairs, go downstairs, standing, sitting). The incremental sizes represented the training data collection durations of 2.5, 5.0,

7.5, 10.0, 15.0, 20.0, 25.0 and 30.0 minutes respectively. The three sets of acceleration data were selected because we wanted to have equal sample size for each movement in this evaluation. The acceleration data from other test users had total samples smaller than 1800 for at least one movement. The 10-fold cross-validation evaluation was performed on each training data set for all three selected test user.

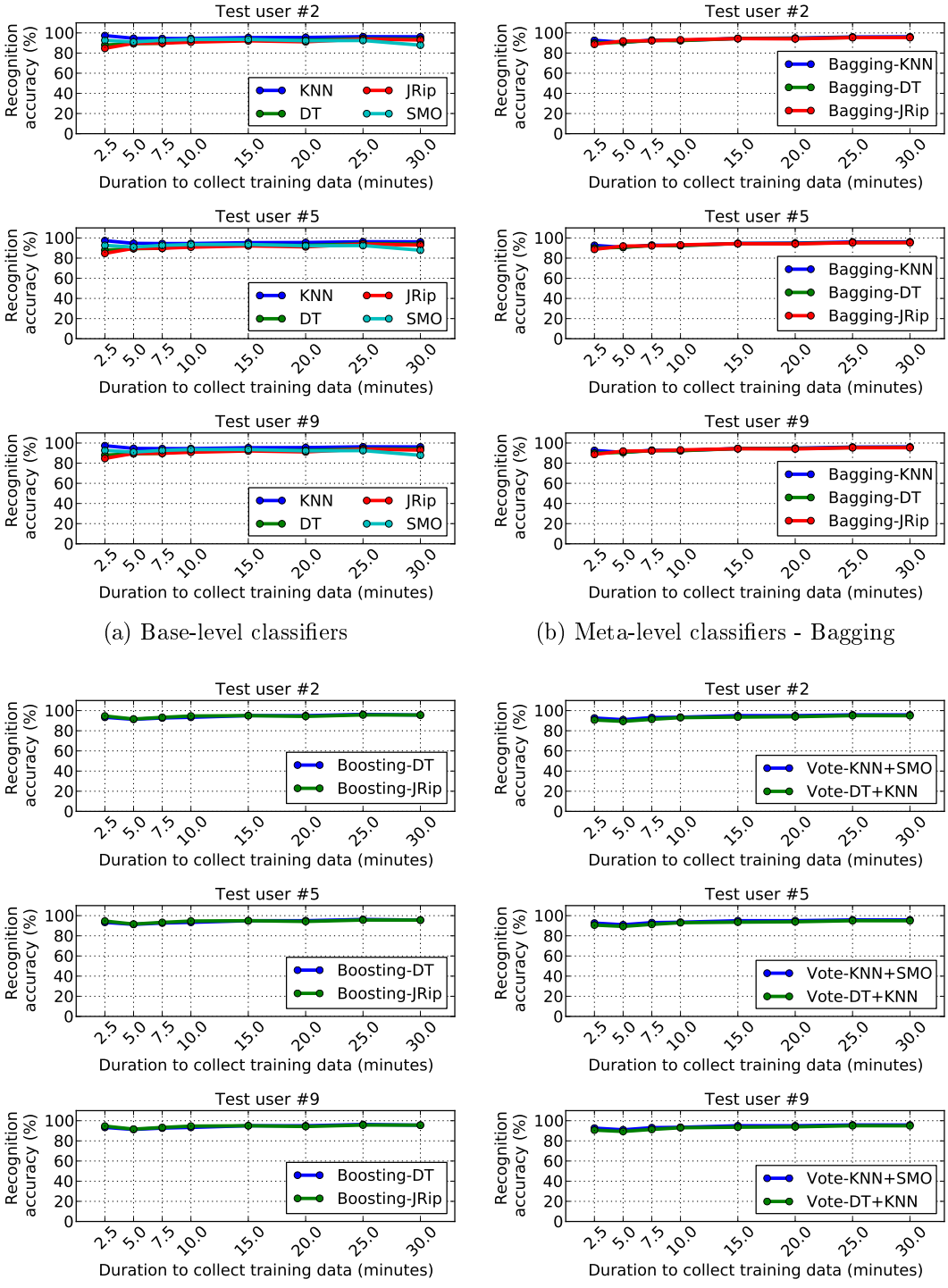
The recognition accuracy for the recommended classifiers are presented in Figure 6.1 on page 109. The results had shown that the recognition accuracy for the recommended classifiers were almost consistent particularly from 10.0 minutes to 25.0 minutes. For the base classifiers, it is observed that the first three iterations (2.5, 5.0 and 7.5 minutes) varied as the duration of training data collection increases. This outcome was not observed with the meta-level classifiers. An explanation for this is the problem of overfitting due to insufficient amount of training data. The meta-level classifiers were able to improve the overfitting problem of the base-level classifiers.

This observation led to the conclusion that meta-level classifiers are suitable candidates for good classifiers for the selected movements. Minimum recommended duration for a training data collection is 10 minutes. This translates to at least 2 minutes per movement. This requirement on the duration is more essential for dynamic movements such as walking, go up stairs and go down stairs. This is due to the varying patterns that may occur as the movements are performed. The higher number of possible patterns helps to improve the classifiers to recognise the respective movements. Static movements such as standing and sitting may not need equally long collection for the generation of a good classifier. The meta-level classifier will be less stringent on the duration requirement than the base-level classifiers.

A shorter duration for the training of the classifier is desired in a user-centric context aware system. Users can teach the activity recognition system by providing smaller number of repetitions for each movement. At 2 to 3 minutes per movement, users may not feel it as mundane as compared to a longer duration, e.g. 5 or 10 minutes per movement. Based on this result, it is also possible to conclude the acceleration data used for the previous evaluations in Chapter 5 had fulfilled the duration requirement for the designated recognition tasks.

6.2 Evaluation of the classifiers using separate training and test data

This section presents the evaluation that investigates the performance of the classifiers in classifying new acceleration data that are not part of the training data. The sets of acceleration data, used in the evaluations in Chapter 5, were reused in this evaluation. Each set of acceleration data from the respective test user was divided into two disjoint sets. The first set consisted of 66 % of the total data. It was selected as the training data. The rest of the acceleration data (34 %) was used as the



(a) Base-level classifiers

(b) Meta-level classifiers - Bagging

(c) Meta-level classifiers - Boosting

(d) Meta-level classifiers - Vote

Figure 6.1: accuracy of classifiers built using different total training data.

test data. This technique is known as the holdout method [1] [2]. The two-thirds to one-third partitioning of a data set is a common practice, though it is possible to use other proportion ratios [1]. In this evaluation, there were totally 15 pairs of training and test data.

Each pair of training and test data from the same test user was pre-processed and computed into the features in the same way. The generated classifiers, based on the selected supervised learning algorithms, were then built using the features computed from the training data. Finally, the features built using the test data were used to evaluate these classifiers to obtain the recognition accuracy representing the performance of the respective classifier for each test user. The classification using the hold out method was repeated 10 times. In each repetition, the actual acceleration data was reshuffled to generate the training and test data at the 66 %/34 % ratio. The average accuracy for each classifier was then calculated for comparison and analysis.

For this evaluation, we selected the recommended parameters and classifiers suggested in Chapter 5. The training and test data had a sampling rate of 32 Hz, a window length of 4 seconds as well as overlap percentages of 50 % and 75 %. The accuracy of the recommended base- and meta-level classifiers are listed in Table 6.1 (using 50 % overlap percentage) on page 111 and Table 6.2 (using 75 % overlap percentage) on page 112. Comparisons were made between the recognition accuracy of the 10-fold cross-validation and the classification using separate training and test data. Similar to the evaluations in the previous chapter, the results were compared using the paired Wilcoxon signed rank test to determine whether the differences between two accuracy are significant at $\alpha = 0.05$.

The recognition accuracy of the classifiers with an overlap percentage of 50 % in the hold out evaluation was slightly lower than the accuracy obtained in the 10-fold cross validation evaluations. However, among the recommended classifiers, only four classifiers produced differences in accuracy that were significant according to the paired Wilcoxon signed rank test. These classifiers were SMO and all Bagging meta-level classifiers. Though these classifiers had shown a decrease in accuracy that is statistically significant, the observed decrease was less than 0.85 %. The meta-level classifiers had produced accuracy around 92 %. For the evaluation using an overlap percentage of 75 %, all recommended classifiers in the hold out evaluation had produced a decline in accuracy that was significant according to the Wilcoxon signed-rank test at $\alpha = 0.05$. The reductions were between 0.25 % and 0.50 %, which can be considered as small reductions that are acceptable.

The evaluation results have shown that, even though the classifiers in this hold out evaluation had 66 % of the acceleration data as training data (a 10-fold cross-validation uses 90 % of the acceleration data as training), their classification accuracy were comparable to the results in the 10-fold cross-validation evaluation. The best classifiers in the evaluation were the meta-level classifiers, especially those built using 75 % overlap. This is because, from the same set of annotated acceleration

Table 6.1: Evaluation result of movement recognition using base- and meta-level classifiers with all five features, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 50 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	$\bar{x}_{10f}$	$\bar{x}_{test}$	$\bar{x}_{10f}$ - $\bar{x}_{test}$	p
-	KNN	91.68	91.41	-0.27	0.06
	DT	91.18	90.89	-0.29	0.08
	JRip	90.92	90.55	-0.37	0.08
	SMO	**91.12**	**90.27**	**-0.85**	**0.00**
Boosting	DT	92.53	92.38	-0.15	0.45
	JRip	92.46	92.29	-0.17	0.39
Bagging	KNN	**92.56**	**92.26**	**-0.30**	**0.04**
	DT	**92.54**	**92.03**	**-0.50**	**0.00**
	JRip	**92.53**	**91.98**	**-0.55**	**0.00**
Vote	KNN + SMO	92.53	92.28	-0.24	0.07
	DT + KNN	91.89	91.66	-0.23	0.06

Legend: $\bar{x}_{10f}$ = average accuracy of the 10-fold cross validation evaluation using all five features, $\bar{x}_{test}$ = average accuracy of the test data classification evaluation using all five features, p = the obtained p-value from the Wilcoxon signed-rank test

data, the classifiers with 75 % overlap were built using higher number of samples as compared to the classifiers with 50 % overlap. As an example, the acceleration data provided by test user #1 had a duration of 37.41 minutes. The total number of samples with 50 % overlap was 1121. However, the same set of acceleration data produced 2241 samples using 75 % overlap. Therefore, classifiers built from the same set of acceleration data has a higher number of samples to be used as training data, if a higher overlap percentage is selected. The use of a higher overlap percentage on the available data produces classifiers with higher accuracy, as compared to the classifiers built with lower overlap percentage using the same data.

The evaluation was repeated by using only simple features to generate the classifiers. This evaluation compared the results of recognition using the corresponding sets of separate test data with the results listed in the sub-section 5.2.6 (Table 5.18 on page 101). The reference accuracy in this comparison were the 10-fold cross validation accuracy for classifiers built using only simple features. Classifiers built using only FFT features were excluded because the accuracy obtained in the previous chapter were significantly lower. The evaluation results are listed in the Table 6.3 and 6.4 on page 114.

Similar to the results shown in Table 6.1 and 6.2, the accuracy from the hold out evaluations were slightly lower than the accuracy from the 10-fold cross validation evaluations. The reductions in accuracy that were statistically significant, as shown

Table 6.2: Evaluation result of movement recognition using base- and meta-level classifiers with all five features, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 75 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	$\bar{x}_{10f}$	$\bar{x}_{test}$	$\bar{x}_{10f}$ - $\bar{x}_{test}$	p
-	KNN	**93.53**	**93.14**	**-0.39**	**0.00**
	DT	**92.43**	**92.02**	**-0.41**	**0.02**
	JRip	**92.11**	**91.85**	**-0.25**	**0.04**
	SMO	**92.17**	**91.67**	**-0.50**	**0.00**
Boosting	DT	**94.14**	**93.66**	**-0.48**	**0.00**
	JRip	**93.68**	**93.34**	**-0.34**	**0.00**
Bagging	KNN	**93.72**	**93.40**	**-0.32**	**0.00**
	DT	**93.73**	**93.36**	**-0.37**	**0.00**
	JRip	**93.66**	**93.16**	**-0.50**	**0.00**
Vote	KNN + SMO	**93.80**	**93.45**	**-0.35**	**0.00**
	DT + KNN	**93.05**	**92.66**	**-0.39**	**0.00**

Legend: $\bar{x}_{10f}$ = average accuracy of the 10-fold cross validation evaluation using all five features, $\bar{x}_{test}$ = average accuracy of the test data classification evaluation using all five features, p = the obtained p-value from the Wilcoxon signed-rank test

in both tables, were also relative small except for the classifier SMO. The highest decrease in accuracy was -1.14 (base-level classifier SMO with overlap percentage of 50 %), whereas the rest were between -0.68 % and -0.23 % for the overlap percentages 50 % and 75 % respectively. Again, the meta-level classifiers with 75 % overlap were among the classifiers with the highest accuracy.

From the evaluations presented in this section, the best classifiers among the recommended classifiers were the classifiers built using 75 % overlap percentage. Meta-level classifiers built using this setting and only simple features produced recognition accuracy between 91.54 % and 91.99 %, although for the hold out evaluations only 66 % of the acceleration data of each test user was used as training data respectively. In other words, the meta-level classifiers are able to deliver among the highest accuracy for the evaluated acceleration data from the test users.

Discussion

In this section, the hold out method was used to test how well the recommended classifiers and selected feature extraction parameters perform. We observed that the reduction in the total of samples used for the training of classifiers did not reduce the recognition accuracy significantly (though some were statistically significant). The reduction observed in the evaluation for the base-level classifiers can be compensated by using meta-level classifiers based on these base-level classifiers. As an example,

Table 6.3: Evaluation result of movement recognition using base- and meta-level classifiers with all features and only simple features, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 50 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	$\bar{x}_{10f-simple}$	$\bar{x}_{test-simple}$	$\bar{x}_{10f-simple}$ - $\bar{x}_{test-simple}$	p
-	KNN	90.92	90.71	-0.21	0.36
	DT	**91.25**	**90.77**	**-0.48**	**0.01**
	JRip	**90.96**	**90.28**	**-0.68**	**0.00**
	SMO	**89.28**	**88.14**	**-1.14**	**0.00**
Boosting	DT	92.26	91.99	-0.27	0.06
	JRip	92.15	91.78	-0.37	0.06
Bagging	KNN	**92.13**	**91.85**	**-0.28**	**0.05**
	DT	**92.35**	**91.80**	**-0.54**	**0.00**
	JRip	**92.06**	**91.63**	**-0.43**	**0.00**
Vote	KNN + SMO	**92.01**	**91.78**	**-0.23**	**0.04**
	DT + KNN	**92.01**	**91.54**	**-0.47**	**0.02**

Legend: $\bar{x}_{10f}$ = average accuracy of the 10-fold cross validation evaluation, $\bar{x}_{test}$ = average accuracy of the test data classification evaluation, p = the obtained p-value from the Wilcoxon signed-rank test, *simple* = evaluations using only simple features

Bagging and Boosting with DT had outperformed the base-level classifier DT for both overlap percentages, making the meta-level classifiers an attractive choice for the designated recognition tasks.

When necessary, classifiers can also be built using only the simple features. The results shown in Table 6.4 were comparatively high as compared to the accuracy produced by classifiers built using all five features. In this case, the removal of FFT features may speed up the recognition process and compensate the additional time needed for the use of meta-level classifiers as compared to the respective base-level classifiers. Such performance factors regarding computing resources will be presented and discussed in Section 6.4.

The observations from this section had shown that the recommended meta-level classifiers are the better candidates for the intended recognition application. This reinforced the observation obtained in the previous section. These meta-level classifiers should provide recognition accuracy that are similar to the results in the performed evaluation in this section, provided the user provides consistent and sufficient (i.e. 2 to 3 minutes per movement) movement patterns.

Table 6.4: Evaluation result of movement recognition using base- and meta-level classifiers with all features and only simple features, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 75 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	$\bar{x}_{10f-simple}$	$\bar{x}_{test-simple}$	$\bar{x}_{10f-simple}$ - $\bar{x}_{test-simple}$	p
-	KNN	**92.68**	**92.25**	**-0.43**	**0.00**
	DT	92.19	91.91	-0.29	0.06
	JRip	91.84	91.63	-0.20	0.28
	SMO	**90.78**	**90.17**	**-0.61**	**0.00**
Boosting	DT	**93.57**	**93.16**	**-0.41**	**0.00**
	JRip	**93.10**	**92.78**	**-0.32**	**0.00**
Bagging	KNN	**93.12**	**92.79**	**-0.33**	**0.00**
	DT	**93.33**	**93.07**	**-0.27**	**0.02**
	JRip	**93.05**	**92.71**	**-0.34**	**0.00**
Vote	KNN + SMO	**93.15**	**92.77**	**-0.38**	**0.00**
	DT + KNN	**92.78**	**92.43**	**-0.35**	**0.00**

Legend: $\bar{x}_{10f-simple}$ = average accuracy of the 10-fold cross validation evaluation using the simple features, $\bar{x}_{test-simple}$ = average accuracy of the test data classification evaluation using the simple features, p = the obtained p-value from the Wilcoxon signed-rank test

6.3 Evaluation of the classifiers using training/test data with lower sampling rates

As observed in the comparison in Section 5.2.5 (on page 91) and 5.2.6 (on page 96), the recognition accuracy for the lower sampling rates in the performed evaluations were lower than those with higher sampling rates. In this section, an evaluation was carried out to compare the recognition accuracy of the recommended classifiers built using the sampling rate of 8 Hz and window length of 4 seconds. Similar to the evaluations in Section 6.2, 10-fold cross-validation and the hold out (at 66 % training data/ 34 % test data) evaluations were carried on classifiers built using 50 % and 75 % overlap percentage.

The results are listed in Table 6.5 on page 115. All features were used to generate the classifiers in this comparison. The differences between training/test evaluation and 10-fold cross-validation that are significant according to the Wilcoxon signed-rank test at $\alpha = 0.05$ are highlighted in red. Most recommended classifiers produced accuracy that is significantly lower according to the Wilcoxon signed rank test at $\alpha = 0.05$. Exceptions were the base-level classifiers DT and JRip (only 50 % overlap), the meta-level classifiers Boosting with DT (50 % overlap) and Jrip (both overlap percentages), Baggin with KNN (50 % overlap) as well as Vote with DT + KNN (50

Table 6.5: Evaluation result of movement recognition using 10-fold cross-validation and training/test evaluation, with all features, sampling rate = 8 Hz, window length = 4 seconds, overlap percentage = 50 % and 75 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	Accuracy with 50 % overlap				Accuracy with 75 % overlap			
		$\bar{x}_{10f}$	$\bar{x}_{test}$	Δx	p	$\bar{x}_{10f}$	$\bar{x}_{test}$	Δx	p
-	KNN	**87.70**	**87.31**	**-0.39**	**0.01**	**90.51**	**89.93**	**-0.57**	**0.00**
	DT	88.00	87.58	-0.42	0.06	89.56	89.35	-0.21	0.06
	JRip	87.97	87.85	-0.12	0.21	**89.45**	**89.11**	**-0.33**	**0.00**
	SMO	**88.23**	**87.46**	**-0.77**	**0.00**	**89.35**	**88.84**	**-0.51**	**0.00**
Boosting	DT	89.54	89.51	-0.03	0.85	**91.50**	**90.90**	**-0.60**	**0.00**
	JRip	89.60	89.34	-0.26	0.12	90.95	90.64	-0.31	0.07
Bagging	KNN	89.36	89.06	-0.30	0.15	**91.12**	**90.57**	**-0.54**	**0.00**
	DT	**89.84**	**89.42**	**-0.42**	**0.00**	**91.26**	**90.93**	**-0.33**	**0.01**
	JRip	**89.93**	**89.56**	**-0.37**	**0.02**	**91.10**	**90.74**	**-0.36**	**0.00**
Vote	KNN + SMO	**89.56**	**89.04**	**-0.52**	**0.00**	**91.19**	**90.67**	**-0.52**	**0.00**
	DT + KNN	88.80	88.49	-0.31	0.07	**90.29**	**89.96**	**-0.33**	**0.00**

Legend: $\bar{x}_{10f}$ = average accuracy for the 10-fold cross-validation, $\bar{x}_s$ = average accuracy for training/test evaluation, p = the obtained p-value from the Wilcoxon signed-rank test

% overlap). Consistent to the results in Section 6.2, the results from the hold out evaluation were slightly lower than the results from the 10-fold cross-validation evaluation (a decrease in accuracy from -0.33 % to -0.77 %). The meta-level classifiers built using 75 % were among the classifiers with highest accuracy ($>$ 90 %)

As compared to the recognition accuracy using acceleration data sampling at 32 Hz, the comparisons for the recommended classifiers using the hold out evaluation (at 66 % training data/34 % test data) were summarised in Table 6.6. All base- and meta-level classifiers showed significant decrease in accuracy according to the Wilcoxon signed-rank test (at $\alpha = 0.05$), if 8 Hz sampling rate is used instead of 32 Hz. The range of differences is between -2.41 % to -4.10 %, with an average of -2.88 %. The meta-level classifiers built using 75 % overlap were the only classifiers using 32 Hz sampling rate with $>$ 90 % average accuracy, except Vote with DT+KNN with an accuracy of 89.96 %.

Discussion

The results have shown that the choice of a lower sampling rate at 8 Hz decreases the recognition accuracy up to around 4.10 % as compared to classifiers built using training data with a sampling rate of 32 Hz. Comparing the classification confusion

Table 6.6: Evaluation result of movement recognition using the hold out evaluation, with all features, sampling rate = 8 Hz and 32 Hz, window length = 4 seconds, overlap percentage = 50 % and 75 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	Accuracy with 50 % overlap				Accuracy with 75 % overlap			
		$\bar{x_{32Hz}}$	$\bar{x_{8Hz}}$	Δx	p	$\bar{x_{32Hz}}$	$\bar{x_{8Hz}}$	Δx	p
-	KNN	**91.41**	**87.31**	**-4.10**	**0.000**	**93.14**	**89.93**	**-3.20**	**0.000**
	DT	**90.89**	**87.58**	**-3.31**	**0.000**	**92.02**	**89.35**	**-2.66**	**0.000**
	JRip	**90.55**	**87.85**	**-2.70**	**0.000**	**91.85**	**89.11**	**-2.74**	**0.000**
	SMO	**90.27**	**87.46**	**-2.81**	**0.000**	**91.67**	**88.84**	**-2.83**	**0.000**
Boosting	DT	**92.38**	**89.51**	**-2.87**	**0.000**	**93.66**	**90.90**	**-2.76**	**0.000**
	JRip	**92.29**	**89.34**	**-2.95**	**0.000**	**93.34**	**90.64**	**-2.70**	**0.000**
Bagging	KNN	**92.26**	**89.06**	**-3.20**	**0.000**	**93.40**	**90.57**	**-2.82**	**0.000**
	DT	**92.03**	**89.42**	**-2.61**	**0.000**	**93.36**	**90.93**	**-2.43**	**0.000**
	JRip	**91.98**	**89.56**	**-2.42**	**0.000**	**93.16**	**90.74**	**-2.41**	**0.000**
Vote	KNN + SMO	**92.28**	**89.04**	**-3.24**	**0.000**	**93.45**	**90.67**	**-2.78**	**0.000**
	DT + KNN	**91.66**	**88.49**	**-3.17**	**0.000**	**92.66**	**89.96**	**-2.70**	**0.000**

Legend: $\bar{x}_f$ = average accuracy for the hold out evaluation, f = sampling rate, p = the obtained p-value from the Wilcoxon signed-rank test

Table 6.7: Confusion matrix of the hold out evaluation for test user #8, classifier = Vote with KNN+SMO, sampling rate = 8 Hz, window length = 4 seconds.

		Normalised classification (%)									
		50 % overlap					75 % overlap				
		Classified as					Classified as				
		a	b	c	d	e	a	b	c	d	e
Ground truth	a = go upstairs	71.01	23.19	0.00	0.00	5.80	81.88	15.22	0.00	0.00	2.90
	b = walking	3.68	93.16	0.00	1.05	2.11	1.92	95.40	0.00	0.38	2.30
	c = sitting	1.15	0.00	98.85	0.00	0.00	0.00	0.00	99.59	0.41	0.00
	d = standing	1.06	2.13	0.53	96.28	0.00	0.00	2.43	0.24	97.33	0.00
	e = go downstairs	10.11	49.44	0.00	0.00	40.45	6.04	42.28	0.00	0.00	51.68

matrices between two classifiers built using same classification algorithm with different overlap percentages, the classification outcome for classifiers with 50 % overlap had more wrong classifications for the movements "go upstairs" and "go downstairs". As an example, two confusion matrices are listed in Table 6.7. The values in the confusion matrices were normalised for a side-by-side comparison.

In this example, the movements "go upstairs" and "go downstairs" were often incorrectly classified as "walking" for both classifiers. This is due to the differences between all three movements measured by the accelerometer were small. There were also For the classifier with 50 % overlap, only 40.45 % of the classified movement "go

downstairs" was correct. The same movement, classified using the classifier with 75 % overlap, obtained accuracy of 51.68 %. Similarly, the movement "go upstairs" also had an improvement of 10.87 %. This suggests that the increase of samples using a higher overlap percentage can help improve the overall accuracy of a classifier for the classification of the selected movement using acceleration data.

From the results and observations, there was a trade-off between sampling rate and recognition accuracy when we compared the classifiers built with 50 % and 75 % overlap. The reason why sampling rate should be taken into consideration is to reduce the amount of data needed to be acquired, transmitted and processed for the designated recognition tasks. The meta-level classifiers can be selected if sampling rate should be kept low but the accuracy should be acceptably high (i.e. $>$ 90 %).

6.4 Implementation performance in terms of durations and energy consumption

The evaluations performed in Chapter 5 have given us useful recommendation that can be used in the implementation of an activity recognition system. In this section we compare the performance of selected classifiers on selected implementation platforms. Our implementation is the CARMA application, which was elaborated in Section 3.3 on page 29 of Chapter 3. As described in the introduction of the CARMA application, the CA component is deployed on a smartphone. It is responsible to collect sensor data, in our case the acceleration data, and send it to the CP component for the pre-processing and recognition tasks.

The CARMA CP component performs of two main functions. Firstly, it receives the acceleration data sent from the user's smartphone, and extracts the needed features for the recognition. Secondly, a pre-selected classifier is loaded to perform the designated activity recognition using the computed features as input data. To compare the performance of the recognition process for an implementation of CARMA, we have performed the experiments using a Nokia N900 as an unobtrusive sensor device and examined the time needed for the recognition tasks. Three time variables are measured - t_{fe} is the time needed for feature extraction, t_{cl} is the time needed for the classification and t_{rec} is the total time CARMA takes to recognise an activity.

For the evaluation in this section, a sample acceleration data was selected as input for the CARMA CP component. The acceleration data was sent to the CP component at a given sampling rate to simulate the acquisition of acceleration data in a real implementation. We have selected two sampling rates, 32 Hz and 8 Hz, because the former has shown best accuracy using the recommended classifiers in the evaluations performed in Chapter 5, and the latter was expected to be more data and bandwidth efficient due to lesser measurements per second, though its recognition accuracy were also lower than the accuracy obtained using classifiers built using a sampling rate of 32 Hz.

Two durations were measured to obtain the time needed for two processes involved for a single recognition of a movement. The first duration, t_{fe}, was the time needed for the feature extraction process. This value depends on the speed of the processor, the number and the types of the selected features, as well as the number of samples in the selected window length.

The second duration, t_{cl}, was the time needed or a single classification. The t_{cl} is dependent on the types of classifier, the speed of the processor and perhaps the total number of features.

The total of both durations is equivalent to the time needed for a successful recognition. We defined this as the recognition performance in terms of duration in this thesis. Idealistically, each recognition in the CP component should not take longer than the window length times the complement of the overlap percentage (i.e. $1 - ol$, where $0 \leq ol \leq 1$). In other words, if the overlap percentage of the window is 50 %, each recognition should take 1 s and 2 s for window lengths 2 s and 4 s respectively.

The following sub-sections present the evaluations and the results from the performed experiments. We first compared the performance of all selected base- and meta-level classifiers using acceleration with a sampling rate of 32 Hz. The experiment was then repeated using a sampling rate of 8 Hz. Both experiments were performed using a desktop computer with an Intel Core i7-920 processor [1] and 8 GB Random-access Memory (RAM), and an IBM Thinkpad T20 with a Pentium III 650 MHz processor and 384 MB RAM as the two test server environments.

Next, both CA and CP components were deployed on the Nokia N900 and the recognition tasks were executed on the smartphone. In this evaluation, two investigations were carried out. Firstly, two time variables, which were the durations of feature extraction (t_{fe}) and classification (t_{cl}), were measured and compared. Secondly, we also investigated the estimated influence of the recognition tasks on the battery life of the Nokia N900. The current (in mAh) consumption was measured and compared in order to obtain estimated values for the feature extraction and classification processes. Idealistically, if we want to use a smartphone as an unobtrusive sensor and computing device, the involved processes and tasks should not drastically reduce the smartphone's operation duration. Therefore, processes and tasks that consume higher amount of energy should be avoided.

Among the recommended classifiers, the KNN-based classifiers were expected to perform slower than the other classifiers, such as DT, and JRip, because it is an instance-based classifier. The whole training data is searched through and compared during classification time to provide the intended recognition. The meta-level classifiers were expected to also be slower than the base-level classifiers, because of the additional iterations required in the classification process. On the slower devices, such as the Pentium III based Thinkpad or the Nokia N900 smartphone, the values

[1] Available online at `http://ark.intel.com/Product.aspx?id=37147`, last checked 1st August 2010

Table 6.8: The average duration for feature extraction ($\bar{t}_{fe}$) with all five features, total instances = 1000.

sampling rate (Hz)	window length (s)	Intel Core i7-920 (E_1)			Intel Pentium III (E_2)		
		$\bar{t}_{fe}$ (ms)	M_{fe} (ms)	s_{fe} (ms)	$\bar{t}_{fe}$ (ms)	M_{fe} (ms)	s_{fe} (ms)
8	4	0.11	0.10	0.12	2.04	2.38	1.34
32	4	0.38	0.34	0.30	9.40	8.50	15.74

Legend: $\bar{t}_{fe}$ = Average duration for feature extraction, s_{fe} = standard deviation

of t_{fe} and t_{cl} were expected to be higher than the faster computers.

6.4.1 Recognition speed on different environments

The two recognition environments were named as E_1 (Intel Core i7) and E_2 (Pentium III) for the following sub-section. The evaluations consisted of repetitions of recognition using the recommended qbase- and meta-level classifiers for acceleration data sampled at 32 Hz, using window length 4s and overlap percentages of 50 %. Evaluation for the overlap percentage of 75 % was not performed because the total number of instances in a given sliding window is the same for both overlap percentages. These were the recommended sampling preparation parameters concluded in Chapter 5. The average t_{fe}, t_{cl} and t_{rec} for each classifier were calculated and used in the comparison.

Table 6.8 lists the time needed for feature extraction on each environment. It was observed that the feature extraction for all five features was a fast process. Even the feature extraction process carried out on a Pentium III 650 MHz environment for a sampling rate of 32 Hz with all five features only took 9.40 ms. The reasons for the longer durations observed in the environment E_2 as compared to E_1 were the smaller amount of available memory and slower processor.

If we reduce the total number of features to only the three simple features, the results are summarised in Table 6.9 on page 120. The time needed for the computation of only the simple features was only approximately 25 % to 45 % of the corresponding time needed for all five features. In other words, the two FFT-based features tested in the evaluations in Chapter 5 increased the total computation time by almost double of what the simple features need. The median of the durations were presented in the tables as a comparison to the average of the durations, in case if the measured durations were not exactly symmetrically distributed.

The durations needed for the classification processes (t_{cl}) are illustrated in Figure 6.2 on page 121. The complete listing of classification durations are listed in Table 6.10 on page 120. For the environment E_1, most classifications performed were within 0.24 ms except for KNN, Vote with KNN + SMO and DT + KNN (around 1 ms), as well as Bagging with KNN (9.61 ms). The classification durations obtained

Table 6.9: The average duration for feature extraction ($\bar{t}_{fe}$) with only simple features, total instances = 1000.

sampling rate (Hz)	window length (s)	Intel Core i7-920 (E_1)			Intel Pentium III (E_2)		
		$\bar{t}_{fe}$ (ms)	M_{fe} (ms)	s_{fe} (ms)	$\bar{t}_{fe}$ (ms)	M_{fe} (ms)	s_{fe} (ms)
8	4	0.05	0.03	0.16	0.51	0.44	1.55
32	4	0.09	0.07	0.17	2.76	1.35	35.18

Legend: $\bar{t}_{fe}$ = Average duration for feature extraction, s_{fe} = standard deviation

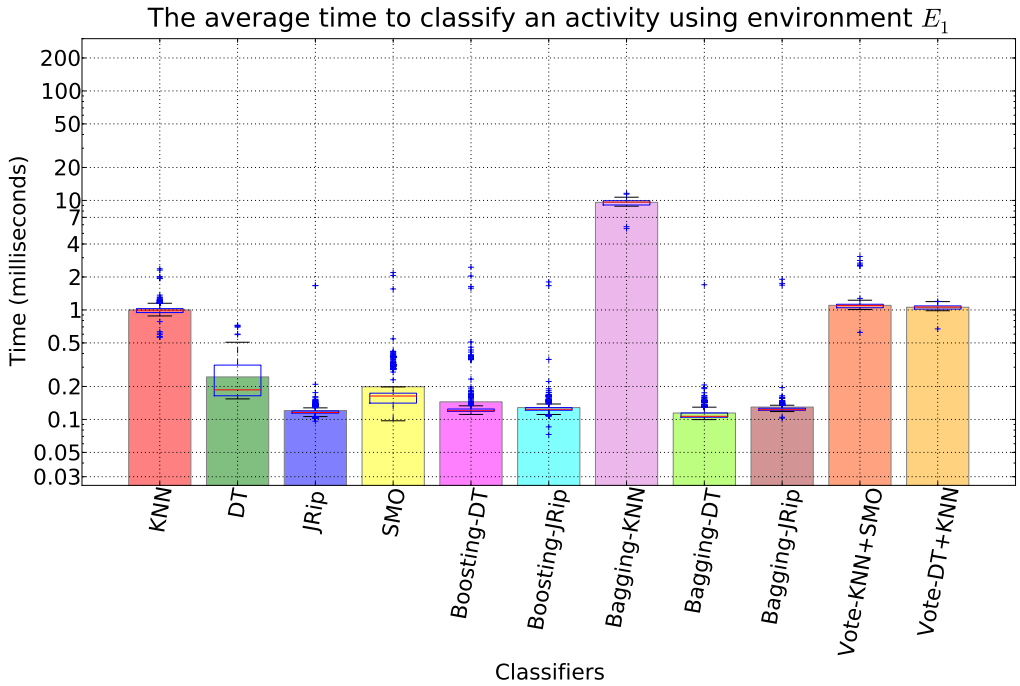
Table 6.10: The average time needed for a single classification, using all five features with total recognitions of N = 1000.

Meta-level Classifier	Base-level Classifier	Intel Core i7-920 (E_1)			Intel Pentium III (E_2)		
		$\bar{t}_{cl}$ (ms)	M_{cl} (ms)	s_{cl} (ms)	$\bar{t}_{cl}$ (ms)	M_{cl} (ms)	s_{cl} (ms)
-	KNN	1.00	1.00	0.11	23.58	28.07	11.86
	DT	0.24	0.19	0.09	1.50	1.29	1.46
	JRip	0.12	0.12	0.07	1.27	1.08	2.83
	SMO	0.20	0.16	0.12	1.68	1.52	2.21
Boosting	DT	0.14	0.12	0.13	1.57	1.38	2.88
	JRip	0.13	0.12	0.07	1.54	1.12	6.72
Bagging	KNN	9.61	9.67	0.49	224.58	225.84	109.77
	DT	0.11	0.11	0.05	1.48	1.33	1.12
	JRip	0.13	0.12	0.09	1.37	1.18	2.01
Vote	KNN + SMO	1.10	1.10	0.14	25.29	26.09	13.59
	DT + KNN	1.06	1.06	0.05	26.85	32.31	13.22

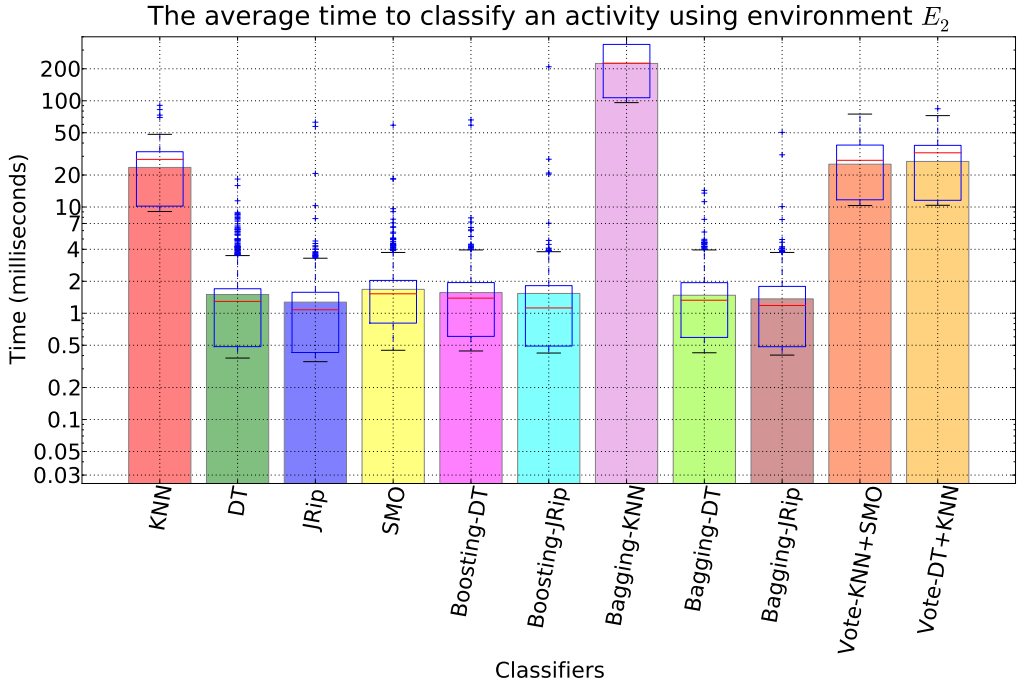
Legend: $\bar{t}_{cl}$ = average of the time for classification, M_{cl} = median of the time for a classification, s_{cl} = standard deviation for the classification time

from the environment E_2 were longer than the environment E_1. For example, the KNN based classifiers needed around 23 times longer duration to classify a movement than the durations estimated in the environment E_1. As comparison, the other classifiers needed only around 6 - 14 times longer durations.

Besides the slower processor, the environment E_2 also had only considerably limited amount of memory (only 256MB allocated for the classification in environment E_2, as compared to 2048 MB allocated for the environment E_1). However, the comparisons were made to investigate whether environments with limited computing



(a) Environment E_1

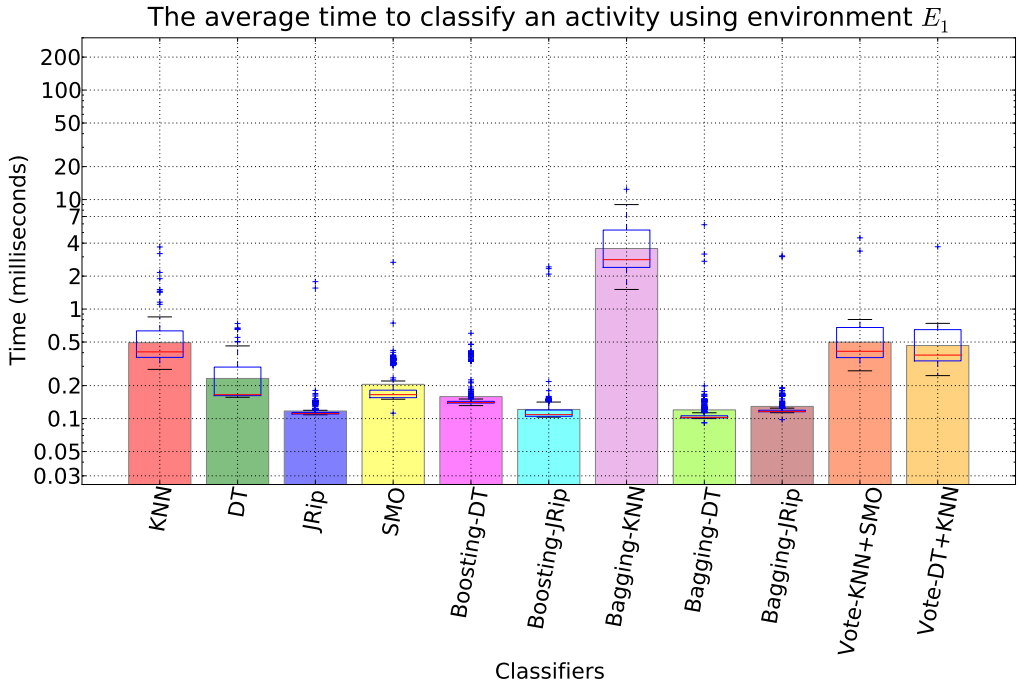


(b) Environment E_2

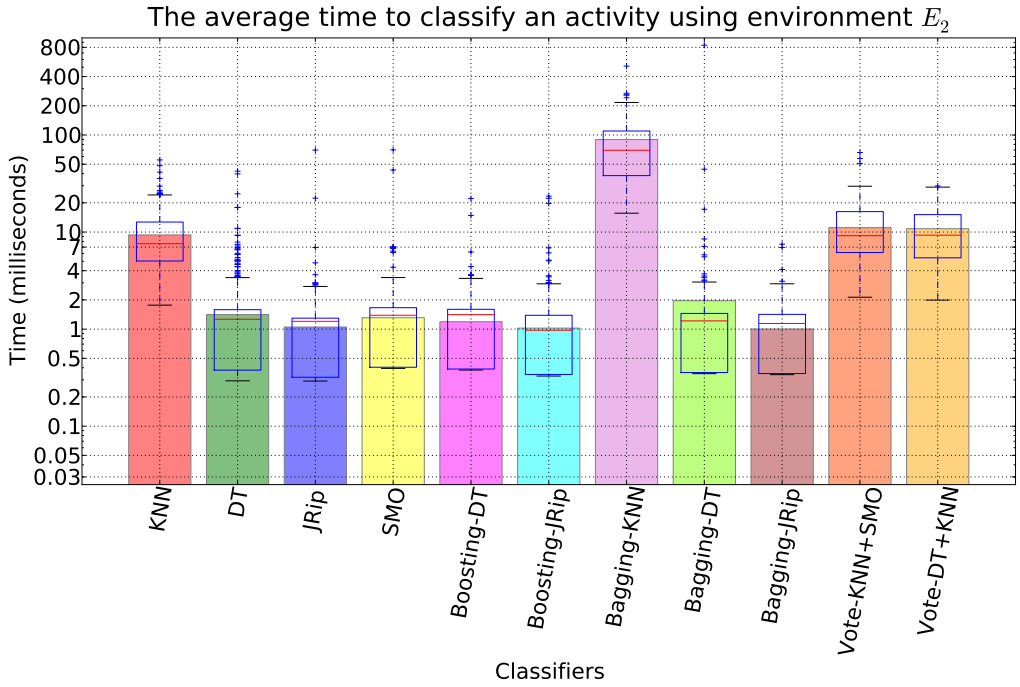
Figure 6.2: The average amount of time needed to classify an activity on environments E_1 and E_2, using all five features. (Y axis uses logarithmic scale)

resources (such as a Pentium III 650 MHz equivalent environment) are still suit-

able for the intended recognition processes. The outcomes from the environment E_2 indicated that it was possible complete a recognition within 300 ms.



(a) Environment E_1



(b) Environment E_2

Figure 6.3: The average amount of time needed to classify an activity on environments E_1 and E_2, using only simple features. (Y axis uses logarithmic scale)

Table 6.11: The average time needed for a single classification, using only simple features with total recognitions of N = 1000.

Meta-level Classifier	Base-level Classifier	Intel Core i7-920 (E_1)			Intel Pentium III (E_2)		
		$\bar{t}_{cl}$ (ms)	M_{cl} (ms)	s_{cl} (ms)	$\bar{t}_{cl}$ (ms)	M_{cl} (ms)	s_{cl} (ms)
-	KNN	0.49	0.41	0.22	9.34	7.57	6.47
	DT	0.23	0.17	0.09	1.41	1.26	2.37
	JRip	0.12	0.11	0.07	1.05	1.20	2.36
	SMO	0.21	0.17	0.11	1.31	1.39	2.69
Boosting	DT	0.16	0.14	0.06	1.19	1.40	1.05
	JRip	0.12	0.11	0.12	1.02	0.97	1.34
Bagging	KNN	3.57	2.83	1.55	89.73	69.57	60.17
	DT	0.12	0.10	0.22	1.96	1.21	26.60
	JRip	0.13	0.12	0.16	1.00	1.14	0.67
Vote	KNN + SMO	0.50	0.41	0.23	11.10	9.18	7.42
	DT + KNN	0.46	0.38	0.19	10.83	9.28	7.01

Legend: $\bar{t}_{cl}$ = average of the time for classification, M_{cl} = median of the time for a classification, s_{cl} = standard deviation for the classification time

The evaluation was repeated by using only the simple features for the classification. The evaluation results are listed in the Table 6.11. The removal of FFT-based features reduced the number of features used in the classification from 15 (five features per accelerometer axis) to 9 (three features per accelerometer axis). However, the durations for most of the classifiers using all five features and only simple features were very close (around 0.1 ms to -0.2 ms) for the environment E_1. Exceptions were the KNN-based classifiers. The KNN-based classifiers built using simple features were generally around 50 - 60 % faster than the respective classifiers built using all features.

For the environment E_2, the differences were similar to the results observed in the environment E_1. Most of the classifiers built using only simple features were only slightly faster (less than a reduction of -0.52 ms) than the classifiers built using all five features, except the KNN-based classifiers (the reductions were around -14.19 to -16.02 ms). The removal of the FFT features was able to speed up the classification time for the KNN-based classifiers. The average duration for the Bagging with DT classifier was 0.48 ms slower for only simple features. Since the standard deviation for this duration was relative high (26.60 ms), the values for the durations varied more from the average. Therefore, the median was used to compare the durations between this classifier built using all features and only simple features. The median of the latter was 0.12 ms faster than the former. Hence, the result here was regarded consistent with the results of other classifiers.

Table 6.12: The average time for feature extraction ($\bar{t}_{fe}$) on a Nokia N900 smartphone, total instances N = 500.

sampling rate (Hz)	window length (s)	All features			Only simple features		
		$\bar{t}_{fe}$ (ms)	M_{fe} (ms)	s_{fe} (ms)	$\bar{t}_{fe}$ (ms)	M_{fe} (ms)	s_{fe} (ms)
8	4	5.60	4.91	3.28	1.07	0.85	4.03
32	4	25.97	22.16	20.68	3.30	3.05	2.65

Legend: $\bar{t}_{fe}$ = Average time for feature extraction, M_{fe} = Median time for feature extraction, s_{fe} = Standard deviation of the average time for feature extraction

In this evaluation the KNN-based base- and meta-level classifiers had the longest durations. This was due to the fact that KNN-based classifiers do not build models for the desired classifications. The KNN-based classifiers compare the given test instance with all instances in the training data set to find k nearest instances (in our evaluations we used $k = 5$). It treats all instances in the training set equal. The reduction in the number of features helped reduced the total instances in a KNN-based classifier. Therefore, there were lesser instances to be compared to during a classification. This caused the reduction in the duration for a classification using a KNN-based classifier.

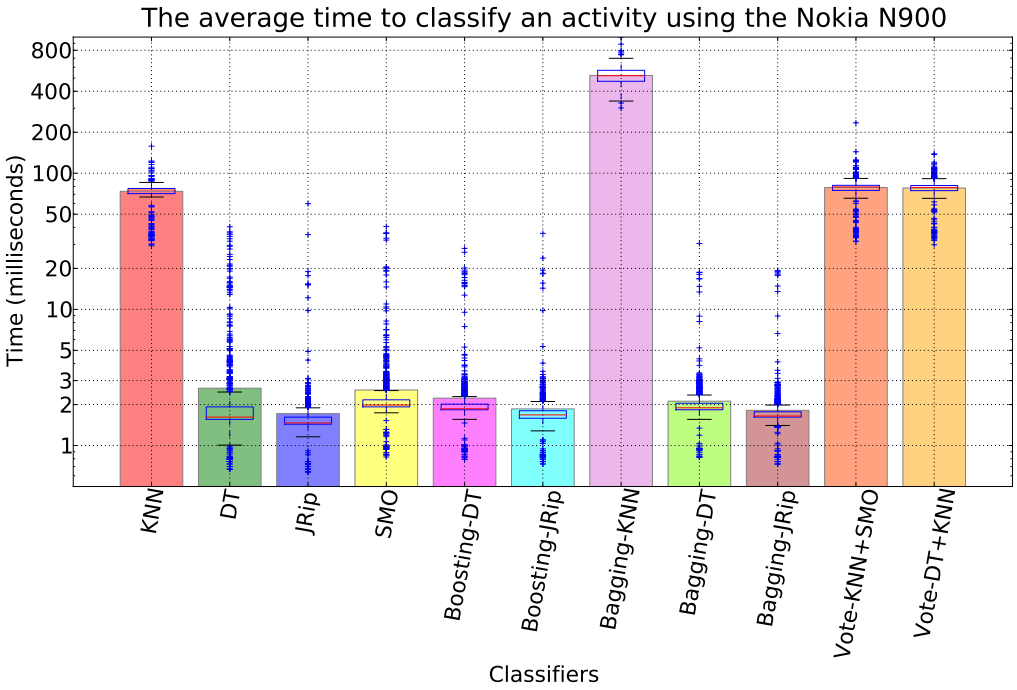
6.4.2 Local real time recognition on smartphones

The previous sub-section presented recognition performance using remote server. In this sub-section an evaluation of recognition time taking place on a smartphone, which is named as local recognition. Both context acquisition and processing components run on the smartphone. For this evaluation the Nokia N900 smartphone is selected. The Nokia N900 has a Texas Instrument OMAP 3430 ARM Cortex-A8 CPU clocked at 600 MHz and 256 MB random access memory (RAM) [2]. The embedded JAVA runtime environment (JRE) [3] is installed and used for the recognition CP component. Besides a different JRE, the recognition CP component evaluated in this sub-section on the Nokia N900 is identical to the CP component evaluated in the previous sub-section.

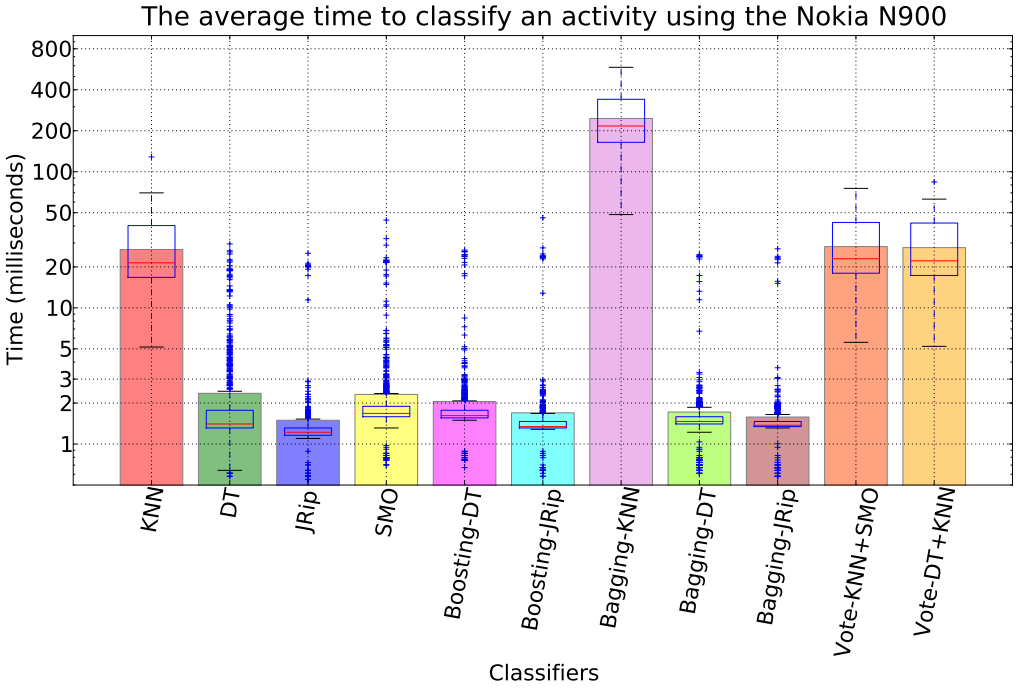
Similar to the previous sub-section, durations were measured for the feature extraction ($\bar{t}_{fe}$) and the classification ($\bar{t}_{cl}$) processes. For the purpose of comparison, two pre-precessing settings were selected - sampling rates of 8 Hz and 32 Hz, both with window length of 4 seconds and 50 % overlap. From the 1000 recognitions for each recommended classifier, the average, median and standard deviation of both

[2]Available online at `http://www.forum.nokia.com/Devices/Device_specifications/N900/`, last checked 20th October 2010

[3]Available online at `http://www.oracle.com/technetwork/java/embedded/overview/index.html`, last checked 20th October 2010



(a) Classifiers built using all five features



(b) Classifiers built using only simple features

Figure 6.4: The average amount of time needed to classify an activity on a Nokia N900 smartphone. (Y axis uses logarithmic scale)

Table 6.13: The average duration for a single classification on a Nokia N900 smartphone, using all five features with total recognition, N = 1000.

Meta-level Classifier	Base-level Classifier	All Features			Only Simple Features		
		$\bar{t}_{cl}$ (ms)	M_{cl} (ms)	s_{cl} (ms)	$\bar{t}_{cl}$ (ms)	M_{cl} (ms)	s_{cl} (ms)
-	KNN	73.66	75.26	10.35	26.86	21.42	12.59
	DT	2.64	1.62	4.12	2.36	1.40	3.24
	JRip	1.72	1.46	2.44	1.50	1.22	2.14
	SMO	2.56	1.98	3.10	2.31	1.68	3.17
Boosting	DT	2.23	1.86	2.20	2.05	1.62	2.60
	JRip	1.86	1.68	1.75	1.69	1.34	2.70
Bagging	KNN	522.61	520.07	70.57	246.15	216.74	104.02
	DT	2.12	1.89	1.61	1.72	1.46	2.07
	JRip	1.81	1.65	1.47	1.58	1.37	1.84
Vote	KNN + SMO	78.51	79.38	11.47	28.20	22.95	12.56
	DT + KNN	77.87	78.67	10.19	27.71	22.16	12.79

Legend: $\bar{t}_{cl}$ = average duration for classification, M_{cl} = Median duration for classification, s_{cl} = standard deviation for classification

durations t_{fe} and t_{cl} were calculated. The results are listed in Table 6.12 for t_{fe} and Table 6.13 for t_{cl} on page 124. The classification durations are also shown in Figure 6.4

For classifiers built using all features, the average duration of feature extraction for acceleration data at 8 Hz and 32 Hz were 5.60 ms and 25.97 ms respectively. As observed in sub-section 6.4.1, the increase in the number of instances also caused an increase in the duration for the feature extraction process. In the case of feature extraction for only simple features, the durations were 1.07 ms and 3.30 ms for 8 Hz and 32 Hz sampling rates respectively. The durations for both combinations of features were seen as acceptable, considering the processes were executed on the Nokia N900 and required only a fraction of the expected recognition duration (2 seconds for window length 4 seconds and an overlap percentage of 50 %).

The values of $\bar{t}_{cl}$ were around 1.72 to 2.64 ms for all recommended classifiers except the KNN-based classifiers. The disadvantage of instance-based classifiers was the reason behind the long classification durations. The slowest classifier was the meta-level classifier Bagging with KNN, where it needed 522.61 ms to complete a classification process. The t_{cl} for other three KNN-based classifiers were between 73.66 ms to 78.51 ms. The durations of classification using classifiers built using only simple features were shorter than the classifiers built using all five features. Similar to the results in sub-section 6.4.1, the KNN-based classifiers showed higher decrease in the durations. For the other classifiers, the differences between all features and only simple features were not more than 0.40 ms.

The faster classifiers observed are consistent with the classifiers identified in subsection 6.4.1. DT, JRip and SMO were the fastest base-level classifiers. The meta-level classifiers with DT, JRip were also equally fast. These classifiers were also faster than the respective feature extraction durations. The evaluation of the feature extraction and classification durations on a Nokia N900 had shown that all the recommended classifiers were fast except for the meta-level classifiers with KNN. Most of them are able to produce a recognition ($t_{fe} + t_{cl}$) under 30 ms. Durations for the classifiers with KNN range from 99.63 ms (base-level classifier KNN) up to 548.58 ms (Bagging with KNN).

If the total duration for feature extraction and classification were selected as a condition for the decision of suitable classifiers, DT- and JRip-based classifiers were the best candidates in this evaluation. Particularly, these classifiers had also produced some of the highest accuracy in previous evaluations. They were also among the fastest classifiers for both remote and local classifications. Therefore, the DT- and JRip-based classifiers are suitable to be used for activity recognition taking place on the smartphone. The next sub-section looks into the current consumption on the smartphone required by the recognition process.

6.4.3 Communication requirements

Based on the results in Section 6.4.1 and 6.4.2, the recognition of activities taking place at the Nokia N900 smartphone is able to produce recognition at similar speed as compared to the recognitions performed at a remote server. The rationale behind the approach to perform the desired recognition on a remote server is to reduce the load on the smartphone. However, one still needs to send the obtained acceleration data to the designated remote server for real time recognition. The transfer of acceleration data, at the recommended sampling rate, will also exhaust the battery life of the smartphone. The ideal implementation should minimise the battery consumption so that the user is not forced to keep recharging his smartphone throughout the day.

A comparison is carried out in this sub section to evaluate the battery consumption of a smartphone in different implementation settings. The idle setting is defined as the state where the smartphone does not perform any task. Network connections are also set to off. This setting provide an estimation of the minimum consumption the smartphone has under normal circumstances. It represents the common situation known as standby time. The second setting is where the acceleration data is sent to a remote server via WLAN. The third setting uses mobile broadband as communication method to transfer the acceleration data to the remote serer. The last setting keeps the acceleration data at the smartphone and performs the recognition locally.

The comparison investigate how much differences do the four settings have as compared to the consumption of the selected smartphone in the idle setting. The

Table 6.14: Comparison of the current consumptions for different settings.

Settings	Description	Estimated current consumption (mA)	Estimated battery life (hours)
Setting 1	idle state	15.42	85.6
Setting 2 - remote recognition (WLAN)	transfering at 32 Hz	227.53	5.8
	transfering at 8 Hz	216.27	6.1
Setting 3 - remote recognition (3G)	transfering at 32 Hz	287.56	4.6
	transfering at 8 Hz	280.75	4.7
Setting 4 - local recognition	base-level classifiers (at 32 Hz)	55.32	23.9
	base-level classifiers (at 8 Hz)	54.69	24.1
	meta-level classifiers (at 32 Hz)	60.44	21.8
	meta-level classifiers (at 8 Hz)	59.66	22.1

consumption measurement is carried out by a script that queries the current remaining battery charge (in mAh). The estimation of the current used within a given time is calculated using the following equation:

$$\text{Current consumption (mA)} = \frac{\text{battery charge}_{last} - \text{battery charge}_{first}}{\text{time(second)}} \times 3600 \quad (6.1)$$

The obtained consumptions for each setting are summarised in Table 6.14. For the idle setting, the estimated current consumption is between 15 mA to 17 mA. At this rate, the standard battery of the Nokia N900 (rated at 1320 mAh) should provide a battery life of roughly 77 to 88 hours (or 3.2 to 3.6 days) [4]. The real time recognition using a remote server requires continuous transmission of the acceleration data to the server. The transmissions using WLAN and 3G mobile broadband are two possibilities for the continuous transmission. The obtained estimation of current consumption using WLAN to transfer acceleration data at 32 Hz and 8 Hz are 227.53 mA (around 5.8 hours) and 216.27 mA (around 6.1 hours) respectively. The use of 3G for the transmission requires 287.56 mA (around 4.6 hours, for transfer at 32 Hz) and 280.75 mA (around 4.7 hours, for transfer at 8 Hz). The quoted maximum hours for continuous use of WLAN and 3G by Nokia [5] are up to 8 and 5.5 hours respectively. The estimated battery life periods for setting 2 and 3 are shorter than the official figures provided by the manufacturer. Most important of all, real time

[4] The estimated battery life is calculated using the equation $\textit{battery life (hours)} = \frac{1320}{\textit{estimated current consumption}}$

[5] Available online at urlhttp://europe.nokia.com/find-products/devices/nokia-n900/specifications, last checked 8th October 2010

Table 6.15: Comparison of the current consumptions for different base- and meta-level classifiers, at sampling rates of 8 Hz and 32 Hz, both with window length of 4 seconds and 50 % of overlap.

Meta-level classifier	Base-level classifier	Current consumption (mA)	
		8 Hz	32 Hz
-	KNN	57.75	57.75
	DT	57.73	57.72
	JRip	57.74	54.12
	SMO	50.50	50.51
Boosting	DT	57.64	55.16
	JRip	57.76	52.26
Boosting	KNN	73.19	72.24
	DT	61.43	64.99
	JRip	54.12	57.75
Voting	KNN + SMO	57.74	61.35
	DT + KNN	61.36	64.90

remote recognition is expected to deplete a fully charged Nokia N900 after 4.6 - 6.1 hours.

As an alternative, the setting 4 performs the recognition of activities on the smartphone locally. No transmission of acceleration data is required. This setting has an average of estimated current consumption from 55.32 mA to 60.44 mA, depending on the pre-processing conditions and classification algorithms. The equivalent estimated battery life periods are between 21.8 to 24.1 hours. As compared to the estimated battery life for remote recognition, local recognition requires a recharge of battery after almost a whole day. Even with some normal use of smartphone functions, such as telephony, SMS, personal information management tasks and low amount of Internet activities, the local recognition setting should not greatly affect the battery life as setting 2 and 3 do. It may be sufficient for the user to recharge his smartphone once a day instead of every few hours.

In the Table 6.14 on page 128 it is also shown in the setting 4 that base-level classifiers have required slightly lower amount of current than the meta-level classifiers. A detailed listing of the current consumptions for each base- and meta-level classifiers evaluated in the previous chapter is shown in Table 6.15 on page 129. The differences between all base-level classifiers are relative small. For the meta-level classifiers, only those with KNN as underlying base-level classifiers have required higher amount of current, such as Boosting with KNN, Bagging with KNN or Vote with DT+KNN. Nevertheless, as compared to remote recognition (setting 2 and 3) the local recognition using base- or meta-level classifiers are observed as more energy

efficient (around 25 % of the current consumption using 3G or 30 % of WLAN)

Based on the obtained results, remote recognition settings are seen as unsuitable because they reduce the total operational time of a phone to 4 to 6 hours. The reason for a remote recognition instead of performing it locally on the smartphone is to shift the processing load to a stationary computing device and therefore conserve battery life on the mobile device. In the case of real time recognition, the wireless transmission of acceleration data (and eventually other sensor data) is observed to consume more current than a local recognition. Hence, unless the smartphone is unable to perform the desired recognition fast enough, local recognition on a smartphone is the better option to realise real time recognition for unobtrusive activity recognition using a smartphone. The recognition performance evaluation in Section 6.4.2 has shown that the Nokia N900 smartphone is capable of performing local recognition within the expected recognition rate. Therefore, a smartphone such as the tested Nokia N900 is estimated to be able to provide almost whole day real time recognition by performing the recognition on the smartphone. In cases where activity contexts are needed remotely, the smartphone can send only the recognised activities to the requesting remote servers or services. The amount of transmission can be further minimised by sending only upon recognised activity changes. In this way, battery consumption is preserved and activity recognition is carried out without interrupting normal usages and operations of the smartphone for common users.

6.5 Classification using orientation independent acceleration data

The orientation independent transformation, presented in Chapter 4, is able to convert the raw acceleration data into acceleration values that are independent from the orientation of the accelerometer. As long as the position of the smartphone is same, the transformation produces the corresponding vertical and horizontal acceleration, regardless of how the smartphone is placed. This technique is seen as useful especially if we want to use the accelerometer of a smartphone as an unobtrusive sensor device for activity recognition. An orientation-independent approach does not require the users to place their smartphones at a fixed orientation.

Though already proposed in [3] and used in [4], to the best of our knowledge, the approach has not been compared to common accelerometer based activity recognition investigations. Most of the previous investigations were carried out with the assumption of fixed positions and orientation for the applied accelerometers. In this sub-section we investigated the recognition accuracy of the classification of orientation independent acceleration data. We also compared the obtained accuracy with the results from classification using raw acceleration data.

Similar to previous evaluations, 10-fold cross-validation evaluations had been per-

Table 6.16: Evaluation result of movement recognition using base- and meta-level classifiers built from orientation-independent acceleration data, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 50 %, total test users N = 15.

Meta-level Classifier	Base-level Classifier	Selected Features							
		All				Simple only			
		Raw	Orientation Independent			Raw	Orientation Independent		
		$\bar{x}_{all-R}$	$\bar{x}_{all-OI}$	$\bar{x}_{all-R}$ - $\bar{x}_{all-OI}$	p	$\bar{x}_{s-R}$	$\bar{x}_{s-OI}$	$\bar{x}_{s-OI}$ − $\bar{x}_{s-R}$	p
-	KNN	91.68	**86.17**	**-5.50**	**0.00**	90.92	**85.13**	**-5.79**	**0.00**
	DT	91.18	**86.91**	**-4.26**	**0.00**	91.25	**86.62**	**-4.64**	**0.00**
	JRip	90.92	**86.63**	**-4.29**	**0.00**	90.96	**86.33**	**-4.63**	**0.00**
	SMO	91.12	**82.09**	**-9.03**	**0.00**	89.28	**79.80**	**-9.48**	**0.00**
Boosting	DT	92.53	**88.19**	**-4.34**	**0.00**	92.26	**87.46**	**-4.80**	**0.00**
	JRip	92.46	**88.13**	**-4.33**	**0.00**	92.15	**87.26**	**-4.89**	**0.00**
Bagging	KNN	92.56	**88.08**	**-4.48**	**0.00**	92.13	**87.45**	**-4.68**	**0.00**
	DT	92.54	**88.46**	**-4.08**	**0.00**	92.35	**87.89**	**-4.46**	**0.00**
	JRip	92.53	**88.38**	**-4.16**	**0.00**	92.06	**87.75**	**-4.31**	**0.00**
Vote	KNN + SMO	92.53	**88.03**	**-4.50**	**0.00**	92.01	**87.58**	**-4.43**	**0.00**
	DT + KNN	91.89	**87.71**	**-4.17**	**0.00**	92.01	**87.39**	**-4.62**	**0.00**

Legend: $\bar{x}_{all}$ = average accuracy for all features, $\bar{x}_s$ = average accuracy for only simple features, $\bar{x}_f$ = average accuracy for only FFT features, p = the obtained p-value from the Wilcoxon signed-rank test

formed using training data based on orientation independent acceleration data with the same condition as sub-section 5.2.2. The values of the each axis were transformed into the vertical and horizontal components of the acceleration. Additionally, the magnitude of the recorded acceleration was also computed to be used as part of the input data. The three computed orientation-independent acceleration data were then pre-processed to compute the features needed for the designated classifiers, identical to the steps applied for the raw acceleration data.

The results of the 10-fold cross-validation evaluations for both base- and meta-level classifiers are summarised in Table 6.17. For the comparison, the evaluation with classification using only the simple features was repeated, as performed in the evaluations in sub section 5.2.6. From the obtained results, it was observed that the orientation independent transformed training data produced lower average recognition accuracy as compared to the classification accuracy based on raw acceleration data. The p-values of the Wilcoxon signed-rank test for all classifiers were less than to 0.05, indicating that the differences between orientation independent and the raw acceleration data were significant. The comparison for these two types of acceleration data using only simple features had shown similar results.

Table 6.17: Evaluation result of movement recognition using meta-level classifiers built from orientation-independent acceleration data, sampling rate = 32 Hz, window length = 4 seconds, overlap percentage = 75 %, total test users $N = 15$.

Meta-level Classifier	Base-level Classifier	Selected Features							
		All				Simple only			
		Raw	Orientation Independent			Raw	Orientation Independent		
		$\bar{x}_{all-R}$	$\bar{x}_{all-OI}$	$\bar{x}_{all-R}$ - $\bar{x}_{all-OI}$	p	$\bar{x}_{s-R}$	$\bar{x}_{s-OI}$	$\bar{x}_{s-OI} - \bar{x}_{s-R}$	p
-	KNN	93.53	**88.79**	**-4.74**	**0.00**	92.68	**87.85**	**-4.83**	**0.00**
	DT	92.43	**87.96**	**-4.46**	**0.00**	92.19	**87.94**	**-4.26**	**0.00**
	JRip	92.11	**88.01**	**-4.10**	**0.00**	91.84	**87.60**	**-4.24**	**0.00**
	SMO	92.17	**84.13**	**-8.04**	**0.00**	90.78	**82.93**	**-7.85**	**0.00**
Boosting	DT	94.14	**90.17**	**-3.98**	**0.00**	93.57	**89.16**	**-4.42**	**0.00**
	JRip	93.68	**89.44**	**-4.24**	**0.00**	93.10	**88.81**	**-4.29**	**0.00**
Bagging	KNN	93.72	**89.58**	**-4.14**	**0.00**	93.12	**89.10**	**-4.02**	**0.00**
	DT	93.73	**90.06**	**-3.67**	**0.00**	93.33	**89.46**	**-3.87**	**0.00**
	JRip	93.66	**89.62**	**-4.04**	**0.00**	93.05	**89.01**	**-4.04**	**0.00**
Vote	KNN + SMO	93.80	**89.60**	**-4.20**	**0.00**	93.15	**89.16**	**-3.98**	**0.00**
	DT + KNN	93.05	**88.86**	**-4.20**	**0.00**	92.78	**88.74**	**-4.04**	**0.00**

Legend: $\bar{x}_{all}$ = average accuracy for all features, $\bar{x}_s$ = average accuracy for only simple features, p = the p-value from the Wilcoxon signed-rank test

As compared to the accuracy, presented by Yang [4], the accuracy obtained in the evaluation of this section was lower. However, Yang had not explicitly investigated movements that are similar to walking such as going upstairs and downstairs. Under such circumstances, if all three movements were to be considered as walking in general, the recognition accuracy should be considerably higher. In this thesis, since the three movements were separately classified, the similarity between these movements was the reason why accuracy was comparatively lower than the results from Yang. Therefore,

From this comparison, the transformation of acceleration data into orientation independent data does not provide equivalent or better recognition accuracy. based on the results, a decrease of 5 % in accuracy is expected if one wishes to use orientation independent acceleration data for activity recognition using classifiers such as KNN, DT and JRip as well as the corresponding meta-level classifiers.

6.6 Summary

Chapter 6 presents further evaluations based on the results from Chapter 5 to investigate issues that help us to understand how classification of activities performs in a

real implementation. The evaluation using separate training and test data examined the accuracy of the classifiers in classifying newer data. Most of the classifiers did not have differences that were statistically significant. The evaluation of classifiers built using a sampling rate of 8 Hz had shown that lower sampling rate produced classifiers with loss of accuracy up to 4.10 %.

The performance evaluations compared issues such as speed and battery consumption. For classification on remote servers, the base-level classifiers such as DT and JRip and the meta-level classifiers based on them were the fastest among the recommended classifiers. The KNN-based classifiers were the slowest, since the entire training data set was searched through during classification. As compared to the classification time, the computation of features can be a lot slower for higher sampling rates and longer window lengths if all five features are selected. The feature extraction durations for simple features were considerably shorter.

As for classification on the Nokia N900, the evaluation results are comparable to the results given by the environment of Intel Pentium III. However for more computational intensive operations, the Nokia N900 is slower due the lesser memory and slightly slower processor speed. The comparison of the battery consumption evaluation had demonstrated that the transmission of acceleration data at real time consumes far more current than local classification. Since a smartphone such as the Nokia N900 is capable of performing the recognition using classification without delay, remote real time recognition is seen as not suitable since it drains the battery of the smartphone within 5-6 hours. The battery life of the selected smartphone is estimated to last up to almost 24 hours if the recognition is performed on the smartphone locally.

Last but not least, we had also compared the classification accuracy using raw and orientation-independent acceleration data. The results had shown that the latter provides lower accuracy. The transformation of the acceleration data into orientation-independent data did not produce equivalent or better accuracy.

References

[1] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[2] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 2005.

[3] D. W. Mizell, "Using gravity to estimate accelerometer orientation," in *7th International Symposium on Wearable Computers (ISWC 2003), 21-23 October 2003, White Plains, NY, USA*. IEEE Computer Society, 2003, pp. 252–253.

[4] J. Yang, "Toward physical activity diary: motion recognition using simple acceleration features with mobile phones," in *IMCE '09: Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*. New York, NY, USA: ACM, 2009, pp. 1–10.

7 Conclusion and outlook

In the preceding chapters, we have presented the proposal and investigations related to a user-centric context aware system using unobtrusive devices and systems. The three main contributions of this thesis are identification of the need to apply unobtrusive devices and applications for context awareness, the empirical evaluations for the use of classification algorithms for activity recognition, as well as the design and development of prototype application for the proposed user-centric context aware system.

The following sections recapitulate of the contribution of the thesis. Section 7.1 provide an overview of the proposed user-centric context aware system and the identified challenges. The sub-sections 7.1.1 through 7.1.3 summarise the contributions. This is followed by an outlook in the area covered within the scope of thesis in Section 7.2.

7.1 Contributions

The advances in the area of context awareness have provided promising results and achievements since the 90s. As many of these results and solutions will evolve into potential technologies and products in the near future, it is a challenge for us to maintain a balance between user control and automation in a context aware system. Based on this challenge, we have proposed the idea of a user-centric context aware system that focuses on using unobtrusive devices and services for the designated context aware features.

Many of the proposed solutions so far have suggested the use of sensor devices that are regarded as obtrusive and non-practical for users to adopt this solution in their daily lives. This may lead to inconvenience or even rejection of these solutions. As an example, in the selected area of activity recognition using acceleration data, many of the investigations proposed the used of multiple sensors placed at different fixed positions of the body. Furthermore, the classification algorithms studied in these investigations are only compared in terms of accuracies. The influence of different steps involved in the classification of activity contexts is yet to be investigated. The applicability of these steps also needs to be investigated in order to find out whether they can be applied in a recognition application running on a commercially available smartphone.

This thesis presents discussions and solutions to the above issues as contributions. They are summarised as follows:

Proposal to apply unobtrusive devices and services for context awareness

- Reviewed and refined the definitions and concepts related to context awareness.
- Discussion on the need to use unobtrusive devices and services for context awareness.
- Discussion on the idea of a user-centric context aware system that utilises unobtrusive devices and services.
- Discussion on the use of a smartphone as a suitable unobtrusive device for activity recognition.
- Design and development of a flexible architecture for the proposed user-centric context aware system.

Empirical evaluations for the use of classification algorithms for activity recognition

- Comparison of the selected base- and meta-level classifiers in activity recognition using accelerometer of a smartphone.
- Evaluation of the performance issues of the selected classifiers in different environments.
- Comparison of the influences of the pre-processing conditions have on the classification accuracy.
- Comparison of the influences of the features have on the classification accuracy.
- Discussion on how one should select the most appropriate pre-processing and feature extraction for the intended activity recognition using unobtrusive device in the user-centric context aware system.

Design and development of prototype applications for the proposed user–centric context aware system

- Design and development of the Context Aware Remote Monitoring Assistant (CARMA) application for activity recognition using a smartphone.

The contributions are detailed in the following sub-sections.

7.1.1 Proposal to apply unobtrusive devices and services for context awareness

In the course of this thesis, we have reviewed and utilised results from the related work to refine fundamental definitions and concepts related to context awareness. Based on this study, we have proposed the use of unobtrusive devices and services in context awareness to ensure users' acceptance and needs can be fulfilled. In Chapter 2, the proposal was coined as the user-centric context aware system.

Chapter 3 introduced the use of unobtrusive sensor devices for activity recognition. In this chapter, a review on related work was presented. The review gave an overview of the various sensors, applications and techniques applied in the past. However, not many investigations have focused on methods that are unobtrusive. This led us to the proposal of using a smartphone as an unobtrusive device for activity recognition. We presented reasons why smartphones are seen as suitable device and proposed an architecture for user-centric context aware system.

7.1.2 Empirical evaluations for the use of classification algorithms for activity recognition

Based on the proposed ideas, Chapter 4 elaborated the steps involved in utilizing the accelerometer of a smartphone for activity recognition using the proposed architecture. These steps are data collection, data pre-processing and feature extraction. In Chapter 5, the selected base- and meta-level classifiers are presented and evaluated using acceleration data collected from 15 test users.

From the evaluations, it is shown that the accelerometer of a smartphone is suitable for the recognition of selected activities. The meta-level classifiers have been compared to the base-level classifiers in the evaluations. It was shown that, for the purpose of activity recognition using acceleration data from smartphones, the meta-level classifiers provided better accuracies then the respective base-level classifiers. Additionally, comparisons of the influences of pre-processing conditions and choice of features have been carried out to investigate suitable ways to perform activity recognition in the proposed architecture and system. It was shown that lower sampling rate such as 8 Hz to 32 Hz (as compared to $>$ 50 Hz in most of the related work) are still able to deliver accuracies more than 90 % using the recommended classifiers.

The impacts of pre-processing and feature extraction have on the classification accuracy for activity recognition using acceleration data have not been discussed in the related work. In this thesis, we have performed extensive comparisons on the two steps in order to understand and select the best conditions for designated recognition tasks. Different window lengths and overlap percentages for the sliding window techniques have been compared. The respective recommended values have been also identified. Generally a longer window length and the overlap percentages

50 % and 75 % provides higher recognition accuracies than other combinations.

Another issue we have tackled in this thesis is the performance of the recommended recognition steps and classifiers in a real implementation. Many of the previous work investigated only accuracy and classification related issues. There are yet no investigation on how these identified techniques and classification are applicable and practical in a whole day real implementation. The evaluations and comparisons have shown that a real-time recognition is more suitable to perform the classification on the smartphone locally. Remote recognition on a dedicated server is not suitable because the transmission of acceleration data over a wireless network reduces the battery life of the smartphone to an operation duration of 4-6 hours. This may create inconvenience for the users and may affect the acceptance of such context aware techniques in their daily lives. Using the techniques proposed in this thesis, the local recognition performed on a Nokia N900 smartphone has an estimated operation duration of almost 24 hours. The smartphone is regarded as a suitable and an attractive unobtrusive device for the proposed context aware functions.

7.1.3 Design and development of prototype applications and tools for the proposed user-centric context aware system

The obtained observations and evaluation results are used in the design and development of the prototype CARMA application. This application is built using the proposed architecture and is applied in the MATRIX research project. Both local and remote recognition can be selected in the application, depending on the requirement of the desired implementation.

7.2 Outlook

The theoretical and experimental results presented in this thesis have established a basis knowledge in the proposal of a user-centric context aware system. There remain, however, still some interesting scientific questions. This section briefly outlines these questions we consider as most relevant.

In the area of recognition technique for activity recognition using unobtrusive devices and services, we have proposed the use of meta-level classifiers that are comparatively fast and accurate to be implemented on a smartphone. In the future, the mobile devices are expected to be faster, more energy efficient and have additional sensors. It is necessary to extend the selections of classifiers and algorithms to investigate alternatives that can improve the recognition accuracy while maintaining performance and efficiency of the system. Similarly, the accelerometer chosen in this thesis is only one of the many sensors available on the smartphone. It will be

interesting to use and combine additional sensors to perform the desired activity recognition.

Another potential investigation is to identify and study further suitable unobtrusive approaches for context awareness. Additional devices and systems can be evaluated and utilised. For example, watch like multipurpose devices can be considered since watches are commonly worn by many people and are also another "always around" devices. By applying recommendations based on the results in this thesis and potential future development, the vision of a user-centric context aware system using unobtrusive devices and services can be further refined and improved.

In this thesis, the concept of activity covers a wide scope of contexts. The performed investigation only considered a relative small selection of activities. It is possible to include more activities that are relevant to the selected application domain and available sensors to classify additional activities. The recognition of specific activities based on basic activities requires appropriate modelling techniques to infer the former from the latter. With the inclusion of more recognisable activities in the designated system, the potential application domains may increase.

Context awareness is a research area that aims to utilise contexts to ease a user's tasks and hence fulfil his needs. Newer and more innovative approaches in different aspects, such as communications, human-computer interactions, applications and systems have been investigated and proposed to enable the acquisition of implicit information from available sensor data. This information, commonly defined as context, supports context aware systems to act and react based on the available contexts and their changes.

In getting user acceptance for the proposed approaches and ideas based on context awareness, it is important to ensure that the target realisations, both software and hardware, are user-centric. The hardware devices should be seamlessly integrated in the users' environment, without requiring them to make noticeable adjustments or even compromise their daily lives. From the software point of view, users should be given means for adequate understanding and overview of a context aware system as well as sufficient control for potential creation, modification and management of the desired services and functions. We name the envisioned system a user-centric context aware system.

With this motivation, this book presents the proposal to use a smartphone with its built-in accelerometer as an example solution for the envisioned system. The possibilities of using a smartphone to enable recognition of basic activities are discussed, such as walking, going up the staircase, sitting etc. The proposal is tested experimentally via evaluations on real data obtained from 15 test users. The evaluations investigate factors and techniques that are deemed applicable and suitable for the vision of an unobtrusive user-centric context aware system.

ISBN 978-3-86219-244-1