Wrist-worn Activity Recognition Based on Smart Watches

Peter Halbmayer

Master Thesis
to achieve the university degree of
Diplom-Ingenieur

submitted to
Johannes Kepler University Linz

Supervisors
Univ.-Prof. Dr. Alois Ferscha

Linz, Austria, June 2015.

Abstract

The recognition of daily activities of human behavior by the application of inertial sensor technology got increased attention after customized sensor platforms became widely available. Since this time, various activity recognition application scenarios have been researched and revised with variable success. Most existing work has in common that it is always applied within a very restricted domain, mainly within laboratory setups. By this restriction usually the quality of recognition results increase within their domain of application but limits system accuracy within practical scenarios due to the unsupervised and unguided applicability in everyday living environments.

The utilization of mobile and embedded devices and appliances opened the potential of performing human activity recognition tasks within different real life scenarios. Emerging systems heavily use inertial sensor technology to perform recognition of specialized activities for example within healthcare, fitness or work life scenarios. Again, they work considerably well within the areas they are explored and generally fail outside their application domains.

In this work a system is presented that specifies and implements an activity recognition system that supports the practical application within daily living situations. By specifying a generalized model approach it is possible to establish (semi-)automatical and personalized activity of daily living recognition routines. They can be utilized for real world appliance control or background personal information management (e.g., do not deliver notifications when the user is in certain situations)

Kurzfassung

Die Erkennung täglicher Aktivitäten des menschlichen Verhaltens durch die Anwendung von Inertialsensortechnologie erhielt durch die Massenverbreitung entsprechender Sensorplattformen erhöhte Aufmerksamkeit. Seit dieser Zeit wurden verschiedene Aktivitätserkennungsszenarien mit gemischten Erfolg überprüft und untersucht. Verwandte Arbeiten haben gemeinsam, dass sie stets in einer eingeschränkten Domäne unter Laborbedingungen durchgeführt werden. Diese Einschränkung erhöht üblicherweise die Qualität der Erkennungsrate innerhalb der Anwendungsdomäne, limitiert jedoch das System unter praktischen Bedingungen, bei denen die Akkurazität existierender Ansätze rapide abnimmt oder undurchführbar wird.

Die Nutzung mobiler und eingebetter Geräte und Apparate eröffnete das Potenzial zur Erkennung menschlicher Aktivitäten innerhalb verschiedener realistischer Szenarien. Entstehende Systeme machen umfangreichen Gebrauch von Inertialsensortechnologie um die Erkennung spezieller Aktivitiäten, beispielsweise im Gesundheitswesen, Fitnessbereich oder in Szenarien des Arbeitslebens, durchzuführen. Wiederum sind solche Systeme innerhalb ihres untersuchten Bereichs überdurchschnittlich erfolgreich und versagen üblicherweise außerhalb ihrer Anwendungsdomäne.

In dieser Arbeit wird ein System präsentiert welches Limitierungen existierender Lösungen umgeht, durch die Einführung eines Systems, das die praktische Anwendung in täglichen Lebenssituationen erlaubt. Durch die Spezifikation eines generalisierten Modellansatzes ist es möglich (semi-)automatisierte und personalisierte Erkennungsroutinen für Aktivitäten des täglichen Lebens zu etablieren. Diese können für die Steuerung von Apparaten oder persönliches Informationsmanagement im Hintergrund verwendet werden. (Beispielsweise sollen keine Nachrichten angezeigt werden, wenn sich ein Benutzer in einer bestimmten Situation befindet.)

Contents

1 Introduction **3**
- 1.1 Motivation . . . 3
- 1.2 Activities of Daily Living (ADL) . . . 4
  - 1.2.1 Time Use Survey (TUS) . . . 5
  - 1.2.2 Activity Terminology . . . 7
- 1.3 The ARC Design Pattern . . . 9
  - 1.3.1 Traditional ARC . . . 10
  - 1.3.2 ARC with Extension . . . 10
- 1.4 Research Objectives . . . 11

2 Related Work **12**
- 2.1 Context Computing and Activity Recognition . . . 12
- 2.2 Approaches to Computational Activity Recognition . . . 13
- 2.3 Wrist Worn Based Activity Recognition . . . 14
- 2.4 Time Use Based Activity Recognition . . . 15
- 2.5 High Level Activity Modelling . . . 15
- 2.6 General Data Acquisition and Pattern Classification Tools . . . 16

3 System Architecture **18**
- 3.1 Core Runtime Components . . . 18
  - 3.1.1 System Environment . . . 20
  - 3.1.2 System Configuration and Configuration Items . . . 20
  - 3.1.3 Producer . . . 24
  - 3.1.4 Consumer . . . 24
  - 3.1.5 Message Types and Processing . . . 25
  - 3.1.6 Data Storage Organization . . . 28
- 3.2 Activity Recognition Model and Processing . . . 30
  - 3.2.1 Chain Tasks and Processes . . . 31
  - 3.2.2 Framework ARC Design . . . 35
  - 3.2.3 Activity Model Organization . . . 36
  - 3.2.4 Classification Model Organization . . . 38
- 3.3 Practical Considerations . . . 39
  - 3.3.1 Choice of Runtime Platform . . . 39
  - 3.3.2 Long Term Data Sampling Considerations . . . 41

4 Application Interfaces **42**
4.1 Mobile Application Interfaces . . . 42
4.1.1 Icon Based Activity Labeling . . . 43
4.1.2 Data Labeling Interaction . . . 44
4.1.3 Development Views . . . 45
4.1.4 Application Views . . . 47
4.2 ARC Tools Implementation Overview . . . 48
4.2.1 Desktop based data recording and feature extraction . . . 50
4.2.2 Activity Trace Visualization - LabelVisu . . . 50
4.2.3 Classification Application . . . 52

5 Experimental Evaluation **55**
5.1 User Experiment . . . 55
5.1.1 Data Samples Overview . . . 56
5.1.2 Summary Statistics . . . 57
5.1.3 User Data Comparison . . . 59
5.2 Visual Evaluation . . . 61
5.2.1 Box Plot . . . 61
5.2.2 Scatter Plot . . . 62
5.3 Classifier Evaluation . . . 64
5.3.1 Confusion Matrix and Heatmap Representation . . . 64
5.3.2 Initial Evaluation Results . . . 67
5.4 Optimization Strategies . . . 71
5.4.1 Signal Based Filtering . . . 72
5.4.2 Activity Based Filtering . . . 72
5.4.3 Evaluation Results for Filtered Data . . . 74

6 Outlook and Conclusions **76**
6.1 Future Work . . . 76
6.1.1 Scientific Considerations . . . 76
6.1.2 Practical Considerations . . . 77
6.2 Contributions to the Research Objectives . . . 77

Bibliography **80**

A Appendix **85**
A.1 Armband Comparison . . . 85
A.2 Smart Watch Comparison . . . 85
A.3 Data Samples Details . . . 86
A.3.1 Participant 1 . . . 87
A.3.2 Participant 2 . . . 90
A.3.3 Participant 3 . . . 91
Curriculum Vitae . . . 93
Declaration of Authorship . . . 96

Chapter 1

Introduction

1.1 Motivation

In the PowerIT project[1], implicit, interaction based energy management in common household environments was researched. One of the main questions of this project was if the activity of a user can be tracked by utilizing the accelerometer sensor of a user's smart watch.

The recognition of human activities by computers is a field that reaches back into the eighties of the last century. It gained increased attention by the widespread possibility to observe different sensors in the environment. The observed data is utilized to maintain a computational context model of the environment where significant changes lead to corresponding reaction of the system. Therefore activity recognition can be seen as an area of context computing.

The key objective was to design and develop a generic software framework that is realized in a modular and service oriented fashion to serve multiple practical requirements of the proposed system. An elaborate evaluation was necessary to provide evidence about the system quality under real world constraints.

First, an exploration of system requirements from a practical and scientific perspective was performed to identify the challenges that needed to be targeted. The determined requirements are illustrated in the following enlisting:

- The smart watch accelerometer sensor carried by a user needs to be logged continuously and collected data safely stored for extensive post analysis.
- Permanent, unobtrusive usage observation and minimal practical interaction.
- An elaborate method body to support analysis processes and allow real world application of obtained results.
- Possibility of implicit environment control with extracted results.

[1] PowerIT (IT for Implicit Interaction based Energy Management): FFG Call: FIT-IT Embedded Systems & Semiconductors, 2nd Call.

- The user must still be kept the sovereign of the system and decides when and how the system gets utilized.

These requirements already dictate a system that is practically handable in everyday usage. From this, general, non-functional requirements are derived that are afforded by a system that

i) is running virtually forever (on smart watch devices)

ii) requires a minimum of user interaction

iii) provides high availability and is self recoverable (silently dismisses periods of inoperation) and

iv) provides techniques that allow practical maintainence and result extraction.

Where existing work often tries to extract activities on a fine granular level (e.g., every single movement of every limb of the body; body and gesture postures are tried to be recovered automatically) this work is more concerned with long term aspects of activity recognition. Due to the practical application of the propsed solution in an all day setting a fully supervised setup where a system operator observes every step of the user is not possible. Instead, simple user interfaces has been developed that enables the user to perform data collection without supervision. To confirm the applicability of the system an extensive data collection process has been conducted with three particapants. Afterwards, the recognition ability that can be achieved on the collected data with this approach is evaluated.

After terms and definitions necessary for this work have been fixed, the state of the art gets revised by presenting an overview of existing work. The system architecture chapter presents the core mechanisms that were needed for continuous sensor data recording and following data analysis. After this, the application of the system and the main interaction interfaces are presented. This is followed by a thorough analysis and evaluation of the acquired data. In the last chapter the impact of this work is revised and corresponding conclusions are drawn.

1.2 Activities of Daily Living (ADL)

To track the everyday activities a user achieves during a day, the concept of *activities of daily living* (*ADL*) is an important notion. Although originally keyed in the domain of medicine and caregiving, in computational science it has a broader meaning in that the whole day of any user is observed and modelled computationally.

The work in [10] presents a concise tutorial about how an activity recognition system can be established conceptually and programmatically. Some of the design issues treated in this work could have been used for the challenges this

document is concerned with. Nonetheless, two aspects are left out or underrepresented:

i) The utilization of time use and time survey concepts and structures and

ii) possible application usages with the observed, tracked and evaluated information.

These objectives needed to be considered to enable activity recognition practically in everyday usage scenarios.

1.2.1 Time Use Survey (TUS)

One of the first questions after the specific requirements have been identified was, how to define an activity. For this purpose the American Time Use Survey (ATUS) study publicized by the United States Department Of Labor [2] provided a useful base to identify the activities interesting for our work.

In the ATUS a great number of people is asked to hand in questionaires where they record a minute-by-minute account of one single day. From these questionaires information is derived where the data is evaluated for demographic properties, like age, gender, employment or wage. This way the typical average day for certain kinds of persons can be drawn. The survey is taken every year and the data provided to the public over the mentioned reference.

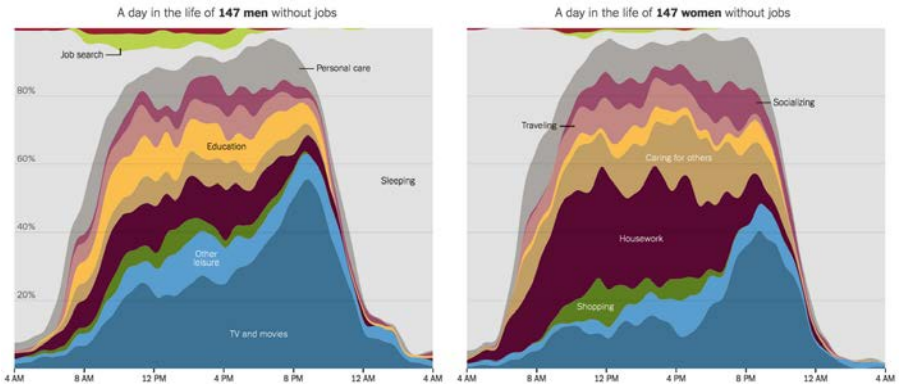


Figure 1.1: ATUS average day representation

In corresponding visual representations [3] often some contrast is exposed like employed versus unemployed people or women versus men. An example can be seen in Figure 1.1 where it is shown how much time is spent on which activity

[2] http://www.bls.gov/tus/home.htm [retrieved: 05, 2015]

[3] http://lede.io/articles/2015-03-03/how-nonemployed-americans-spend-their-weekdays-men-vs-women.html [retrieved: 05, 2015],
http://www.nytimes.com/interactive/2015/01/06/upshot/how-nonemployed-americans-spend-their-weekdays-men-vs-women.html?abt=0002&abg=1&_r=0 [retrieved: 05, 2015]

for unemployed males compared to unemployed females. The x-axis in the figure shows the timeline of a single day from 00:00 to 24:00 o'clock. On the y-axis the average of how many people do which activity at the certain instance of time is depicted. By this approach the corresponding stacked area plot is created.

This representations obfuscates certain details like the fact that it is not determinable if a little number of people is spending all of their time on one specific activity or if lots of people spend a little share on this activity but the specific value is only given by building the average. Therefore another chart type better maps the specific time ranges for every user which is shown in Figure 1.2 (a). Again, a comparison regarding the demographic property gender is conducted. With this representation the differences between the activities performed during a day can be elaborated on an individual basis. In Figure 1.2 (b) the color to activity mapping for the above figures is shown.

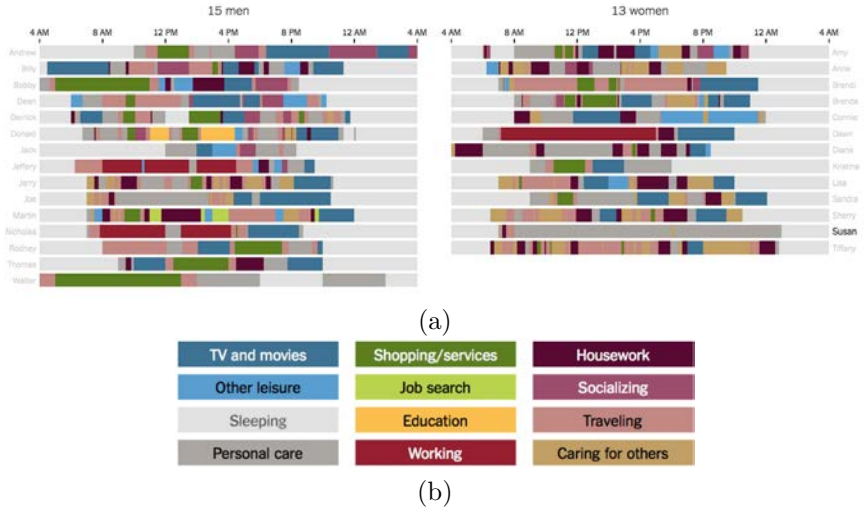


Figure 1.2: ATUS multiple individual day representations and color legend

For our purposes a representation approach similar to the one depicted in Figure 1.2 (a) is shown. One version that shows the activites of a single day of a user is presented in Figure 1.3. Again, the timeline for a single day is displayed on the x-axis, with consecutive items drawn, that represent the different activities the user performs during the day.

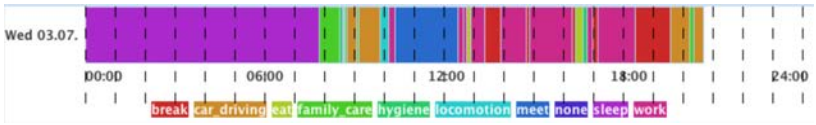


Figure 1.3: Single Day Activity Trace

On the y-axis an arbitrary number of different days can be drawn which is depicted in Figure 1.4. It has shown out, that a week is a decent unit of processing regarding required computational resources which is manually handable in a convenient way. Details, regarding the timeline views and their utilization in this work, are discussed in Chapter 4.

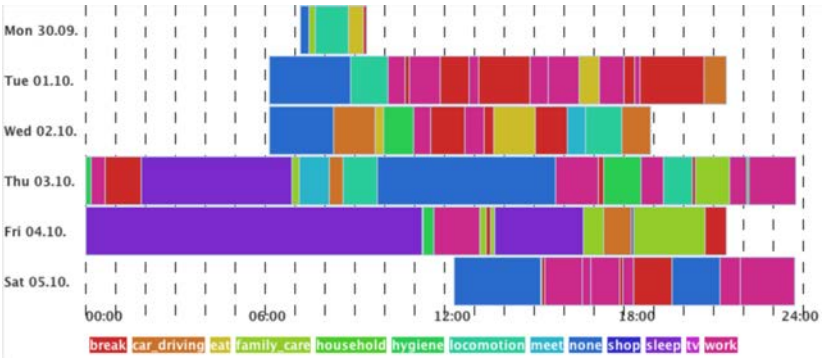


Figure 1.4: Single Week Activity Trace

1.2.2 Activity Terminology

After this little excursion regarding the visualization of user activities we come back to our original question about which activities to specify for application in our framework. The legend in Figure 1.2 (b) already presents some indication of which activities will be of interest.

In our approach activities are seen primarily as situations instead of dealing with a very detailed granularity like hands up or down or step forward or backward, for example. The reason for this is that fine granular activity training is a very elaborate task and infeasable within practical, everyday, real-world deplyoments as it handicaps the user too much from achieving the usual daily routine.

Not all of the above activity items are directly applicable to be collected by our smart watch sensor platform as for example “job serach” or “volunteering” could not directly be represented by a hardware sensor of our type. Therefore, only activities that could easily be differentiated within the given application case had been utilized.

Additionally, at the start of the experimental data collection, feedback of the participants was used to finally determine the set of activities that was deployed in the eventual runtime system. The activities that were considered in our system are the following:

• Break	• Entertainment	• Meet	• Sleep
• Car Driving	• Family Care	• Don't Care (None)	• Socialize
• Computer	• Household	• Pray	• Sports
• Cook	• Hygiene	• Relax-Think	• Travel
• Eat	• Locomotion	• Shop	• TV
• Education			• Work
			• Write

Table 1.1: Relevant Activities

To make activity recognition computationally accessible, continuous parts of signal streams have to be augmented with some corresponding, high level activity description that represents a natural activity in the common human sense. So when an activity like "walk" is performed the common interpretation of taking a step after a step to change between locations is meant. Of course does any activity, dependent on the person or individual performing it, have particular characteristics, but these do not change the general meaning.

After the activities interesting for our work have been specified in accordance to the already presented timeline representations, concepts that are essential for the rest of the explanation are:

- *Activity Type*: The type of an activity is its description according to the possible assignments as given by Table 1.1. The activity type does represent the same *activity class*, which is relevant for later activity pattern classification.
- *Activity Item*: An activity item is the smallest element that denotes the sensor data sequence that is assigned with an activity type from the start until the end of the item. An activity item is described by certain properties like start timestamp, end timestamp, number of sensor samples or average sensor values. Throughout this document the terms activity label and activity item are used interchangably.
- *Activity Trace*: Is a sequence of activity items ordered sequentially over time. This way corresponding activity traces are automatically retrieved by a user assigning data with corresponding activity labels over the smart watch applications. By aligning the main activities a user performs during the day, one after the other, a consecutive trace of activities is obtained. This trace is then used to try to recover the same activities performed at other instances of time by applying several recovery routines that are addressed in a later section.

- *Activity Frame*: This is the unit of data that is used for visualization or classification. It is specified as a temporal property by denoting a chunk of data that can reach over an hour, a day, a week, a month or a year. The yearly resolution is out of scope of this work as we do not have the data for such a period of time. In evaluation it turned out that a week is the best size for an activity frame regarding comparability between frames and computational efficiency.

This definitions lead to two further related concepts that are needed when on one side the activity assignment to the data by the user is considered and the other time, when the activity tries to be estimated by the system by application of a previously learnt recognition model:

- *Activity Label*: This is the framework representation of an activity item and is the entity that is assigned by the user at execution of the system. *The process of assigning activity items to the sensor data is also called activity labeling.* Each activity label is a particular instance of an activity item. The activity label is used in combination with the collected sensor data to perform the activity recognition.

- *Predicted Label*: The predicted label is the activity item that results from passing an activity label with its corresponding numerical sensor signal data to a classification routine. Depending on the equality of an activity label to the extracted predicted label the classification is correct or incorrect. Implications of these classification results are discussed in Chapter 5.

To emphasize the distinction to the representations drawn from the ATUS it has to be noted that:

- In ATUS only a single day per participant is recorded (although a far higher number of particiapants could have been used - this might be an issue for further work; cf. Section 6.1) and that the activities of participants are used to elaborate differences in lifestyle based on demographic criteria whereas in this work the recognition of recurring activities based on sensor signal data and trace period analysis is pursued. The ATUS target shall be handable by our system by achieving widespread application.

- Although representations are similar, in this work a different approach is taken by trying to map the day of a user automatically. The application of a user worn smart watch sensor platform does not allow to observe every activity that is considered in the ATUS, but to track activities that can be recorded by our system.

1.3 The ARC Design Pattern

Conventionally, when talking about computational activity recognition the *activity recognition chain* (ARC, see [10], [38] or [34]) is an essential concept that

describes the necessary processing steps to extract information derived from raw sensor data that is transformed into high level context information.

1.3.1 Traditional ARC

The classical activity recognition chain (ARC) as it is used throughout related work is presented in Figure 1.5. The different stages are Preprocessing, Segmentation, Feature Extraction and Classification. Raw sensor data gets preprocessed over signal filters and similar methods and is ordered into chunks of data, so called segments. Feature extraction calculates specific key values from the preprocessed data segments from which the final classification stage tries to infer specific patterns.

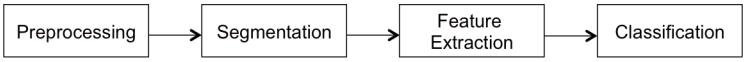


Figure 1.5: The ARC design pattern

This is a very brief overview of how an ARC is designed and more detail is presented in Chapter 3. Still, extensions are possible as explained in the next section.

1.3.2 ARC with Extension

The ARC, as it is handled in literature, omits several intermediate stages whether by including them into existing stages or assuming the information from this intermediate stages as provided. In this work, these intermediate stages are made explicit, because some key aspects of this work relate on them.

Figure 1.6: The Extended ARC design pattern

They are depicted as dashed boxes in Figure 1.6 and explained in the following bullet list.

- Data Recording: The data acquisition and collection stage is often assumed as a sub-process of the preprocessing stage but made explicit here because a live runtime system has been specified and implemented that is better mapped by a separate construct than integrated in a stage with a different purpose.
- Feature Filter: In this work it was experienced that filtering is not only of interest to the preprocessing and segmentation stages (where filtering usually is performed) but also the filtering of featured data does have great influence to the classification stage.

- Application: Application handling interfaces have been specified and corresponding actions implemented to provide environment control commands based on computationally recognized activities a user is performing. It is denoted explicitly here because corresponding connection interfaces have been specified and implemented.

Details regarding the extensions to the ARC are addressed in Chapter 3, where their technical specification and implementation considerations are explained.

1.4 Research Objectives

At the end of this chapter the most important questions that this work is concerned with and tries to answer are summarized again. Throughout this work, these questions are explored to great detail to eventually provide the base argumentation for the profound answers given.

- Can activities be recognized by the application of a smart watch accelerometer data collection and evaluation system. How accurate can activities be recognized.
- Which aspects must be considered to meet real-world requirements with special regard to every day use and practical result utilization. Which constraints apply in practical application and how can they be handled.
- Which functionalities and analysis routines must a corresponding system implementation support to meet the requirements mentioned in Section 1.1, Motivation.
- Can initial recognition accuracy be increased? Show strategies to improve recognition results.

Chapter 2

Related Work

2.1 Context Computing and Activity Recognition

In a broad sense, activity recognition is a subset of context recognition, more generally context computing. Observing the context of users can happen in various situations and is therefore not deterministically framed or formally specified.

Context-aware computing was defined by [40] in times when the first personal home computer wave faded out and the internet started to gain widespread attention in academics and business. Around ten years after [18] presented a toolkit that allowed the maintenance and operation of real-world context in the computing domain for arbitrary application cases.

A valuable overview of the current state of context computing can be found in [33] where different approaches throughout the last years are compared against each other. Conclusions are drawn how context computing manifests nowadays and may emerge in the future.

Accelerometer based activity recognition is a subset of context computing in that it eventually maintains some definition of context computationally that is related to activity recognition. It goes back around the turn of the millenium where in [27] different feature types are extracted and simple natural activities like "walking" or "stepping stairs" are analyzed. A classic work in this sense is given by [4] as it was one of the first that provided evaluation results based on practical settings. This work concentrates on the movement apparatus of the human body. Interestingly, the concepts coined in this time slowly start to weaken to see activity recognition in a broader context and integrate other concepts to the ones presented in this work.

[34] performs data acquisition with a custom mobile platform that sends sensor data over wireless transport to a mobile handheld computer. Remarkably about this work is that a rather vast analysis of different classification algorithms is presented.

Other examples that apply mainly the same methods for different use cases can be found in [42], [33] or [17]. In [42] an ARC is used to analyze household activities based on the amount of water taken from water pipes installed in water basins in the kitchen or bathroom. [33] presents a survey regarding context computing in the internet of things domain. It revises existing approaches on a very broad scale with accelerometer based activity recognition being a part of the global recognition approach presented. [17] even goes this far to trace the activities of animals (birds in this case) to derive different behavior patterns.

Activity recognition from a technical view is nowadays seen as implementing the ARC pattern (cf. [38]), addressing specific use cases. In this work the recognition task is specialized into several processes that are applied subsequently to extract high level activity labels for input sensor and label data. The ARC processes utilized are: Sensor data acquisition, preprocessing and segmentation, classification and classification result application.

2.2 Approaches to Computational Activity Recognition

Activity recognition in the domain of context computing can be achieved by applying different computational approaches. Some examples are listed below (in arbitrary order):

- Environmental sensors are a superclass of some of the below examples like visual and audible recognition but also other sensors can be utilized to observe parts of the sensors surrondings. Video observation is a very popular approach as addressed in [11] or [28].
- Inertial sensors: Are mainly used for observing physical properties like velocity, orientation, and gravitational forces. All accelerometer, gyrometer and similar sensors that are referenced in this work fall into this category. A vast literature exists, for examples see [48] or [2].
- Audible Recognition: For obstacle detection and recognition over ultra sonic responses or the classification of various sounds in the human audible spectrum. For an example see [41].
- Personal sensors: Smartphone and mobile sensors contained therein as addressed by[30].
- Arbitrary usage patterns regarding mechanical and computational appliances as utilized in [**?**].
- The time use survey method mentioned in the introduction is gaining more interest in ubiquitous computing as addressed in [32] or [8, 9].

From this brief list of examples it is already clear, that general activity recognition is a broad field that can be applied to various computational observation

models. A good overview of existing work mainly regarding activity recognition with inertial sensors can be found in [22]. The second part of this work is concerned with enhancing mobile sensor battery lifetime by adapting sensor sampling rates dependent on performed and recognized activities.

A recomendable introductory reading is presented in [10] where standard activity recognition approaches are discussed. Also, different accelerometer sensor placements on the human body are evaluated against each other for their recognition accuracy.

Activity Specifications

Usually, throughout existing literature a set of common sense activities are used to augment different kinds of raw sensor signal data or feature streams. In [37] a dense set of common low-level household activities like cleaning dishes or tables, preparing coffee (with all sub-activities like cup to machine, add sugar and so on) or open doors of kitchen furniture are considered.

Modes of locomotion (e.g., walk, sit, stand, lie, run, ...) are an often researched set of activities. Examples can be found in [34] where additionally household and personal care activities like vacuuming, brushing or situps are addressed.

A different sensor approach, where activities are tried to be detected by the usage of water at a water sink in the kitchen and bathroom, is presented in [42]. The activities that were considered are shave, brush teeth, wash hands, flush toilet, fill up teakettle, make a salad, rinse a fruit, take a glass of water, do dishes (light and heavy load).

A high number of concepts for activity definition exists. In this work, mainly the ATUS approach for activity specification was followed, due to reasons already mentioned in the introduction. How this is put in the context of our work is addressed in a section below.

2.3 Wrist Worn Based Activity Recognition

For wrist worn based activity recognition it is necessary to revise existing work specific to this branch of activity recognition research. Existing wrist worn wearable platforms are enlisted as well as developments and approaches building on these platforms need to be considered.

In wearable device markets, wrist wearable platforms are a product category that came in the focus of research only a few years ago in the form of smart watches. Also more intimate wearables appeared as armbands currently mainly used for fitness applications. Here we talk about *wrist wearable platforms* when smartwatches or armbands are considered.

As smart watches are a rather young device category, their utilization in research is not that mature as with other classes like smartphones or wristbands. Their application for activity recognition is pursued in different directions. In [29] different sequences of fitness exercises are tried to be recognized, whereas

in [5] classical activities (walk, run, cycle, rest/office work, active, sleep) are revised with low sensor sample rates.

Inertial sensor equipped wristbands in combination with physiological sensors that measure signals like heart rate or skin humidity are popular in healthcare and fitness research like in [2].

Accelerometer based gesture recognition is often treated as a subdomain of activity recognition. [45] shows the utilization of the accelerometer sensor to provide interactions on the smart watch directly. [16] develops a gesture alphabet to map different letters to certain gestures.

Smart watch research is currently also very active in the domain of interaction as addressed in [6].

2.4 Time Use Based Activity Recognition

Motivated in the introduction, time use based activity consideration is getting increased attention in the last years. Borazio and Laerhoven [8] make use of the ATUS to revise how accurate the properties exclusive to this study can be used to recall certain patterns. Besides the time of certain activities also the age or location of participants, at the time the activity is performed, is considered.

By the same authors (cf. [9]) an extension is propsed to combine body worn accelerometer data with time use properties explored in the previous work. They conducted an experimental study performed by 17 participants over a period of 14 days each. This approach is close to the one presented in this work. An essential difference is that in our work less people have participated but each of them for a longer period of time. Altogether, in both works nearly the same number of days were observed.

A related, alternative approach is proposed in [21], where a method is shown that puts long term data into histograms and performing classification on it. Also the problem of correct alignment according to the time when an activity is really performed practically may significantly influence classification results. This property is again discussed in the evaluation chapter of this work and a related approach to correct wrongly assigned start and stop timestamps of activites automatically is proposed in [25].

An annomination with the upcoming keyword “quantified self” is the “editable self” presented in [31], where a web based approach to activity labelling of time use based data is presented.

2.5 High Level Activity Modelling

For a top-down approach to activity recognition often generic models try to get established whose theoretical concepts are mapped onto specific computational requirements. This is also known under the term symbolic processing and ontologies have proven as a viable method to maintain such models.

Ontologies can provide a high level model description of how activities are interrelated. Although ontologies basically are a generic philosophical construct that is used widely throughout scientific domains, in this work we only consider their application within the activity recognition domain.

As an example, extensive work about the structure of formalisms that can be packed into corresponding ontologies for activity recognition in the ubiquitous and pervasive computing domain can be found in [46] and [47]. There, the global relationships between the user, its environment and how the interaction of the user with this environment is happening, are mapped. The established models are evaluated in common household situations, e.g., the user is in the living room interacting with the TV set. For this purpose also natural objects, like the TV set or the belonging remote control are considered by the ontological model. Similar concepts are explored in [12] and [13].

Another elaborate model is contained in Opportunity Framework [24] that is a continuation of the work in [37] where a corresponding high level model is defined above the initial model description to technically describe low level activities on an abstract layer.

There are several endeavours to extend ontology language formats by adding parameters that were not intended in original Web Ontology Language (OWL). [19] for example, adds probabilistic parameters to connectors between entities to denote that relationships are not fully static but follow some random pattern that is tried to represent. Another work, where a similar approach is followed, can be found in [35].[36] extends OWL with timing constraints to express temporal properties between connections like the semantics of one entity can only be revised after an upstream entity has been examined or that some transition can only be taken at a certain wall clock time. This approach also shows enhancements in the retrieved results.

Where ontologies provide an extensive field of reasearch on their own, alternate approaches provide models that take numerical and statistical properties more into account (this is not considered in pure ontologies at all). [14] provides an approach that is built on conditional random fields (CRF) and semantic attribute sequences. For evaluation it makes mainly use of exercise data in fitness training. A similar direction is followed in [15].

[39] provides a context-driven activity theory where probabilistic and Markov chain theory is used to assign complex activity signatures to streams of data and derive corresponding activity definitions from signature sequences generated from these streams.

2.6 General Data Acquisition and Pattern Classification Tools

General purpose signal data acquisition tools are the CRN Toolbox [3], Device Analyzer [43] or Funf [1]. They are general purpose in the sense that they provide

[1] http://www.funf.org [retrieved: 05, 2015]

extensible interfaces and generic data objects that are then applied to specific use cases to show the feasibility of invented models and operations.

There are already several solutions that serve pattern recognition in multiple respects, starting with support for feature extraction and decimation up to presentation of evaluation results in multiple display versions and textual formats suitable for post processing. Prominent representatives for pattern matching software toolkits are:

- KNIME: http://www.knime.org [retrieved: 05, 2015]
- RapidMiner: https://rapidminer.com [retrieved: 05, 2015]
- WEKA: http://www.cs.waikato.ac.nz/ml/weka/ [retrieved: 05, 2015]
- MOA: http://moa.cms.waikato.ac.nz [retrieved: 05, 2015], stream classification

These example were chosen because they are used widely throughout academia. There exists multiple other examples throughout science and economics. Other examples include ODM, Shogun Toolbox, Orange, Apache Mahout, MCMLL.

The purpose of these tools is mainly to set up analysis models that can be evaluated in different ways to find out possibilities to re-detect relatives of a given data set or set of data sets. What is not treated by these tools is the fact that they are not constructed for near real time requirements and therefore they do not consider this aspect at all. But in our case exactly this requirement gives one of the main contributions this work addresses. Fortunately, it is possible to build up on the concepts also handled by general purpose tools that are utilized in the prototype framework reported in this work.

Details regarding the methodological concepts and mathematical routines regarding machine learning and pattern recognition and classification can be found in the standard works of [20], [7] or [1]. For the purposes of this work, mainly the WEKA framework was reused.

Chapter 3

System Architecture

To provide activity recognition on a practical level, core processing and communication functions needed to get implemented over which the activity reconition related routines can be handled. The components that must be available are:

- Continuous sensor data collection, logging and saving to persistent storage.
- The ability to assign activity labels to recorded data at the time the sensor data emerges.
- A compact data format that allows further processing steps.

Finally, corresponding interaction methods need to be implemented to make the system user ready for deployment in the experimental evaluation. They are treated in the following Chapter 4. In the following paragraphs first the core runtime components are described and afterwards the framework parts regarding activity recognition are described in detail.

3.1 Core Runtime Components

The dynamics of the system regarding processing capabilities and communications are presented in [23] and are briefly summarized in the following paragraphs. The diagram presented in Figure 3.1 provides an overview of the components specified within the framework. They are categorized into the following sections:

- The *System Environment* provides the local single point of access component that is responsible for registration of communication channels, passing incoming data to interested listeners and managing system configuration items.
- Items of the type *Producer* provide the sources for sensor data and socket instances to retrieve sensor and derived data from remote system entities and forwarding it to the local framework.

- *Consumers* can be registered if they have to listen for data of a certain type emitted by a corresponding producer. It is then up to the implementation of the specific consumer how received data will get further processed.
- The *Task Manager* has a special role as it implicitly reuses the core producer and consumer mechanisms, this time for the purpose of activity recognition.

It has to be noted, that data emitted by producers for which no consumers are registered will get silently dismissed. The same applies for consumers where no corresponding producer is available. Such consumers will never retrieve any data and therefore will never trigger their processing operations.

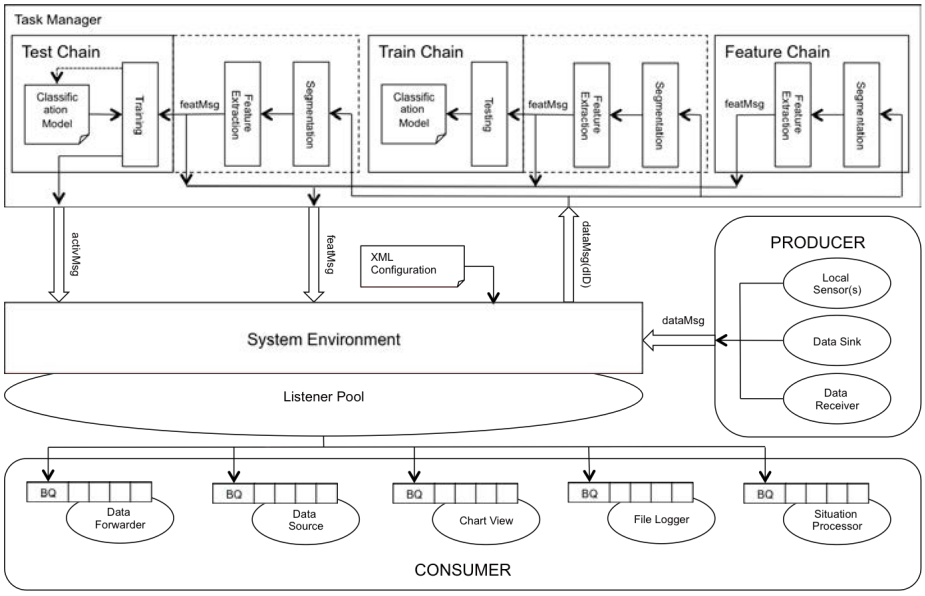


Figure 3.1: Local System Archtecture Diagram

All the mentioned components are realized as OSGi bundles which makes the framework very flexible. This way it is possible to exchange parts of the framework even during runtime without hindering other parts of the system not directly in relation to the exchanged component.

Service Oriented Architecture

To meet the practical requirements denoted in the introduction section, it was necessary to specify an architecture that is able to provide a continuous service that is (at least partially) capable of self recovery to maximize user experience and system availability. The presented setup was developed within the PowerIT project with aspects of maximization of code reusability, ease of extensibility,

component oriented design and runtime performance in mind. With this approach it was possible to provide a multi platform capable software solution that is able to execute in an unattended fashion on device classes in the range from ordinary desktop computers down to smart watch sensor platforms with modified configurations but without the necessity to cross compile any software components. This is also owed to the fact that Java has been chosen as the implementation programming language and reusing its multi platform capability.

3.1.1 System Environment

The System Environment is the central item in the framwork as it

- maintains the system configuration as well as communication and component registrations.
- provides the generic classes for process execution, multi threading and data transmission over corresponding queues.
- implements socket components for retrieving data from remote sensors and forwarding them to interested remote listeners.
- includes the object capsules regarding different data types transmitted throughout the framework.

The System Environment has no direct operation on its own but just provides the components that will get utilized by classes that perform any concrete task. Producers push data to the listener pool where registered consumers are notified in accordance to the threads implemented by them.

3.1.2 System Configuration and Configuration Items

The system configuration file is the central component of the framework, responsible to instantiate the core functionalities that are later required to enable activity recognition mechanisms.

The system architecture is divided into three main categories that are enlisted below. A graphical representation of the system architecture is shown in Figure 3.1. There, the system components available locally on a host node are depicted.

1. The base functions like configuration handling, message passing and thread pools.
2. Communications including device and service discovery and transmission of different message types.
3. Activity recognition, locally and remotely.

The system core consists of a structure for managing the runtime environment of the framework which also includes a listener pool to which elements interested in any specific messages register. To forward and distribute messages within the system a producer - consumer pattern is specified where producers offer data to the system from local or remote sensor sources and consumers forward data to remote hosts or are responsible for local persistence.

Initially, a communication system that allows the propagation of sensor data from low power embedded devices up to full-blown server installations to access the data at different device instances with varying computation performance was designed and developed. The system implements general patterns that allow the execution of the same software on different host nodes with varying configurations.

Listing 3.1: Runtime XML configuration

```
<config>
(a)     <system>
                <devicename>MotoACTV</devicename>
                <is-file-log>true</is-file-log>
                <is-act-rec>true</is-act-rec>
                <acc-sensor>true</acc-sensor>
                <backup-period>3600000</backup-period>
                <nrows>2</nrows>
                <ncols>2</ncols>
                <autostart>true</autostart>
                <nightmode>true</nightmode>
                <log-types>acc,feat,rssi</log-types>
        </system>
(b)     <fwds>
                <fwd>
                        <name>GGPH</name>
                        <type>feat</type>
                        <conn>bt</conn>
                        <mac>AC:22:0B:A4:22:93</mac>
                        <uuid>00001101-0000-1000-8000-00805F916001</uuid>
                </fwd>
        </fwds>
(c)     <rcvs>
                <rcv>
                        <name>HS</name>
                        <type>feat</type>
                        <conn>ip</conn>
                        <ip>10.0.0.1</ip>
                        <port>16001</port>
                </rcv>
        </rcvs>
(d)     <snks>
                <snk>
                        <name>GGPH-ctrl</name>
                        <type>ctrl</type>
                        <conn>bt</conn>
                        <uuid>00001101-0000-1000-8000-00805F918001</uuid>
                </snk>
        </snks>
(e)     <srcs>
                <src>
                        <name>GGPH-cnt</name>
                        <type>cnt</type>
                        <conn>ip</conn>
                        <port>22001</port>
                </src>
        </srcs>
</config>
```

Depending on the configuration of every host node, an end device can take a different role, which is explained in the following subsection. The configuration depicted in Listing 3.1 represents a default configuration that illustrates possible settings. It is used as example to enlist the main configuration items that will get deployed across the nodes included in the system.

The config item (a) defines basic system control properties, whereas items (b) - (e) are different types of socket connections to distribute data between involved devices. The meaning of the configuration items is as follows:

(a) System: This item defines general system properties which inform if local sensors (if available), file logging and activity recognition are enabled. This already partially defines the role (cf. Section 3.1.2.1) of the host, on which this configuration item is deployed. Additional elements state which types of sensors get logged and some infrastructure settings like view extents or if components get started automatically.

(b) Forwarder (fwd): This is a client item which connects to a corresponding sink to actively forward data to a remote host.

(c) Receiver (rcv): This is the opposite of the above forwarder in that it actively requests data from a remote source.

(d) Sink (snk): The sink is a server entity that receives data from connected forwarders and pushes this data to the local framework instance.

(e) Source (src): The source on the other side is used to provide data to interested receiver clients.

Without any connection configurations only local operation would be possible. For full adaptive participation in the system the corresponding communication channels need to be stated.

Although it is not shown in the listing, it is possible to setup several connections of every type together to form the different roles required in the system. Additionally, it is possible to drive different connection types simultaneously which even extends the number of possibilities how system parts can be connected together.

By using this definition for system entities, a message passing system in the style of a Model-View-Controller (MVC) [26] design pattern has been specified. From the configuration items presented in the listing, it can be seen that IP, as well as bluetooth based connections are possible (as depicted in configuration sections (b) and (c)).

3.1.2.1 Device Roles and Operation Modes

A notable aspect of this generic configuration setup is that when multiple devices are included in a deployment, those devices can take on different roles. It is possible to execute the same framework on different platforms where only the

corresponding configuration has to be modified dependent on the deployment platform.

The advantage of this approach is that resource intensive processes can be moved to more capable devices and that complex interaction mechanism can for example be performed on a smartphone and executed on a smart watch. The considered device roles are:

- *End device*, e.g., smart watch; the notation end device is used here because this kind of platform serves as the sensor data source.
- *Proxy*, e.g., smartphone: A smartphone could also be used for activity recognition the same way as presented in this work. But we concentrated on smart watch sensor deployments exclusively due to the usage behavior of such a device. Nonetheless, can the smartphone be used as an interaction medium and an intermediate storage base from where sensor data will get forwarded to a backup server.
- Home automation/home entertainment *central hub*; this role is owed to the project mentioned in the introduction to provide an intermediate layer for the storage hierarchy.
- Backup *server*; this is the location where all the sensor data is eventually stored and taken from there for analysis.

Additionally, depending on the particular platform the framework is executed on, different modes of operation and interaction are emerging.

On a standard desktop PC extensive elaborate models in a posterior, offline fashion can be handled and evaluated that can allocate several gigabytes of memory. On the contrary, low power smart watch sensor platforms require different runtime features regarding processing power, memory utilization and battery lifetime. The main modes of execution that are considered in this work are:

- Watch stand-alone: In this mode the data collection process has been achieved.
- Watch-smartphone combination: This mode was used to examine certain interaction mechanisms.
- Main server for evaluation: Due to the large scale of data, evaluation had to be handled on sufficiently equipped machines.

How the different device roles are utilized in runtime deployments is presented in Chapter 4. Evaluation is concerned with the main server mode as this is the location where the data of all participants get stored and processed for analysis.

3.1.3 Producer

The Producers in the system are differentiated into two types:

- Local Sensors: This type of data producer is used when the sensor source is available as a piece of hardware on the local host on which the framework is executed. Adhering to a generic interface additional sensors can be integrated without much difficulty. Also due to the requirements in the belonging PowerIT project the sensors mainly utilized are: accelerometers, energy readers and data types derived from these basic types (like featured data).
- Sockets: To make the data of local sensors available remotely corresponding socket types were established. The data sink implements a server where data forwarders (implemented as consumers when seen from the local framework side) push the data to the local framework. This happens in a passive fashion. The data receiver on the other side pro-actively requests remote sensor data from a corresponding data source.

3.1.4 Consumer

The different types of consumers required in the framework are as follows:

- Sockets: Data Source and Data Forwarder provide the sockets for remote hosts to access the data emerging at the local entity. At the remote host the corresponding producers have to be set up to perform successful data transmission.
- Logging: Records all types of data transmitted through the framework regarding the set items in the system configurations. Details regarding data logging are explained in the next section with the application of storing activity recognition relevant data.
- View: Similar as with logging, corresponding listeners are present to register for framework data but this time not for persistence but displaying it in an online fashion.
- Situation Processor: Is a generic analogous type as the task manager but here only with the purpose to retrieve items derived from sensor data that can be utilized for upper level system control and the like.

This non directly activity recognition related system setup was described to clarify the runtime environment under which activity recognition will be performed. Without these base components and their corresponding interplay it would not have been possible to provide a system that is capable of unattentive and continuous operation with a minimum of user interaction.

3.1.5 Message Types and Processing

After explaining how data will get exchanged within the framework, the types of data and how the core handling is done, is presented. A generic interface has been specified that all message types will have to extend and implement to be usable by the system. The following paragraphs explain the types of messages implemented for our system.

An interface of messages provides the top level structure for all messages in the system and all components interested in messages basically bind on this interface for receiving messages. Specialization reduces the scope of types the component interested in them shall be able to handle. As a message describing an activity for any message at any one time can be important for all different messages in the system, by convention an activity message interface has been added to which all actual message types are bound. This way it is ensured that every message has activity information contained although it is not used all times. Messages, for which the addition of activity information is not strictly necessary, can be directly inherited from the top level message interface. This is the case of command or control messages.

The different message types the presented framework is able to process are:

Raw Accelerometer	RSSI
Segment/Window	Position Virtual
Featured Accelerometer	GPS
Activity	Energy Meter Readings
Command	Temperature
Control	Humidity

The different supported message data types are mentioned to show the versatility of the framework. Generally, the following methods can be applied to every type of message data the framework is capable of. In this work the message format for the accelerometer is treated, as this is the most common sensor for wrist worn smart watches. In this work only raw and featured accelerometer and for testing control messages are considered that are shown on the left side in the message type enlisting above.

The other message types (shown on the right in the above listing) are of relevance in the PowerIT project where behavior analysis dependent not only on persons activities but also their locations and the energy consumption of household appliances is researched. Also other infrastructure sensors like temperature or humidity has been considered there.

- General message format

deviceid	type	count	timestamp

  The general message format consists of a device identification stating from which sensor device the message originated as well as a specific type, denoting the kind of sensor data. The count field is used for internal management and debugging and the timestamp is the Unix wall clock time, the message is emitted to the system.

- Activity message format

deviceid	type	count	timestamp
situation label	activity label	predicted label	

The activity message format is directly derived from from the top level message format and adds fields regarding activity recognition. The "situation label" field can be used to determine more global situation assignments. Currently it is used to switch between different activity sets, which is described in Section 4.1.1. The "activity label" is the user set activity during data labeling. The "predicted label" contains the label that was derived during classification. It is used to determine classifier accuracy by comparing it to the user provided "activity label".

- Accelerometer message format

deviceid	type	count	timestamp
situation label	activity label	predicted label	
accX	accY	accZ	

Extending the activity message format three double values denoting the raw sensor values of the accelerometer sensor at the time the data was queried from the sensor are added.

- Feature message format

deviceid	type	count	timestamp		
situation label	activity label	predicted label			
meanX	meanY	meanZ	varX	varY	varZ

The feature message format represents a high level representation of a set of activity messages, containing the mean and variance for each of the three accelerometer sensor axes. How exactly the features are calculated are described below in Section 3.2.1.

- Command message format

deviceid	type	count	timestamp
command			

It can be seen that the command message format is directly inherited from the top level message format and only adds a field that contains a specific command. This message type is used to control different parts of the framework, by emitting different commands during runtime like starting or stopping a specific component.

By specification of hierarchies of message types, generalization can be achieved when a component is able to process arbitrary types of messages and specialization is realized when a component can handle only one or some of the specialized message types.

3.1.5.1 Raw and Feature Data Examples

Data is persisted in raw and featured form. The raw format is used to be able to afterwards simulatively perform the extraction of alternate features and for simulation reasons. Featured data is used to directly feed recognition processes as described in Section 3.2. The following paragraphs provide examples of raw and featured data sets.

Raw Data Format Listing 3.2 provides a reduced example of a raw data CSV file. The header is shown in the first line and corresponds to the message format described above. For presentation reasons the accelerometer sensor data samples are shown with only two decimal places. In the practical deployment the full range of the double data type is used to keep the highest possible precision. Also, only a few examples are shown with the dashes denoting more data samples between the depicted messages.

Listing 3.2: Raw Data Format Example

```
sid,typ,cnt,ts,sl,al,pl,acc_x,acc_y,acc_z
MotoCX,wws,19765,1372942801000,adl_situation,work,none,-0.91,-2.75,9.50
MotoCX,wws,19766,1372942801349,adl_situation,work,none,-1.22,-2.75,9.80
...
MotoCX,wws,19782,1372942801986,adl_situation,work,none,-2.14,-0.61,10.11
MotoCX,wws,19783,1372942802025,adl_situation,work,none,-0.61,0.30,8.88
...
MotoCX,wws,19803,1372942802978,adl_situation,work,none,-1.53,-5.51,9.50
MotoCX,wws,19804,1372942803017,adl_situation,work,none,-1.53,-2.45,10.11
...
```

Feature Data Format In Listing 3.3 a corresponding feature message listing is depicted. The example has been trimed the same way as raw data samples above. It can be seen that the long data type timestamp only shows second precision. This corresponds to the processing directive, that a feature vector is calculated for every one second window.

Listing 3.3: Feature Data Example

```
sid,typ,cnt,ts,sl,al,pl,mean_x,mean_y,mean_z,sd_x,sd_y,sd_z
MotoCX,feat,2986,1372942801000,adl_situation,work,none,-0.94,-0.96,4.98,
        1.04,5.42,0.59
MotoCX,feat,2987,1372942802000,adl_situation,work,none,-1.02,-0.88,4.95,
        2.41,17.57,1.89
MotoCX,feat,2988,1372942803000,adl_situation,work,none,-1.00,-0.92,4.95,
        2.28,2.80,0.98
...
```

3.1.6 Data Storage Organization

In this section the data persisting routines are presented. They are made up in a way, so that the user is unaware of the techniques running in background and does not have to perform any interaction related to data collection.

The persistence of collected sensor data as well as any accompanying data sets was handled by allocating a special directory in the users home directory under which all items required by the framework will get persisted. In case of a cold start (without any prior data items existing) default items are put into according subdirectories.

The different subfolders that are required for data logging, analysis and result storage are depicted in Figure 3.2. The shown directory structure is taken from a development deployment for presentation reasons. In a full practical deployment especially the recorded data under the "logs" folder would be larger.

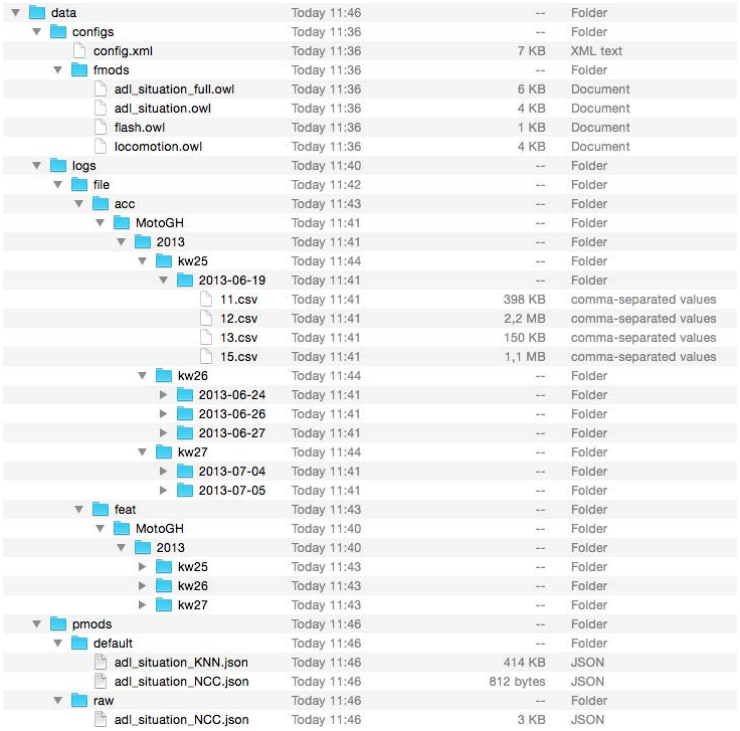


Figure 3.2: Framework directory structure

By default a "data" directory where all storage related items, the framework is concerned with, will get created in the users home directory. Then three different subdirectories are created that are explained in the following bullet list:

- configs: Here the available configuration items are stored. They contain

the config.xml file that has already been discussed in Section 3.1.2 as well as formal model files that provide the commonly available activities used for labelling, training and classification.

- logs: This is the directory where all data collected by the framework gets persisted.
- pmods: In this directory the model files that result from the training stage of classification are put. Their format and application are described in Section 3.2.4. They are part of the practical runtime system and have therefore a slightly changed meaning to the file conventions used in offline data analysis.

On every new day a new folder for this day gets created. The generic data logging subdirectory structure is kept under the logs directory and has the follwing characteristic:

- *file*: this directory type denotes the kind of persistent storage; here only file is used, with "db" as another possible option (unused in this work).
- *sensor type* (e.g., acc or feat): for every type of sensor a new subdirectory gets created to make sensor data available ordered by the type of sensor from which it emerged.
- *specific platform name* (e.g., MotoN): This directory represents a human readable name of the specific sensor platform the data emerged from. In case of the three participants contributing to the experiment conducted in this work, three different subdirectories are allocated here.
- *wall clock year*: The structuring by year is based on the thought, that for very long term observations a yearly assignment supports in easier relocation of data.
- *week of year*: As further described in Chapter 5, this subdivision is used to provide data chunks that are big enough to extract representitive results and small enough to be processed in a performant way.
- *day of year*: This subdirectory is used to keep the day identifier in the directory structure instead of adding it to the file name. Also for testing and development purposes the data sizes are more convenient for a day than for a week.

Within each "day of year" directory a new comma separated value (CSV) file gets created for every new hour. This approach was chosen to increase overall stability. If, for example, the whole framework crashes only the currently open hour file might be corrupted and not a data item representing the data for a longer time range.

Logging then, works fully automatic. Every new sensor data sample is passed to a top level instance of the logging logic which then determines the subdirectory structure and filename, allocates corresponding structures and eventually writes the data sample contents to the specific file.

Moreover, a batch backup job was implemented that periodically moves saved data to a backup server. This was added to ensure that no data will get lost and as an easy mean of access for offline analysis without access to the mobile platform.

One important aspect to mention, related to data storage, is that in this work only file storage methods have been utilized, exclusively. A DBMS approach has been evaluated but dismissed, as the available DBMS systems (namely SQLite) do not show the required performance, especially on low resource mobile platforms like the smart watch utilized.

If later a DB storage strategy for already existing data would be required, corresponding transformations need to be implemented. This might become necessary for bulk analysis of large data sets, e.g., for longer periods than have been recorded for this work. On mobile platforms the file based approach has to be preferred as existing DBMS solutions start to generate significant file writing lags, especially for moderate to high frequent updates which is the case for inertial sensors.

3.2 Activity Recognition Model and Processing

Activity recognition is defined in terms of task chains that contain data segmentation, feature extraction, training and testing of classification models. Multiple task chains can be handled concurrently on a single host. To best describe what each task is responsible for, a short introduction regarding system setup for activity recognition is following.

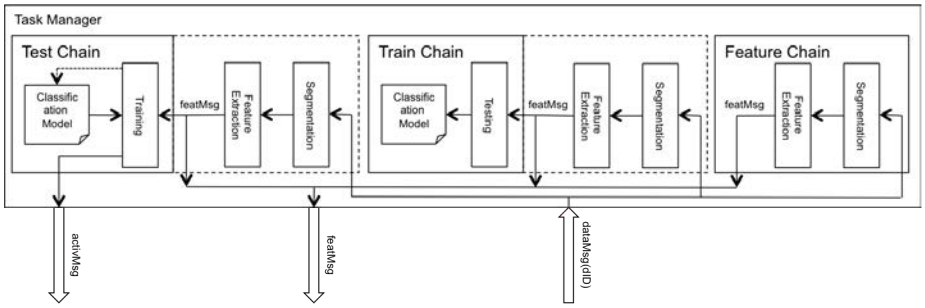


Figure 3.3: Activity Recognition Archtecture Diagram

A diagram regarding the different types of chains that are maintained in our system is depicted in Figure 3.3. There exist three different types of chains. Each chain consists of different subprocesses where each chain is called *task* in our framework. The different types of task chains implemented are:

- *Feature Chain*: The feature chain does not directly perform any activity recognition but provides the preprocessed data that is required for classification. It is responsible for segmentation (in form of summarizing data into windowed segments) and feature extraction where from the windowed data corresponding features are extracted and passed back to the system for persistence. It is also possible to forward feature data to a remote host and perform classification model training and testing on the remote receiver by application of the mentioned device roles.

- *Train Chain*: This task type is responsible for creating classification models based on the retrieved feature data. Details of this process are explained in the following Section 3.2.1. The feature chain that is plugged to the train chain in dashed lines denotes the standard feature chain mentioned under the previous bullet and shall only emphasize that this is obligatory necessary for the training process to work correctly. The same is the case for the test chain. As a result of this type of chain a corresponding classification model is emitted that is used in the next type of task chain.

- *Test Chain*: This task takes the classification model built up in the last task and applies any received feature data on this model. The return value of this task is a prediction based on the incoming feature data value and how the applied classification model classifies the data. This result is also directly passed back to the base system where it may directly be used to trigger any general state changes based on set triggers regarding the classified activity.

How the different types of chains are integrated into the runtime system is explained in Section 3.2.2.

3.2.1 Chain Tasks and Processes

The following paragraphs provide more detail regarding the functionality of the single processes required for the ARC. Some parts have not been depicted in the corresponding diagrams but are mentioned where they were relevant for the related implementation.

3.2.1.1 Data Recording

Data recording has been defined to be running outside the activity recognition specification. The purpose for this approach was to enable data collection and persistence even if the activity recognition methods have been disabled. Nonetheless, it is an essential part of the whole framework. Therefore it is mentioned here once again.

3.2.1.2 Preprocessing

Preprocessing is often concerned with applying signal filters to the raw sensor data to make it even post-processible by following stages. Without this stage data would often be completely useless to the following segmentation step, as it is unable to read the raw sensor hardware data format. Approaches like low pass filters suppress effects like signal noise (occuring from technical sensor properties) to deliver a smooth signal to the following stage.

In this work, preprocessing is handled in the data recording stage, and not depicted explicitly, as it is assumed, that for the rest of the processing stages, data is available in a format handable by the system.

3.2.1.3 Segmentation

Segmentation is the task of organizing preprocessed raw data into chunks of data that are also often known as windows. Over these windows the following feature calculation is processed. There exist multiple approaches to segmentation for sensor data like overlapping segments or adaptive window sizes. It is also possible to combine segmentation with preprocessing and apply filters on segments instead of raw data streams. This is dependent on the application case and no further considered here.

In this work, we use a non-overlapping, fixed window size of one second. With this method it is irrelevant how much samples in this one second appear as the resulting size of a window is always the same. Only forn second time ranges where no data was retrieved at all, missing data results.

3.2.1.4 Feature Extraction

This task performs feature extraction on the filtered and segmented data. It is kept separate to allow the extraction of features on a local platform and forward feature data to a remote host where the classification model construction and activity testing may be performed.

Multiple approaches to feature extraction exist. In this work the arithmetic mean and variance of the sensor data chunks are calculated for all data samples contained in a window, yielding one derived value for every of the three axes per window. The corresponding formulas are restated below:

Arithmetic mean $\bar{x} = \frac{x_1+x_2+...+x_n}{n}$ used for all samples in a window per unique sensor axis.

$$\text{Mean}(X) = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Variance used to observe variations in sensor signal regarding a particular activity. It is expressed according to every mean above.

$$\text{Var}(X) = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

In this work, often the standard deviation is used instead of the variance, which is the square root of the variance with a similar information content. They are used interchangably in this work.

3.2.1.5 Feature Filtering

When activity traces get filtered by their activity assignments or temporal properties alone, this can already be done in the data recording stage. For practical reasons this step is treated here, handling feature data only, as a significant data reduction occurs when the amount of extracted feature data is compared with the raw data size.

The idea is that classification results are not influenced by the numerical sensor data alone, but also by properties regarding when and how activity assignments have been performed. With the Label Visualization Tool, presented in the following Chapter 4, it can be made visible that data recordings apparently contain incorrect or unintended assignments. In Chapter 5 it is shown that dismissing these wrong assignments has positive effects for classification results.

3.2.1.6 Training

Training is the process of setting up a classification model from featured sensor data. For our purposes, well established algorithms and methods for data classification have been applied. The following list provides a brief overview of the different classification routines deployed and shortly summarizes their functionality. Some descriptions are partly overtaken from [44, 20, 7] and Wikipedia[1].

- *Decision Tree (DT)*: Depending on certain ranges of the values of observed features different sub-branches are created, after that new samples will be classified to belong to one class or another depending into which (sub-)branch it fits. For our sensor data samples this means, that depending on a certain range of values dependent on the activity class a new branch will be created. The leafs of the tree contain the final activity class decision dependent on the value ranges that direct the path through the branches of the tree to the corresponding leaf.

- *Naïve Bayes (NB)*: This type is a probabilistic classifier which (naively) assumes that features are statistically independent according to Bayes' rule. This implies that the features in our case would be independent, which is practically not true. When one accelerometer axis get changed, at least a second one will be changed too, which means that there is dependence given. Nonetheless, the independence assumption is applied with this classifier. Correspondingly, classification results are not that accurate when compared to other classifiers.

[1] http://en.wikipedia.org/wiki/Main_Page

- *Support Vector Machine (SVM)*: Supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

- *Neural Network (NN)*: A family of statistical learning models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which send messages to each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

- *k-Nearest Neighbor (k-NN)*: A non-parametric method used for classification. The output is a class membership.

  An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

  k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms. A shortcoming of the k-NN algorithm is that it is sensitive to the local structure of the data and that it is rather memory intense as all samples have to be saved.

The training stage does not produce any data to be passed to the framework, but creates a model structure that is used to classify new data samples. This model can be saved to persistent storage for reuse at a later time for testing, addressed in the following paragraph. The format and storage organization of model files is explained below, in Section 3.2.4.

3.2.1.7 Testing

Testing is the application of a model built in training with user specified test data. Cross validation is a special case and treated in the following paragraph. Generally, testing evaluates a built classification model with arbitrary testing data and outputs recognition rates based on the accuracy the observed classifier can reach given the model and the training data.

Algorithm 3.1 Pseudocode for N-Fold Cross Validation

```
n ... number of folds
divide data set into chunks of 1 ... n, of size 1/n each
for i = 1 to n

    testset = chunks[i]
    trainset = chunks - chunks[i]
    model = buildModel(trainset)
    recRate = evalTrainsetOnModelOn(trainset, model)
```

This is the process of evaluation and its particular application for this work is addressed in Chapter 5. An exponential number of possibilities exist of how training data can be evaluated against arbitrary test data. A sensible average has been pursued to meet on one side the practical requirements of this work and maximize expressiveness of obtained results.

3.2.1.8 N-Fold Cross Validation

This is a special case of the train and the test task by permuting train and test sets, building a model from each of the train sets and evalutaing the classifier on the trained data with the left back data chunk used as test set. The routine in pseudocode for n-fold cross validation is provided in Algorithm 3.1.

The overall recognition rate is then given by building the average of the recognition rate from each of the different folds. The classification results presented in Chapter 5 are extracted by applying this method.

3.2.2 Framework ARC Design

Following the conceptual representation regarding the ARC, depicted in Figure 3.3, additional implementation details of the ARC are explained in this section. For a great extent of flexibility the task manager creates and maintains a new chain task dependent on different sensors supported or methods of classification that shall be applied. This way an arbitrary number of chain tasks can be maintained at the same time with the processing of the task itself handled by the specific chain task.

Usually, the different types of tasks (segmentation, feature extraction, train or test task) implement a certain interface for easier integration within the rest of the framework, but it is possible to develop the core functionality of the task completely outside of framework interfaces and still interact with the framework by providing only the minimally necessary interface connection points. The task manager itself provides a simple multiplexer dependent on type of feature message and message source.

For every recognition chain that has to be instantiated a new chain task is created. Depending on the purpose the chain task has to serve, different process stages will get added to the task.

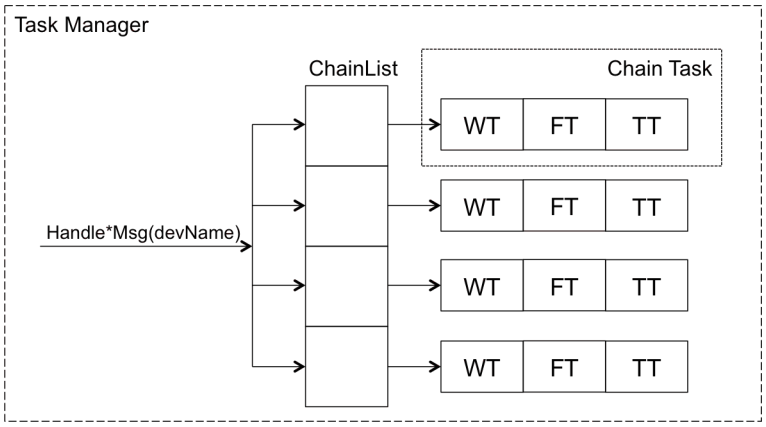


Figure 3.4: The ARC Implementation Design

For a single sensor only a single feature chain is sensible, especially on mobile platforms. Multiple feature chains with different parameters would significantly increase the demand for system resources. In the diagram above multiple feature chains are depicted. This is used to again emphasize that the system is generally capable of such a setup.

So, for a single sensor setup a single feature chain is instantiated which can be hooked into different train and test chains. Therefore, from each of the train or test task chains multiple instances with different configurations (different classifiers applied) can exist simultaneously. Therefore, multiple test scenarios can be carried out concurrently, which decreases computation times and increases device resource utilization.

More details regarding activity representation and classification model handling within the framework are presented in the following sections.

3.2.3 Activity Model Organization

To provide activity labeling to the user, it is first necessary to have a formal description of activities available, that can be utilized for augmenting data streams with the corresponding activities a user performs. For this purpose a formal activity model description has been designed, that utilizes the web ontology language (OWL)[2] format. This format was chosen as it is easily extensible and corresponding input and output mechanisms are existing and free to use.

In contrast to the model type described in the next section, the activity description model is called *formal model* in this work as it provides the conventional interpretation of activities used in the system.

A trimmed example of the activity description model is presented in Listing 3.4. Currently, it is merely an enlisting of different activities available in our framework. Extension in the direction of mapping general situation properties

[2]http://www.w3.org/TR/owl-features/ [retrieved: 05, 2015]

are possible, e.g. statistical items like overall mean or inter-ontological relations like probabilities for how likely it is that a certain situation follows a given one. Also reasonings over time within a single situation model as well as over sets of models is needed.

Listing 3.4: Runtime XML configuration

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [     <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
        <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" > ]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
        xml:base="adl_situation"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xml="http://www.w3.org/XML/1998/namespace"
        ontologyIRI="adl_situation"
        versionIRI="adl_situation">
        <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
        <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
        <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
        <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/
            >
        <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
        <Declaration>
                <Class IRI="#activity"/>
        </Declaration>
        <Declaration>
                <Class IRI="#break"/>
        </Declaration>
        <Declaration>
                <Class IRI="#car_driving"/>
        </Declaration>
        ...
        <Declaration>
                <Class IRI="#tv"/>
        </Declaration>
        <Declaration>
                <Class IRI="#work"/>
        </Declaration>
        <SubClassOf>
                <Class IRI="#break"/>
                <Class IRI="#activity"/>
        </SubClassOf>
        <SubClassOf>
                <Class IRI="#car_driving"/>
                <Class IRI="#activity"/>
        </SubClassOf>
        ...
        <SubClassOf>
                <Class IRI="#tv"/>
                <Class IRI="#activity"/>
        </SubClassOf>
        <SubClassOf>
                <Class IRI="#work"/>
                <Class IRI="#activity"/>
        </SubClassOf>
</Ontology>
```

3.2.4 Classification Model Organization

In contrast to the formal model above, in this part the corresponding *prediction model* is presented, that is the result of a train task and is used to generate predictions based on the classification model established for a certain set of data. In the diagram in Figure 3.3 it is depticted under the test chain task.

The results of a training task need to get stored to make them accessible to the corresponding testing task. For this purpose the Javascript Object Notation (JSON)[3] format has been utilized as it is a condensed and flexible format. Important properties with respect to the usage in this work are:

- A clear text model with an encoding based on regular expressions. This way prediction models can easily be exchanged for reuse on remote hosts (whether the local host forwards data and classification is performed remotely or the remote host performs classification on its own data but gets the prediction model from another host)
- Contains all the data and parameters for the trained classifier with respect to the chosen classification method (k-NN, DT, ...).
- A generic header contains characteristics that are applicable to all classification models the same way.

The properties that are unique to a chain task are saved with the filename of the corresponding model file. The corresponding file storage strategy is analogous to logging as specified in Section 3.1.6. This way different sensors and classification methods can be supported.

Listing 3.5: Example NN model JSON file

```
{
"featSig":["mean_x","mean_y","mean_z","sd_x","sd_y","sd_z"],
"activs":{"car_driving":466,"meet":106,"locomotion":576},
"hiddenLayers":"a",
"seed":0,
"momentum":0.0,
"autobuild":false,
"normalizeFeats":false,
"driftThreshold":0,
"numEpochs":0,
"reset":true,
"learnRate":0.0,
"useDefaultModel":false,
"decay":false,
"featRanges":[3.99,4.53,4.34,39.42,30.50,28.51],
"featBases":[0.95,-0.43,-1.22,39.42,30.50,28.51],
"inputs":[{"mId":"mean_x","mType":1,...},...],
"neuralNodes":[{"mType":24,"numOutputs":1,"outputList":["
    car_driving"],...},...],
"outputs":[{"mId":"car_driving","mType":18,"numInputs":1,"
    numOutputs":0,...},...]
}
```

[3] http://json.org/ [retrieved: 05, 2015]

The example in Listing 3.5 shows a classification model result file, in this case for the neural network classifier (NN). The first two fields are generic and apply on every type of prediction model regardless of the classifier instantiated. The "`featSig`" element describes the feature types that were in use when this model was created. Implicitly, the number of items contained in this element denote the dimensions of the feature vector.

The second element, "`activs`", references the different activities that are regarded in this model and how much samples per activity are concerned. These numbers are required for setting up weighting for certain models. It can also be used to backtrack how this model was specified.

The following elements in the JSON file are specific to the classifier that created this model file and contain different parameters for each classifier type. Usually, there is one item in the specific elements that contains classifier specific values for a particular feature or activity. In the example, inner details are omitted and depicted with ellipses.

3.3 Practical Considerations

After the computational concepts, this work is concerned with, have been explained, the runtime platform to perform data collection is discussed in this section. The choice of a wrist worn sensor platform has already been made, as it turns out the most practical approach regarding deployment and everyday use. To find out which particular product to employ, a comparison of available products has been conducted where different properties like local processing and storage capabilities or battery lifetime have been taken into account.

3.3.1 Choice of Runtime Platform

Before particular devices have been revised a set of requirements were established that describe the minimum prerequisites the platform has to be able to deliver. Factors that led to the final decision are:

- The availability of the system on the smart watch has to last for at least more than a common working day (> 8 hours) with continuous sensor operation and sporadic user display interaction.
- The ability to operate in a fully autonomous way without the need of permanent (or temporal) coupling with an accompanied master device (sattelite constellation with a smart phone - as is the usual operating mode with more recent devices).
- Sufficient local storage and processing power to execute the whole framework in a standalone fashion.

The particular products that were selected for comparison come from two different categories. First, smart watches are already clear as they indicated the most promising device types. Another category are arm bands. Although equipped

with lesser computational power, their advantage against smart watches is that they usually have longer battery lifetimes between recharging cycles.

Current market solutions also enforce devices to run in a sattelite mode only, coupled with a smartphone or tablet device. Longer, autonomous traces are not possible in a standalone mode without any paired master. This looks like a marketing trick to increase sales of smartphone products. For this work the difference would not have been relevant technically but a truly autonomous solution brought much more comfort in everyday use compared to situations where two devices have to be carried around and maintained constantly.

The table in Appendix A.2 provides an overview of revised smart watch devices. The first five devices are fully capable smart watches that possess computational power in the dimension of first generation smart phones. But due to this fact battery lifetimes of more than two days are not possible (while executing any special task). The last two devices are more accountable to the armband category, regarding their computational capabilities. They are still listed in the smart watch category, as they provide the smart watch like displays and interaction methods.

The second comparison treating armband devices is summarized in the table in Appendix A.1. Armband solutions come in two forms, whether as *commercial closed ecosystem solutions* that currently mainly serve the healthcare and fitness markets or as *open source solution*s that are often utilized in academia and research. The disadvantage of one kind is the advantage of the other and vice versa.

- Closed ecosystem solutions provide good runtime capabilities (regarding system stability and battery runtime) but have the disadvantage that data and processing stage descriptions are only very limited accessible (only cloud access).

- Open source solutions have the disadvantage that currently the system stability is not guaranteed and that battery runtime is often very poor. On the other side data and computational routines are freely available.

Due to the facts, that armbands have rather low local processing capabilities and are therefore bound to a satellite operation mode, this category was no further regarded for practical utilization.

Eventually, the final decision fell on the Motorola MotoACTV smart watch device, depicted in Figure .3.5.

Figure 3.5: Motorola MotoACTV Smart Watch Platform

Although it is not the most recent model available on the market, it combines all features required:

- More than eight hours of active operation with the required accelerometer sensor switched on all the time.
- No requirement by the underlying operating system of a constantly connected companion device.
- The support of WiFi, for backup data transport, which enables the autonomous runtime behavior.
- A local storage system of eight or sixteen gigabytes of data, that allows active operation for multiple days.

3.3.2 Long Term Data Sampling Considerations

In this work a way was explored to explain how practically an activity recognition system for real-world requirements can be established. Further, this work provides experimental evaluation strategies that are applied on the collected real-world data. Due to the extensive amount of available data, not all potentially possible methods have been applied on the data. Experimentally the most interesting strategies were practically evaluated.

Keywords like big data, quantified self or life long learning quickly appear in focus when being confronted with the amounts of data and information that are produced by such approaches. With this thesis, the general approach to such a system is shown. Furthermore, evaluation structures and methods are presented that show the general recognition ability. They can be taken as a starting point for more complex recognition methods that are beyond the scope of this work. Such aspects are finally discussed in the outlook section.

Chapter 4

Application Interfaces

After the basic framework for performing activity recognition has been discussed, the corresponding access interfaces need to be considered. They are required to conduct the corresponding real-world experiment and to perform analysis on the experimental data.

For this purpose two different types of application cases had to be considered:

i) To perform data labeling practically, a corresponding mobile application needed to be developed to provide the user interfaces for every day use.

ii) The application of corresponding desktop software is needed to perform a profound analysis of the data collected in the experimental phase.

After describing the application of the framework on the mobile smart watch platform, the evaluation tools utilized get discussed.

4.1 Mobile Application Interfaces

Interaction support on the smart watch is needed mainly for data labeling. For this purpose a smart watch interaction scheme has been developed to allow users to assign the performed activities over an interface available on the smart watch.

Objectives for user interaction that are considered with the practical application for data labeling:

- The user decides when to label activities and when not.
- For daily use, only main activity situations are assigned.
- Fine grained activity resolution is out of scope for the targeted experiment. Convenience in usage was more important than detailed resolution (which would be increasing interaction steps and more complex interaction handling).

The mobile part of the developed application can be operated on smartphones, tablet computers and smart watches alike. On the smart watch, other interaction methods apply mostly owed to the smaller display size and resolution.

- The display has a resolution of of 220x176 pixels (In contrast to current mid-level smartphones that are somewhere around 1080p Full HD resolution).
- The amount of content that can be shown in s single view is rather low.
- The number of subsequent interaction steps needs to be kept low, as longer interaction intervals are tiring the user as it has to use both hands (one that holds up the watch while the other is operating it).

4.1.1 Icon Based Activity Labeling

Regarding the activities denoted in Section 1.2.2, a corresponding interaction scheme has been designed that enables the user to augment its daily life with the activities it is performing.

Activities are represented by corresponding iconographs to provide a fast visual display for training and visualizing evaluation results. The activities potentially available for training and testing are presented in the following table.

It was already mentioned in Section 1.2.2 that fine granular activity tracking is not the focus of this work. Therefore a less granular approach was pursued, where the user only augments activity situations instead of any details regarding certain body movements or similar.

Before the interaction methods are discussed, the set of activity icons that were utilized, is presented. Regarding terminology specified initially, an activity icon is a visual representation of an activity item. The full set of possible activity icons is depicted in Figure 4.1.

	read		eat		meet		smoke
	break_smoke		education		dontcare		socialize
	car_driving		entertainment		phonecall		sports
	coffee		family_care		pray		travel
	computer		household		relax_think		tv
	cook		hygiene		shop		work
	drink		locomotion		sleep		write

Figure 4.1: Full activity set

Although, basically, it is possible to consider all above shown activities for recognition in a practical setting it has shown out that only a subset suffices to capture most activities performed over a day.

Therefore two different sets of activities can be chosen in the user interface, one time the full set which is rather inconvenient if activities lie separated far from each other and a reduced set with the most important activities that are relevant within the daily routine which are faster to reach (cf. Figure 4.2). The reason for this sub setting is that the display on the watch device is rather small why for a usable visual representation, only a single activity is shown at any one time with the possibility to swipe through the set in left or right direction.

If the full set is used many swipe gestures might be needed to switch from one activity to another which can be reduced by using a smaller number of activities. Especially activities that are required seldom are not included in the reduced set but can be requested by switching over to the full set.

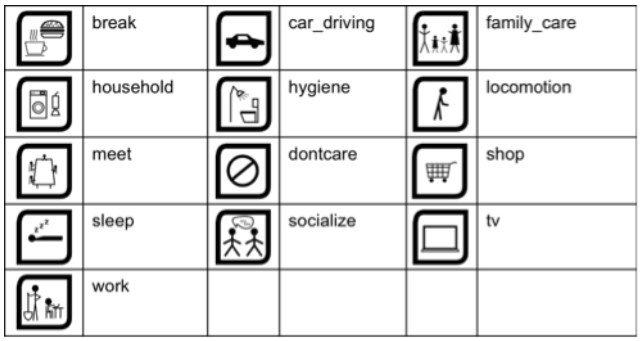


Figure 4.2: Reduced activity set

4.1.2 Data Labeling Interaction

When a new situation arises like the user starts watching a film on TV or starts preparing a meal (labeled as the activity situation cook) it chooses the activity from a swipe view and after the new activity situation choice has been selected this means for the system that the newly set activity is active. From then on, all received data samples will get assigned the newly set activity item.

The swipe view depicted in Figure 4.3 is used to select the currently performed activity on a smartphone. With a swipe gesture to the left or the right, the list of activities can be scrolled through. This list is realized as an infinite scroll list, so when the end of the list is reached it starts again from the beginning.

Pressing an activity icon selects the corresponding activity item. Then the view is switched back to the development view presented in Section 4.1.3.

Figure 4.3: Smartphone activity swipe view

The same swiper is represented on the smart watch like depicted in Figure 4.4. Comparing it with the image above, the motivation behind the specification of a reduced activity set is obvious.

Figure 4.4: Smart watch activity swipe view

4.1.3 Development Views

During the implementation of the framework a view capable of different runtime modes has been developed, that is not directly suitable or interesting for end user application but that contains valuable, debugging relevant information. This view can be utilized on smart watch and smartphone devices alike, as explained in the following paragraphs.

4.1.3.1 Smartphone Development View

A choice of different views is shown in Figure 4.5. They were specified to revise the online functionality of the train and test ARC stages. Activity labeling is handled as a parallel sub-process.

Figure 4.5 (a) shows the user interface (UI) in the initial state, when no recording process have been started. At the top of the view buttons to start and stop the process are added (This is not the case for the experiment version of the tool as it is running all the time). In the center the currently selected activity item is depicted. The icon is added to a button, that directs to a swipe view for selecting the current activity (cf. Section 4.1.2 above). The rest of the UI contains runtime mode selectors (train or test mode) and runtime information (that is initially empty).

In the image in Figure 4.5 (b) the runtime system under operation of the train task is shown. It can be seen, that options that can only be selected

during an idle system mode are disabled. Only the button to end the process, the activity item selector button and the activity set selector can be handled. Below the activity item selector button, runtime information like the number of collected samples is shown.

Figure 4.5 (c) shows the system when executed in test mode. In the image two different recogntion chains have been installed. Each line represents first the detected activity, given the stored formal model and the selected classifier and then shows the recognition rate achieved with this model.

(a) (b) (c)

Figure 4.5: Development UI on a smartphone

4.1.3.2 Smart Watch Development View

The same development view, as shown above, was also operated on the smart watch. In vertical direction more information needs to be displayed than fits on a single view of the watch. Therefore, the standard development view was integrated in a scroll view to reach all content. This is shown in Figure 4.6, below.

These displays resemble the same views as Figure 4.5, this time on the utilized smart watch device. By the implementation within a scroll view, all content can be reached. The watch display has to be swiped in the vertical direction and the figures are overlays of multiple screenshots to put every operation mode into a single view.

(a) (b) (c)

Figure 4.6: Development UI on a smart watch

For reference, the image sequence in Figure 4.7 shows the sequence of steps that are necessary to switch the activity in the corresponding development view. In this example the switch from the "work" activity to the "break" activity is depicted. Only the topmost part of the development views on the smart watch are shown.

Figure 4.7: Example of activity item switch

4.1.4 Application Views

It has already been mentioned above, that the mobile views presented so far contain elements that are suitable during development, but too complex and not of interest for an end user. Therefore, an alternative UI setup has been implemented that further reduces interaction steps and simplifies the UI by presenting only view elements that are of interest to an end user.

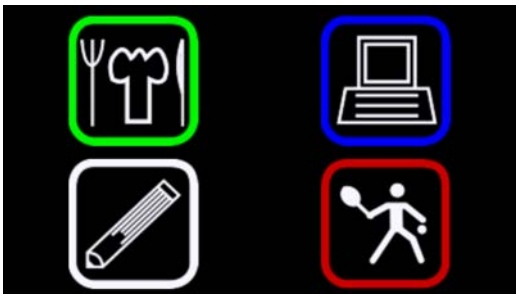


Figure 4.8: End user application UI

An example is shown in Figure 4.8. This is a modified version of the swipe view presented in Section 4.1.2, above. For this setup, the swipe view is used exclusively without presenting any details about how many data samples were collected and similar. A color coding is used to represent different states for an activity. The view depicted in the figure does not resemble a correct application run but is used to explain the different color codes.

Initially, when nothing is selected, the activity icon is shown with a white frame. A green frame denotes the user selected activity. With a blue frame a correct prediction is signaled, with a red frame an incorrect prediction is indicated.

Some minor points to be noted are that in the example images above, two versions of icons are used with different visuals for activity item frames. This different views emerged during the development of the visuals and does not have any technical consequences. They can be applied interchangably.

The other issue is that in some cases we have black activity icons on white background and in other cases the color reversed version with white activity icons on a black background. This represents a daylight and night version of the view and this property can be set over the device configuration, discussed in Section 3.1.2.

4.2 ARC Tools Implementation Overview

In addition to the mobile application views, addressed above, a set of tools have been implemented to provide extended modification and analysis features that cannot be handled this way on mobile platforms.

In this part of the work, mainly the elements of the corresponding views are mentioned. Details regarding the technical internals of each tool are presented in the following Chapter 5.

The different views that are in use are depicted in Figure 4.9. Within a register view (on the top below the main application menu) the following elements can be chosen by the user:

- The RecordVisu view, depicted in Figure 4.9 a), enables an online view of incoming sensor data in raw or featured fashion.

- The LabelVisu view is used to provide offline analysis and filtering prior to classification. It is used as a method to research how input data modifications influence classification results. It is depicted in Figure 4.9 b).
- The EvalVisu view represents the custom classification development, reusing lots of code from open source tool set libraries and offering additions regarding the application cases that are targeted by this work. An example is shown in Figure 4.9 c) and corresponding details are explained in Chapter 5.
- As a reference classification tool, Weka[1] was used. Its primary classification and analysis tool has simply been integrated into the framework of this work, by instantiating the corresponding view object contained in the Java archive (JAR) library provided by Weka. In Weka, this tool is called "Explorer" and this was added to the tool in the WekaVisu tab, shown in Figure 4.9d)

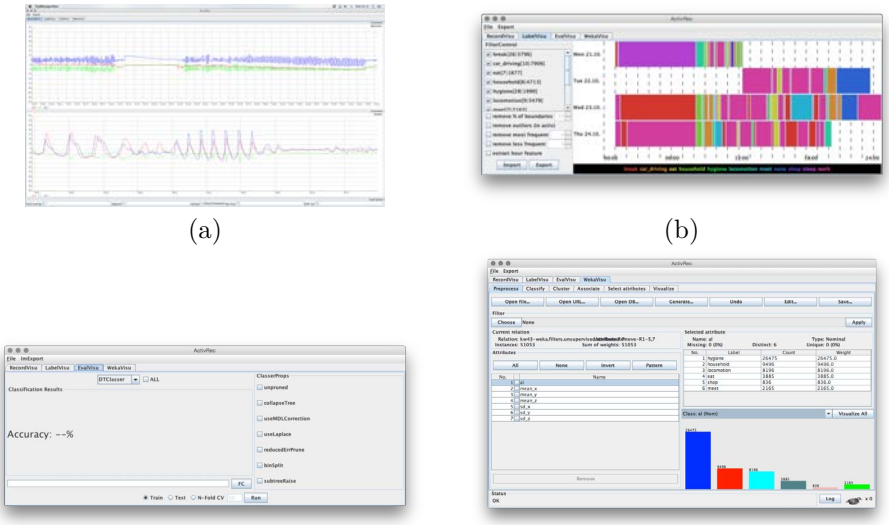


Figure 4.9: Different view tabs for different ARC stage tools

Additionally to the presented toolset, other common purpose tools like R[2], Matlab[3] or different script programming languages (like Bash[4] or Python[5]) have been utilized. Their application is mainly related to the evaluation chapter of this work.

[1] http://www.cs.waikato.ac.nz/ml/weka/ [retrieved: 05, 2015]
[2] http://www.r-project.org/ [retrieved: 05, 2015]
[3] http://de.mathworks.com/products/matlab/ [retrieved: 05, 2015]
[4] http://www.gnu.org/software/bash/ [retrieved: 05, 2015]
[5] https://www.python.org/ [retrieved: 05, 2015]

4.2.1 Desktop based data recording and feature extraction

As a simple tool for revision and testing of data collection a desktop based recorder has been implemented, that utilizes the same core runtime features as the mobile recording application. The corresponding visual is shown in Figure 4.10. Incoming sensor data streams are shown as online trace charts that are updated over time as new sensor data is retrieved.

On the left side the online trace for the raw data of the three different accelerometer sensor axes are shown. On the right side, the correponding features extracted from the raw data samples are rendered as online traces.

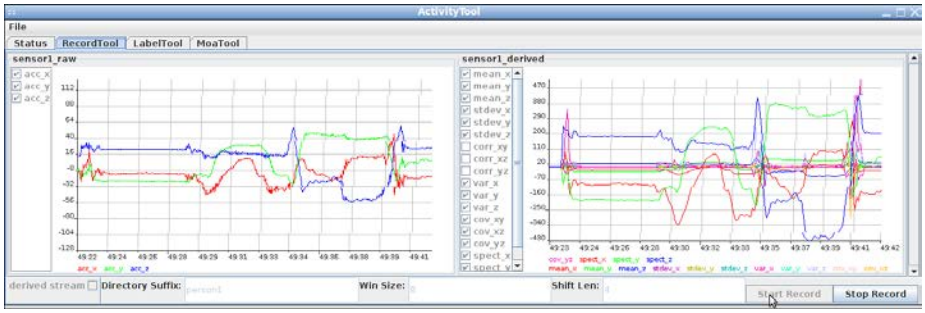


Figure 4.10: Desktop Data Recording View

An additional option is to select the raw data streams and features that get considered. This is contained on the left of every chart.

In the bottom line the window parameters “window size” and “shift length” can be adjusted. Over the window size the numbers of samples that are considered for a window, will be set. The parameter shift length determines how many sample instances the window is moved before a new window segment will get allocated. This way overlapping windows (each window containing information from the previous one) or even windows disjunct over time can be made up.

The data streams and features that will be recorded can be selected while the tool is not recording. During runtime these properties can not be changed.

4.2.2 Activity Trace Visualization - LabelVisu

Parts of this view have already been used a few times to show certain aspects of this work, the visualization for activity traces has not been fully specified up until now.

The image in Figure 4.11 shows an example trace visualization for one week. The main part of the view is the canvas where the activity traces are drawn. One row in this trace view denotes a day in the life of a participant and how this day trace has been labeld. The exact day can be retrieved from the text on the left of the activity traces. At the bottom the color codings for the activity labels are contained. The activity shown with a particular color can be found in

the trace view above. To determine at which times an activity was performed indicators showing the time of day were added.

By convention, a time frame of one week was chosen as the default unit for visualization. This is a compromise to obtain still feasibly handable amounts of data for visualization and computation. A representation that contains a longer time period in a single view can be found in the evaluation chapter. In the following paragraphs the elements and operation of this tool are explained.

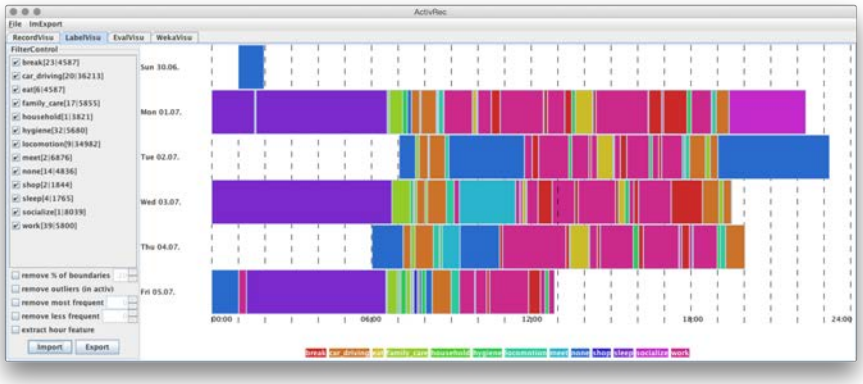


Figure 4.11: Activity Visualization Default View

On the left side there two fields with several selectable options (Filter Control) and two buttons at the bottom that trigger the execution of the tool in one of two modes.

The upper part of the Filter Control provides options based on the activity labels available in the loaded dataset. Summary statistics in the form of how often the activity appears in the data set and how much data samples contribute to the activity are included in the label. By deselecting a checkbox beside the activity label, the activity would get removed completely from the data set. This is also represented visually.

The lower part of the FilerControl contains general filters that apply to the data set as a whole. The different options are:

- Remove % of boundaries: During the data collection experiments it has often been seen that activities have been switched too soon or too late by the users. Therefore a certain amount of data samples at the beginning and the end of each single activity item can be removed by this filter.

- Remove outliers: For each type of activity it might be the case that single activity items last too long or too short compared with the average duration of the same activity type within the loaded data set. This might occur when the user forgot to switch the activity over a long time (while performing some completely different activities) or sensor malfunctions.

- Remove most frequent: It also appears throughout all collected data sets that some activity items have much more data samples compared to the average sample size over all activities of one type. This might influence classification in the way that too much samples are classified as the "oversampled" activity although they belong to another activity class. To remove such "oversampled" activities this filter can be applied (for all activities of a certain type).

- Remove less frequent: This filter works the same way as the previous filter, this time only for activity items that have an uncommon low number of data samples contained (e.g., activities that were selected accidentally and never performed in the real world).

Over the "Import" Button a file chooser can be reached that provides the choice of which data set to load. The "Export" button on the other side exports the loaded data set in a single run. Depending on filters that are selected, a filtered version of the data set gets exported.

One aspect that needs to be noted at this place is that this view does not contain any indications of the numerical accelerometer sensor data. It is concerned with the time based properties of the labeled activities, exclusively. But of course, this data is contained in the exported data files (as also indicated in Section 3.1.5). How the different filters are applied on the data and which implications come up for classification, are described in the following Chapter 5.

4.2.3 Classification Application

The last type of desktop tool realized in this work is the evaluation visualization depicted in Figure 4.12. It represents the access interface to a set of customized classifiers. The classifiers are the same as utilized in standard classification tools but with some extensions to improve the runtime behavior of classification. This way concurrent processing or simultaneous classifier chain handling can be achieved, which is not available in the standard classification tools versions.

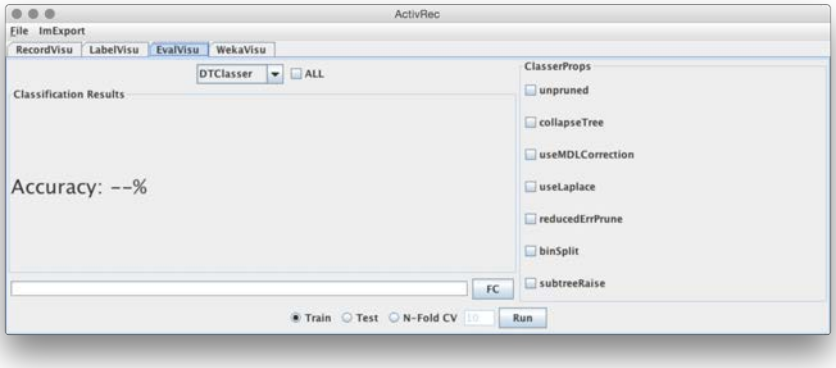


Figure 4.12: Desktop Classification Tool View

In the top part of the view the type of classifier that shall get evaluated, can be set. On the right side (ClasserProps) specific properties relevant for the selected classifier can be set.

At the bottom of the view the runtime mode can be set. The different modes are explained in the evaluation, Chapter 5. Above the mode setting controls the data set on which the classification evaluation will get applied on can be chosen, and above this selector, classification results are printed after an evaluation procedure is finished.

As a reference the Weka tool has been added to compare our customized classifiers with the results of a standard classification implementation. An example of the data selection part of this tool is depicted in Figure 4.13.

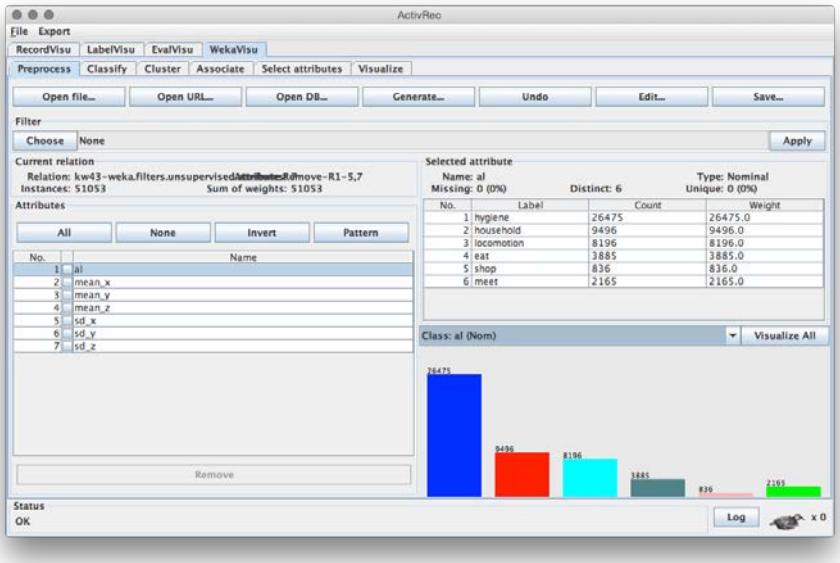


Figure 4.13: Desktop Weka View

It can be seen, that standard filter mechanisms are included, like removing certain types of data values or calculating normalizations for specific attribute types. The time based properties of our activity traces require other types of filters, that are not available in the general purpose tool. They are handled in the label visualization tool, where they are developed to meet the requirements of our specific data types.

Chapter 5

Experimental Evaluation

After the architecture of the proposed system and application access have been presented, the real world application of the system and the evaluation of collected data are addressed in this part of the work.

An extensive real world experiment has been conducted where three participants utilized the mobile application for periods of several weeks. They augmented their data with the activities they performed with the interaction methods presented in the last chapter. The resulting data set has been evaluated with different methods and initial evaluation optimization strategies are presented in the last part of this chapter.

In detail, the following stages are discussed in this chapter:

- Extensive data acquisition phase that contains several hundred megabytes of user labeled accelerometer data.
- Visual representation and inspection of the data with different visualization techniques.
- Classification of data with standard machine learning and pattern recognition tools. Visualisation of recognition results in accordance with standard evaluation metrics utilized for this class of problems.
- Optimization strategies by application of presented filters and evaluation of benefit for accuracy enhancement.

After an overview of the acquired data is provided, the application of analyisis methods, corresponding results and optimization approaches are presented.

5.1 User Experiment

The presented mobile application (cf. Section 4) has been applied under real world settings by three different participants. They were asked to wear the smart watch at least some hours of a day and assign labels to the activity they are performing at the corresponding instance in time.

To make the application convenient to use, there was no strict plan of when which activity had to be labled in a certain manner. Rather, the intention was to let users develop certain habits at how they use the application so that behaviors of activity label assignments and also errors start to follow some certain patterns that can be detected and removed to increase overall performance. The following enlisting gives an overview of directives that should have been broadly followed in conducting the user experiment.

- It was up to the user at which granularity activities were assigned. A fine granular assignment would not be possible by design, but this was not the purpose of the work and is an approach that is mainly followed in related work. In this document mainly long term aspects are pursued, that can only be recognized when sufficiently large data sets are available.

- For situations, where it is not completely clear as to which activity to set (in situations where multiple activity situations apply, e.g. doing computer work while watching TV) the more important activity can be set for which the recorded trace represents the corresponding activity.

  On the other hand, when the user is in a situation where it is inconvenient to achieve multiple interactions with the system to set new activities (e.g. when doing hard manual work) a default "don't care" activity can be set which can later be discarded when the corresponding model is established. Therefore the system will not limit or interrupt with the normal behavior of the user, while retaining its full capabilities. The only case that can happen is that models need longer to reach a certain quality.

From the following descriptions it can be seen that different users apply the mobile application in various ways, which makes inter-person recognition somewhat harder. The main purpose was to increase recognition rates for each single user by observing common behaviors of each user.

In the rest of this chapter, only feature data is considered that was extracted as explained in Chapter 3. Raw data format handling is not further considered in this work. In the following paragraphs an overview of the data collected in the experiment is presented.

5.1.1 Data Samples Overview

To provide an overview of collected data the same approach has been followed that was already used for storage of observed data. In this way it is easy to divide recordings regarding the timeline they were observed in and by activity class type.

Table 5.1 shows the total amount of data that has been recorded per user. The data size in an uncompressed format on hard disk is then 1,200 megabytes for participant 1, 221 megabytes for participant 2 and 320 megabytes for participant 3. This results in more than 1.7 gigabytes of data for all 3 participants. Data has been recorded over a time range of 23 weeks by participant 1, 8 weeks

by participant 2 and 18 weeks by participant 3. The total number of samples and individual activity item labels reflect the storage requirements.

Participant	# of activity items	# of samples	# of weeks	MB collected
P1	2582	6,125,433	23	1,200
P2	198	858,306	8	221
P3	405	1,208,746	18	320

Table 5.1: Participant data samples key characteristics

This general overview is elaborated in more detail in the next Section 5.1.2, where an arrangement according to observation period, activity type and participant are established and discussed.

5.1.2 Summary Statistics

To retrieve a condensed overview of the available data set, summary statistics are a first approach. They are used to provide key characteristics related to activities like duration or average sample size per activity appearance. Also the arithmetic mean of the series of means and its corresponding variance from a sensors signals can provide indications how classification may behave. In this work we call the first type *activity based* summary statistics, and the second one are called *signal based* summary statistics.

Activity based summary statistics are concerned with expressing time based and sample size characteristics regarding the assigned activity labels. When activity labeling is seen as a permanent process continuous over time, then activity label assignments can be represented for the time they appear in. This is done with the label visualization tool. The different activity labels can then be summarized by their type, their time of appearance and their duration. This can be used for feature filtering, shown below in Section 5.4.1.

During data analysis experiments it have been discovered that for the approach chosen in this work, a reasonable unit of time is one week. This has to do with the fact that some activities are always only conducted once a day, so there will be no comparison with the same activity within a single day. With a data set for one week classification routines behave in a resource sensible manner. They are in the range of some tenths of megabytes. Bigger data sets will need more storage and the classification routines will need longer runtimes.

Signal based statistics are best observed over different approaches to visual evaluation. This is addressed in Section 5.2. Regarding activity based statistics, an overview is presented in this section, more details can be found in the corresponding appendix sections.

In Table 5.2 an overview according to activity type, sample size and participant is provided. It shows the number of samples per activity for each participant over the full recording time range. Also the number of activity items is included, which means how often the specific activity item was set throughout the whole dataset. This is a frequency related key characteristic.

The third column for each participant is the average sample rate per activity. It is simply the total number of samples divided by the number of activity items (the numbers of the second column divided by the numbers of the first column). The average sample size provides indication about the duration of a certain activity for a participant. This enlisting can also be achieved for the activity items duration explicitly, yielding the corresponding time related key characteristic. This is not done in this work, as both key characteristics show the same behavior.

Participant	**P1**			**P2**			**P3**		
Activity	#AIs	#Samps	Avg	#AIs	#Samps	Avg	#AIs	#Samps	Avg
break	367	447766	1220				7	2712	387
break_coffee				2	3373	1686			
car_driving	266	1838375	6911	49	322737	6586	50	54914	1098
coffee				2	4488	2244			
computer				25	92225	3689			
cook				12	71390	5949			
eat	139	142907	1028	20	48358	2417	3	16418	5472
education	1	417	417	1	53	53			
family_care	178	361803	2032	5	17657	3531			
household	74	190011	2567	15	77556	5170			
hygiene	472	870673	1844	7	28027	4003	7	13805	1972
locomotion	132	388910	2946	14	55814	3986	156	587583	3766
meet	44	73134	1662	6	10768	1794	17	21284	1252
none	169	387691	2294	2	8	4	3	402	134
pray				5	15628	3125			
read	1	865	865	1	463	463			
relax_think				2	930	465			
shop	63	113680	1804	4	18984	4746	3	2287	762
sleep	49	63192	1289				15	4563	304
socialize	33	75541	2289	11	24896	2263	14	69305	4950
sports				1	7171	7171			
travel	3	3466	1155						
tv	13	27004	2077				5	1026	205
work	578	1139998	1972	14	57780	4127	125	434447	3475
Total	**2582**	**6125433**		**198**	**858306**		**405**	**1208746**	

Table 5.2: Total feature activity item and sample size for 3 participants

The complete overview, where the number of recorded data samples aligned with the week when they have been collected is contained in Appendix A.3. Again, the number of activity items is included in this overview, too (within the brackets next to the feature sample size value). This overview is of interest because it can be used to explain certain classification results. This is addressed below, in Section 5.3.2.

Some initial interpretation of the data can already be drawn from the the sample sizes alone as they are rather obvious and not directly related to the computational framework responsible for the technical management of the data.

- It can be seen that for some users some activities are rather important while for others they are not selected at all.
- Not selecting an activity does not mean that a user is not performing the corresponding activity in real life but has reasons not to label this accordingly:
  - personal: just don't want to, no time, ...
  - technical: recharging smart watch battery

  Also, the "none" activity does not mean, that the user is doing nothing at this time, but just out of some reason was not able to assign a sensible label at the time of recording. This label was introduced to keep the sensor data for alternative analysis approaches.
- It seems that some activities were selected accidently in the mobile application like the "none" activity for participant 2 that contains only 8 samples. This is clearly to be dismissed as is also the case for example for the activity "education" as it has a low number of samples compared to the total sample sizes of other activities.
- The total sample sizes per week (shown in Appendix A.3) give a rate of how detailed labeling has been performed. The higher the total number of samples, the more labeling has been done for the corresponding activity. The lower total numbers at the beginning and end of the recording period indicate that first a process of settlement had happened until participants were familiar with utilizing the mobile application and in the end got somewhat tired of doing more labeling.

5.1.3 User Data Comparison

An example where all the data of all participants was visualized in one view is shown below in the following figures. For this visual comparison, the LabelVisu tool is reused to show all data for a participant in a single view. It can be seen, that the tool would need adaption for visualization. If even bigger datasets will get displayed in one view, additional measures need to be taken for clear visualization.

Also classification and other post-computation stages will need advanced methods to process such data sets sensibly. This goes into the direction of handling of big data in the context of the quantified self. This is outside the scope of this work and a few additional remarks are provided in the future work chapter.

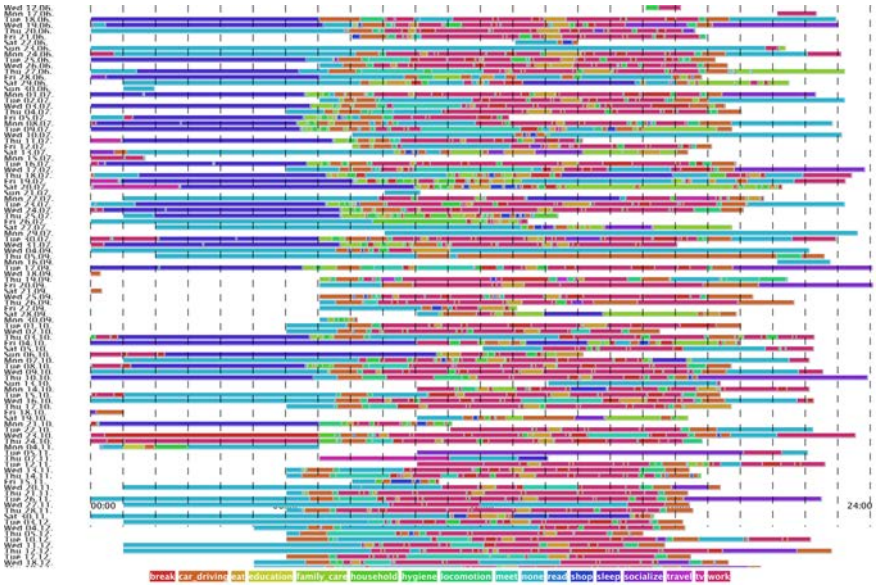


Figure 5.1: Total trace view for participant 1

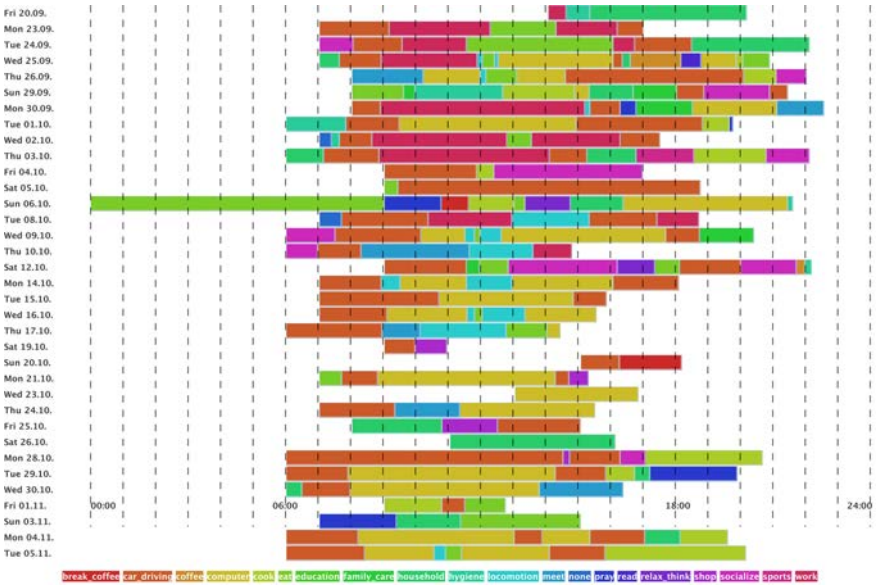


Figure 5.2: Total trace view for participant 2

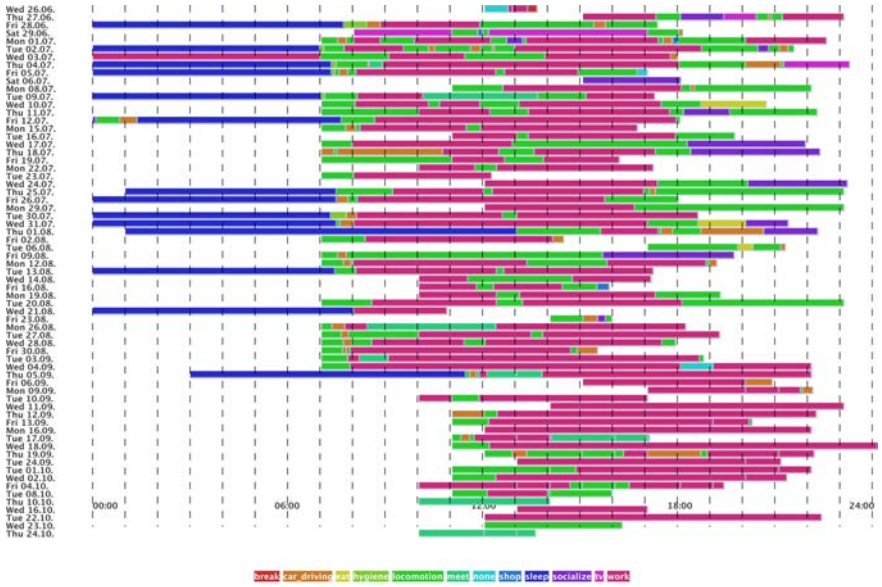


Figure 5.3: Total trace view for participant 3

What should be indicated with these visualizations is that some properties can be directly derived from the graphical representation. So does, for example, participant 2 never record any data during night (e.g., sleep; there is one outlier "education" where strong indications exist, that this was incorrectly labeled), which can be seen from Figure 5.2. Figure 5.3 on the other side does show, that participant 3 took all day recordings in roughly the first half of the period and later switched more to the second half of the day.

5.2 Visual Evaluation

Visual evaluation can be done in multiple ways for the given data. We already met one option in the form of the time based label representation of the LabelVisu tool. The LabelVisu tool will be of further support when performing feature filtering, treated below in Section 5.4.1.

But other possibilities exist to visualize the data. Especially signal based visualization can further be used to make phenomena visible, undetected else.

5.2.1 Box Plot

For the visual representation of feature data, two different types of representations are considered. In this part, box plots are revised, which provide summary statistics over all feature samples, and an example is shown in Figure 5.4.

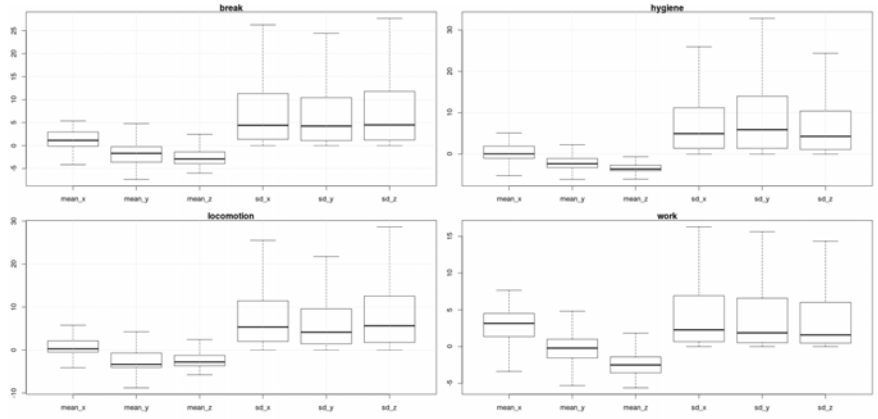


Figure 5.4: Boxplots for four activities for participant 1

By looking at this figure it can be discovered that the various boxes for the mean values show more difference than the boxes for the standard deviations (especially from the absolute median values).

Therefore it is expected that classifying over the mean feature values would provide a higher accuracy than classifying over the standard deviation features. After inspecting another plot type for visual evaluation, the validation of this assumption is done in Section 5.4.1.

5.2.2 Scatter Plot

The scatter plot is another tool for data visualization that is commonly used prior to classification to visually estimate how good signal features demarcate against each other.

With a scatter plot all features are compared pairwise and their values plotted on a two dimensional canvas. In Weka, a view matrix is created where every pair of features can be selected to present a more detailed view. An example of such a detail view is depicted in Figure 5.5, where the mean features of the x- and the y-axis are shown. Similar to the LabelVisu tool, the different activity types are assigned a different color.

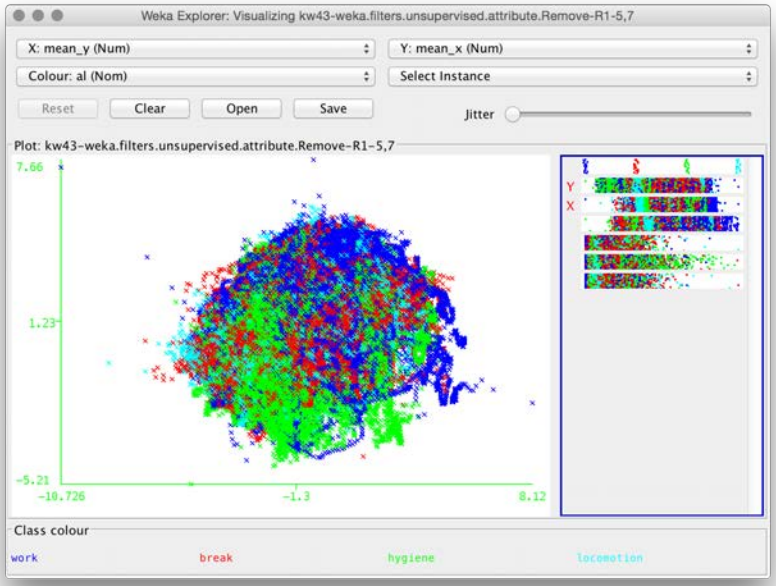
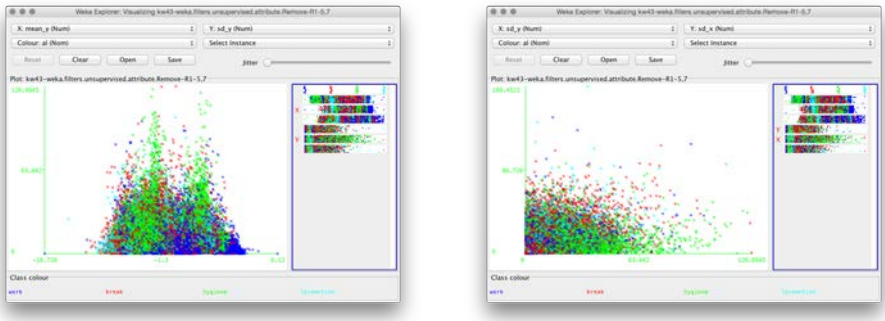


Figure 5.5: Scatter plot for the x and y values of the feature mean

From this plot is could be seen that on one side clear regions for certain activities exist but on the other side, there are quite a lot of feature samples that are wildly spread over the whole area, especially overlapping feature spaces.

The scatter plot for the same data, this time comparing the mean in y-axis feature and the standard deviation in y-axis feature, is depicted in Figure 5.6(a). It can be observed that feature data masses together even more which still increases when combining the standard deviation x- and y-axis features (shown in Figure 5.6(b)).



(a) (b)

Figure 5.6: Scatter plot for (a) the mean y and standard deviation y features and (b) the standard deviation x and y features

From the above diagrams it could be expected, that classification accuracy could be increased by reducing the data set to utilize the mean features only. Corresponding classification results are presented in Section 5.4.1.

5.3 Classifier Evaluation

In this section the general evaluation approach to classification algorithms is presented. Furthermore, the results of classification when applied to our data are presented. When talking about classification the class that is searched is the same as the acitivity type.

5.3.1 Confusion Matrix and Heatmap Representation

The standard method to quickly show the accuracy of a classifier for a multi-class problem is by its confusion matrix. After specifying the formalisms important for a confusion matrix, a heatmap visualization is presented that allows a quick overview of a classifiers accuracy in terms of rather good and not so good.

The confusion matrix is a standard method to compare the recognition quality for a certain activity class in relation to the other activity classes contained in the dataset. The following definitions apply:

- tp . . . true positives: They are the values in the main diagonal, meaning that the feature data sample has been classified with the same activity label as it has been labeled by the user during the data collection process.
- tn . . . true negatives: This characteristic is mainly useful in two class problems, where it is explicitly stated that if an activity is not of one class then it is of the other and this is classified correctly. This does not apply to the approach taken here, as we have a multi-class setup which for a specific activity would render all other correctly classified samples of another

activity type into the true negatives assignment. In our case the true positives and true negatives are represented by a single value (meaning that the tn value in the formulas can whether be replaced by the tp value or dismissed at all).

- $fp \ldots$ false positives: Looking at the column an activity is in, all entries not belonging to the activity represent samples that have been classified to these other activities although they should have been classified to the one activity in question.
- $fn \ldots$ false negatives: This is the inverse reading of the false positives. For the row of an activity it shows all other activities that have incorrectly been classified to be of the activity type considered.

With a corresponding ordering of the columns and rows, the confusion matrix will take following form. Then all false positives can be read from the upper triangular matrix and the false negatives can be taken from the lower triangular matrix.

TRUE POSITIVE	FALSE POSITIVE
FALSE NEGATIVE	TRUE NEGATIVE

$$\begin{aligned} \text{Precision} &= \frac{tp}{tp+fp} \\ \text{Recall} &= \frac{tp}{tp+fn} \\ \text{True negative rate} &= \frac{tn}{tn+fp} \\ \text{Accuracy} &= \frac{tp+tn}{tp+tn+fp+fn} \end{aligned}$$

The precision of an activity type is then the fraction of correctly classified samples in relation to the correctly classified samples plus the ones that have incorrectly been classified as other activities. The recall on the other side is the fraction of correctly classified samples in relation to the correctly classified samples plus the ones that have incorrectly been classified as the activity under consideration. So the precision is a measure of how good an activity type can be recognized among the others, whereas the recall tells how much an activity type is responsible that others are classified incorrect.

5.3.1.1 Heatmap Representation

A valuable visualization of the numeric confusion matrix is represented by a heatmap. By putting confusion matrix values into a color scheme where a bright color represents a high value in the corresponding cell, and a dark color shows a low value, the accuracy of a classification model can be observed visually. Corresponding examples are shown in Figure 5.7.

By default, the cells of a confusion matrix of a classification model contain absolute sample values. In this work, we used a relative weighting, where every

cell value is divided by the total number of samples in the whole matrix. This is done because under certain circumstances (e.g., very low accuracy increase or decrease) a change is hard to see visually, but can be detected by observing the numbers. Another implication of this relative weighting is that by summing up the rows or the columns the precision (row) and the recall (column) can directly be infered.

General properties of heatmaps, that later will be shown in application, are:

- The color coding scheme is usually arbitrary, for this work a dark brown color for low correspondences and a bright red color for high correspondences were chosen. Inbetween orange is faded between the two main colors.

- An optimal classifier (accuracy of 100%) would have all the highest bright colors only in the elements of the main diagonal and the lowest dark colors in the upper and lower triangular matrix. This can only happen if all activities are detected correctly and the sample size is equal for all activity classes.

- In our practical examples, where we have an inequal number of samples per activity class, even for good performing classifiers we get a view similar to that shown in Figure 5.7(b). The diagonal cell for hygiene which is darker than the diagonal cells for the other activities does not mean, that this activity is badly classified (which it isn't when compared to the other values in the column), but that the feature sample number is quite lower compared to the other activities. Usually, this is unproblematic, but a fact one needs to be aware of, when interpreting the heatmap representation. The classification model behind this heatmap has an accuracy of more than 90%.

- Aberrations from this scheme (which are natural as a 100% accurate classifier normally indicates overfitting effects) quickly show graphically which combinations of activity lables lead to the most misclassifications in the investigated example.

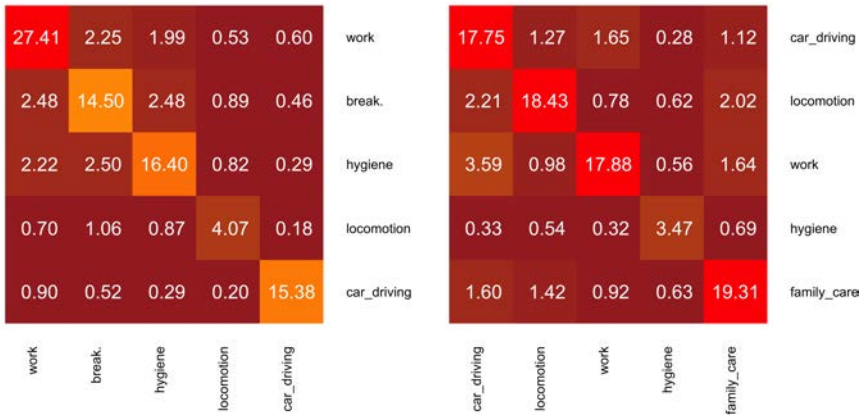


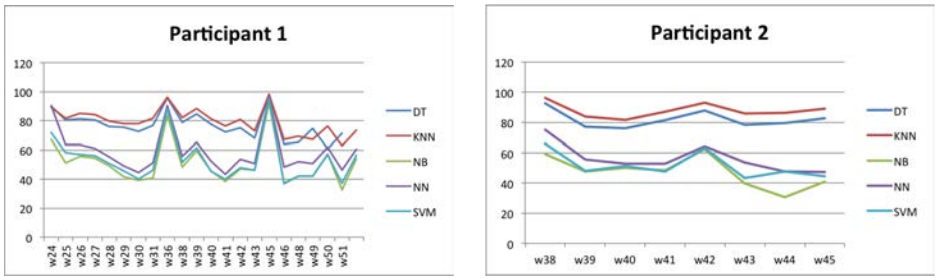
Figure 5.7: Confusion Matrix Heatmap Example

The heatmap examples presented in this work already make use of the activity type filter to reach better visual representations in print. When providing heatmaps for the full data samples more detail can be observed. For the presentational purpose in this work smaller sets of activity types suffice as they are easier to read and the common approach stays the same.

This section showed the general setup of the heatmap representation of a confusion matrix. How it is used for evaluation is shown in the following section when applied for intra-person and inter-person classification and observation of classification improvements.

5.3.2 Initial Evaluation Results

In this part of the work, evaluation results on unfiltered data are presented. Standard 10-fold cross validation was used for the classifiers discussed in Section 3.2.1. The results for the three experiment participants are depcited in Figure 5.8 and Figure 5.9. It can be seen that in all cases the k-NN classifier is performing the best, closely followed by the DT classifier. Then come the NN, SVM and NB classifiers, respectively. All of the last three roughly performing in this order.



(a) (b)

Figure 5.8: Cross validation results for participant 1 and 2

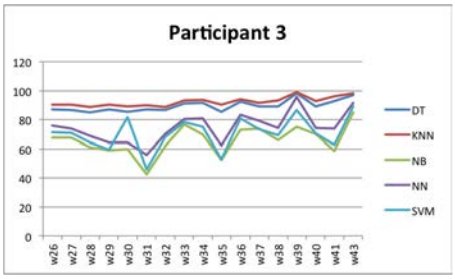


Figure 5.9: Cross validation results for participant 3

Cross validation results have to be seen differentiated because they are also dependent on the number of samples per activity. If in the set for a certain week the sample size of a single activity is of magnitudes much higher than the sample sizes of the other activities, it is obvious that the model is automatically weighted towards the activity with the higher sample size.

This is also the reason for the two outliers for participant 1 at week 36 and week 45. In case of week 36 the activity "car driving" has been recorded much more often than all other activities, in week 45 it is the same case for the activity "tv".

5.3.2.1 Intra-Person Recognition

This type of recognition is used, when the model trained for one period of time gets applied on the data set of another period of time from the same participant. This means to extract the recognition rate when the model and the training data do not originate from the same data set. In our case a week has been chosen as the time period.

A precondition that must be met is that the different activities in the trained model have to be the same as the activities in the test data. For this purpose the overview in the tables in Appendix A.3 shows which activity types have been labeled in one week and which not (including their sample sizes and number of

activity items). Generally, all sets can be used, but activities contained only in one of the two required activity sets need to be filtered out before the recognition process is started.

The properties by which activity sets are chosen are:

- All activities in a set should have roughly the same sample size (the exact same sample size would require additional filtering, which was not considered here).
- Activities with a much higher sample size than the rest of the activities (e.g., the activity "car_driving" in case of participant 1) are not considered at all.
- Use data samples that performed best in the initial n-fold cross validation above.
- All other tests are only conducted for the classifier that performed best in the initial n-fold cross validation above (which is k-NN).

As an example, two activity sets of two different weeks of participant 3 were tested this way. The classification results are represented graphically in Figure 5.10. The activity sets used were from weeks 27 and 28. Figure 5.10(a) shows the case where the classification model was created with week 27 and week 28 used as a test set. In Figure 5.10(b), the same has been done in the other direction. The corresponding overall accuracies are 62.80% for the first case and 63.96% for the second one.

It can be seen, that only the activity locomotion could have been relatively well recognized (although there are severe wrong predictions towards the activities "car_driving" and "work"). The activity "work" is in more than half the cases classified as another activity. The case gets even worse for the activites "car_driving" and "hygiene" that are more predicted for something else than the correct activity.

This brief interpretation can further be elaborated to find out reasons for the wrong classifications and take corresponding countermeasures.

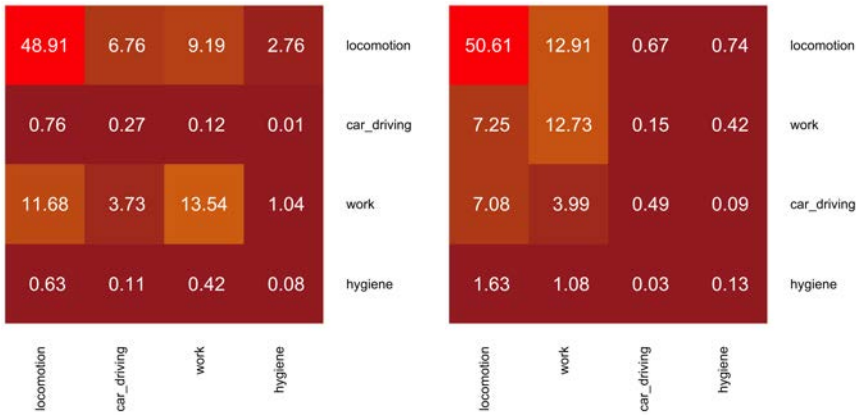


Figure 5.10: Person intra-frame recognition (in both directions)

5.3.2.2 Inter-Person Recognition

To find out the accuracy of a recognition model for one participant when applied on the sensor data from another participant the same precondition as for intra-person recognition must be considered. This time Table 5.2 can be used first to initially see, if a certain type of activity has even been labeled by both participants. If this is the case, still the tables in Appendix A.3 need to be examined to see if the corresponding activity is in both data sets that have to be used for inter-person recognition.

Some notes worth to include are:

- The personal interpretation of an activity might be completely different, e.g., doing office work is other than doing manual work.
- Therefore a rather low recognition accuracy is expected.

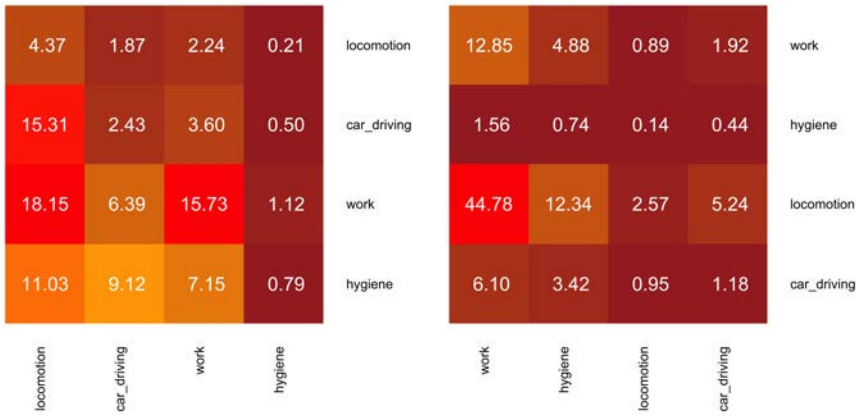


Figure 5.11: Intra-person recognition (in both directions)

In the example, visualized in Figure 5.11, week 27 of participant 3 was evaluated against week 43 of participant 1. They were chosen, as they showed the best comparability regarding activity labelings. In Figure 5.11(a) the classification model was created with week 27 from participant 3 and week 43 from participant 1 used as a test set. In Figure 5.11(b), the same has been done in the other direction. The corresponding overall accuracies are 23.31% for the first case and 17.35% for the second one.

In the first example, the activity "work" has a higher self-recogntion rate but is also very often wrongly predicted as the activity "locomotion". All other activities show low accuracies and high false positive rates compared to the other activities. The other direction is similar, where the activity "work" shows the best single activity recognition and all other are more classified to other activity classes than the correct one. Remarkable is the wrong prediction of "locomotion" as "work".

Such observations already provide indications about which combinations perform better between participants and which have to be treated very carefully. Especially, to provide a mapping between participants, more elaborate models at the numeric and nominal semantic layers have to be developed.

5.4 Optimization Strategies

Some of the filters that have been applied to the data set have already been shown, especially activity class type filters that completely dismiss activities of a certain type. What have not been discussed by now, are the effects that these filters have on classification. Additionally, another filter type is considered, which increases accuracy.

5.4.1 Signal Based Filtering

Suggested in Section 5.2, the implications of removing features from our data sets is briefly shown.

From the visual inspection, we agreed that the mean feature types have a much more separating characteristic as the standard deviation features. So one should think, that by removing the standard deviation features, the recognition rate would increase. In the following Table (5.3) the recognition accuracies for different combinations of features are shown. The data set applied is the same that is used to generate the plots of Section 5.2. Additional combinations, for which no plot is available are tested, too.

Combination	mx/my	mx/mz	my/mz	mx/my/mz	my/sy	sx/sy	sx/sy/sz
Accuracy	64.30	59.61	61.96	**75.45**	53.69	44.77	45.53

Table 5.3: Recognition accuracies for special feature combinations [%]

mx/my/mz stand for the mean values in all three dimensions, sx/sy/sz are the corresponding standard deviation features. Our first assumption, regarding the better performance of mean based features against standard deviation based features turns out to be true. But, although all mean feature combinations perform better than all standard deviation features, using mean features only pairwise performs not so good as when using the classifier with all three mean features together (75.45%). This is even better than what was reached in initial classification (72.90%), yielding an improvement of **2.55%**.

As conclusion of this part of the evaluation it can be said, that the standard deviation feature of a signal alone can not beat the mean feature. The indication of better performance when using all mean features together, while completely dismissing the standard deviation features shows potential to be further explored.

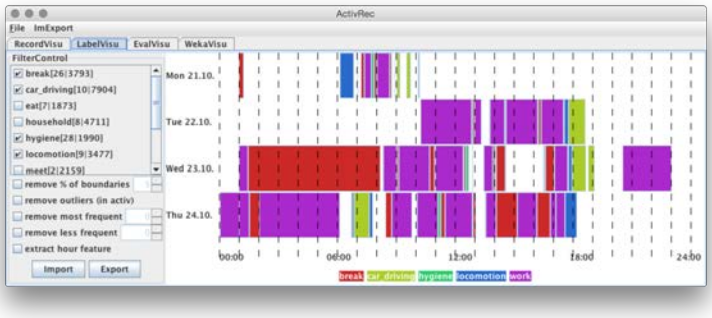
5.4.2 Activity Based Filtering

Besides enhancements over signal based analysis, properties specific to the activity type and its characteristics within the data set can be used likewise. Here, we end the discussion about the functions of the LabelVisu tool by showing an example for filtering depending on activity properties.

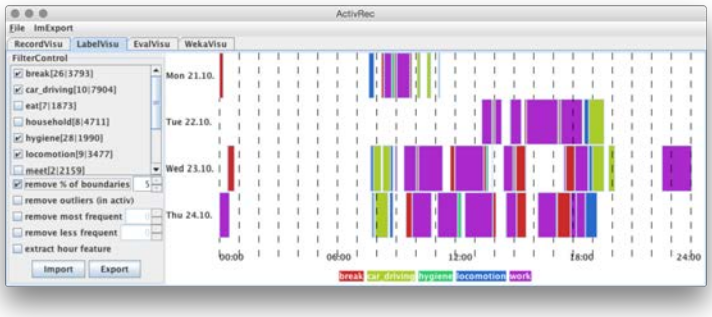
The type of filters that can be applied with the LabelTool have already been presented in Section 4.2.2. Here, first the application of a specific filter type is discussed and in the next section the corresponding consequences on classification are presented.

In Figure 5.12(a), the activity traces of week 43 of particiapant 1 without any filter applied, are shown. From the panel on the left it can be seen, that some activities have been deselected. Actually, the same data set, whose classification results are shown in 5.13 (c) and (d), is used for this representation.

In Figure 5.12(b), in the left control panel it can be seen, that the "remove % of boundaries" filter with a rate of 5% has been applied. The corresponding view is shown in the main panel of the window.



(a)



(b)

Figure 5.12: Week 43 of participant 1 in (a) raw and (b) filtered form

A remarkable, unusual ocurrance is that the bigger labels for break and working on the morning of the last two days nearly vanishes after filtering. The reason for this disappearance is that the filter operates on the sample sizes of activities instead of their durations. It removes 5% of the samples from the start and the end of each activity item existing in the data set.

In case of these two activity items, it looks that the activity was only set at the start and the end of the activity item, not changed in between but the smart watch in reality not used to perform the corresponding activity. This way it gives a rather long duration for the activity although in reality it consists of only a low number of samples.

By the application of the mentioned filter such phenomena can be suppressed and a more realistic picture is retrieved.

5.4.3 Evaluation Results for Filtered Data

Finally, in this section the data filter presented in the last section is applied to classification, and it is revised if any improvement for classification can be reached, by the application of the considered filter.

In Figure 5.13, corresponding heatmap representations are shown for considering four (cf. Figure 5.13(a) and (b)) and five (cf. Figure 5.13(c) and (d)) activities. The corresponding classification accuracies are presented in Table 5.4. It can be seen, that in both cases the accuracy increases, when using five instead of four activities, the increase is even a bit higher.

	before filtering	after filtering	change
4 activities	76.79	78.56	+1.77
5 activities	77.76	80.31	+2.55

Table 5.4: Recognition accuracies for bound filter [%]

In Figure 5.13, on the left side the unfiltered heatmaps and on the right side the heatmaps after the filter has been applied is depicted. Interestingly, only from the visual representation there is no real change observable. This is the reason why the relative percentage values instead of the sample distribution values have been used for the heatmap representation. Although the views do not look really different, when looking at the corresponding values it can be seen, that the values in the main diagonal are increased by a small portion whereas the values in the upper and lower diagonal matrix are decreased a little. By this it can be seen, that a true recognition enhancement can be reached.

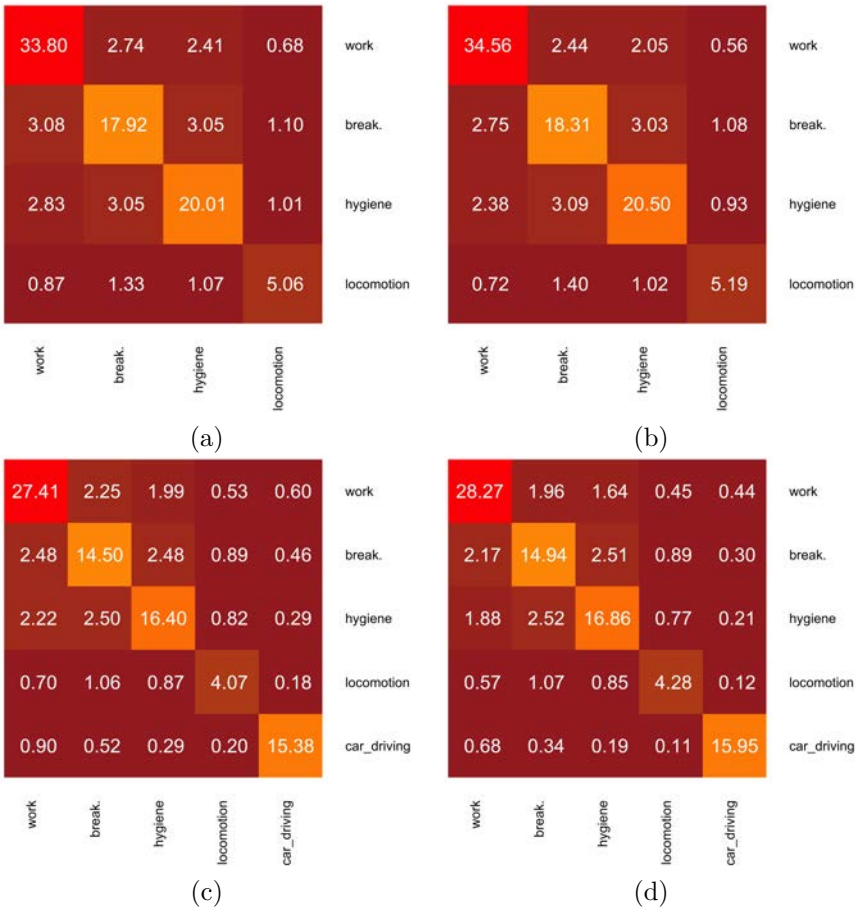


Figure 5.13: Heatmap representation of implications of bound filter in combination with activity filter

In this chapter various methods regarding smart watch accelerometer data representation, analysis and evaluation result illustration have been shown. From the initial results obtained, it can be seen, that high potential for further increasing result quality and application of more complex modelling exist. This is out of scope for this work, but some suggestions are provided in the outlook section.

Chapter 6

Outlook and Conclusions

This thesis provides answers to challenges when developing and deploying activity recognition systems within real world scenarios. Starting with a system specification to achieve long term data collection up to different approaches of how to evaluate the acquired data a wide range of topics have been considered that all contribute to the systematic treatment of activity recognition.

In this thesis the question of how good can activity recognition be performed on wrist worn smart watch platforms was explored.

After introductional and formalizing statements the state-of-the-art was researched. To justify the initial assumptions a framework and corresponding editing tools (workbenches) were designed and implemented.

To see how the proposed and utilized software components are applied in real-world settings a vast dataset was recorded and evaluated. This evaluation analysis showed which performance based on the given dataset applied on the software components can be expected.

6.1 Future Work

During the description of this work a lot of issues appeared that show significant potential for improvement. Due to the practical nature of this work, application dependent extensions are imaginable in multiple directions. In this section the most important points are enlisted of which measures can be taken in the future to further enhance recognition results and advance practical application for every day use.

6.1.1 Scientific Considerations

- As is seen in Section 5.1.3, the handling and management of big data sets is a challenging task. When the concepts of this work are applied to life long and long-term data collection and evaluation, additional methods have to be considered to process data effectively.

- The consideration of time based properties need to be extended, to rate classifications not only dependent on sensor sample values, but also on temporal effects. Not only the key characteristics of the sensor data values will be of interest but also the time at which a sensor sample emerged in relation to the activity it is representative for. This is also of interest for the aspect that misclassifications might be evaluable more accurate because reasonings can be achieved as at which time a sensor sample was classified incorrectly.
- The feature filters, presented in Section 5.4, can be enhanced in the way that they are automatically applied according to key characteristics extracted from summary statistics. Metrics have to be specified as which compositions of feature values (and their appearance in time) give which characteristics and which are desired and which not. In this work this process is outlined manually, for further work this can be automated and extended for additional characteristics not considered here.

6.1.2 Practical Considerations

Extensions for practical deployment are strongly influenced by application specific subjects. The following bullets give an overview of which aspects shall receive increased attention for further work.

- The run time system has to be equipped with quality of service routines, especially regarding classification, so that wrong predictions are intercepted before they will be forwarded to any following decision entity. The integration of multiple sensor modalities (e.g., additional inertial sensors or environmental sensors) can be used to enhance overall classification results by dynamically deciding which predictions to trust and which not.
- Another issue is the extension of the practical experiment to a larger number of study participants. By such an approach more evidence will get created to retrieve more reliable results. This work was mainly concerned of how such a system can be established practically with a lower number of experimental showcases to show the general feasibility.
- Real-world simulation methods shall get forced, as this would on one side simplify runtime software development and on the other side soon enough show effects that have to be taken care of in a real-world runtime system.

6.2 Contributions to the Research Objectives

At the end of this work the research objectives, stated in the introduction, are considered once again and a recapitulation in the context of the experiences acquired in this work is done, to state corresponding answers.

- *Can activities be recognized by the application of a smart watch accelerometer data collection and evaluation system. How accurate can activities be recognized.*

  In principle this question can be answered positively as initial evaluation results have shown that depending on the participant, the classifier used and the filters applied rather good results can be achieved. In detail there exist special cases that restrict a global validity of the presented results. To get around such special cases deeper analysis of the true causes, why these cases fail, needs to be done.

  Depending on the participant initial person inter-frame recognition rates show poorer results than the initial cross-validation outcomes. The case get even harder when performing inter-person recognition. Therefore, it can be said that person inter-frame recognition rates can be enhanced by applying the optimization strategies presented. For inter-person recognition additional approaches have to be taken, as the differences of recordings between persons is much higher compared to the commonly existing properties.

- *Which aspects must be considered to meet real-world requirements with special regard to every day use and practical result utilization. Which constraints apply in practical application and how can they be handled.*

  To answer this question, different aspects need to be taken into account that are mainly involved with practical concerns.

  - For the developed runtime system it has been shown that it can be successfully utilized in real-world settings. This is also emphasized by the elaborate experiment conducted.

    Technical problems like battery lifetime of mobile devices, system availability or automated handling of processes were treated in this work to meet the identified requirements. For a widespread usage, these technical constraints will need more elaboration to successfully work in completely uncontrolled environments.
  - Regarding data analysis, a fully implemented toolset that allows deep inspection of every stage of the ARC has been designed, implemented and shown in practice. Of course, improvements can be added in various directions but the general outline was clearly presented in this work.

- *Which functionalities and analysis routines must a corresponding system implementation support to meet the requirements mentioned in Section 1.1, Motivation.*

  Various methods have been presented that show the feasibility of activity recognition under practical constraints. The realization of the different stages of the ARC has been specified and implemented for real world deployment. Beside the traditional ARC approach containing preprocessing,

segmentation, feature extraction and classification, extensions have been introduced that take care of aspects which are treated to a lesser extent in related work. They are: data recording, feature filtering and application. By these extensions the real world system deployed in the practical experiment could have been established which led to the successful conduction of the experiment and extraction of promising results.

- *Can initial recognition accuracy be increased? Show strategies to improve recognition results.*

  Several strategies have been shown to increase recognition rates. This part can be further elaborated by whether including extended sensor modalities or extending the methods for optimization. Additional analysis of sensor samples, feature key characteristics or classification methods can increase detection rates. An important point is that always practical considerations must be kept in mind. By the measures presented in Section 5.4 it can be seen that potential for improvement of recognition accuracy exists.

Appendix A

Appendix

A.1 Armband Comparison

The following table gives an overview of models that has been revised for arm band based activity recognition.

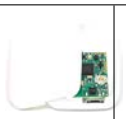
	 Fitbit	 Withings	 iHealth	 Shimmer	 Meta Wear	 Angel Sensor	 Galaxy Gear Fit
Type	Arm band/- Fitness	Arm band/Fit- ness	Arm band/- General Purpose	General Purpose	General Purpose	Arm Band/- General Purpose	Arm Band/- General Purpose
Model	Commercial	Commercial	Commercial	Academic	Academic	Academic	Commercial
Data format	Pre- processed	Pre- processed	Pre- processed	Raw	Raw	Raw	Pre- processed
Battery Life	a few days	weeks	weeks	hours	hours	NA	a feaw days

Table A.1: Arm band and general purpose wearables overview

A.2 Smart Watch Comparison

The following table gives an overview of models that has been revised for smart watch based activity recognition. Reasons for prefereneces are explained in the text.

SC stands for single core, DC stands for dual core.

	 Moto ACTV	 Galaxy Gear	 Galaxy Gear S	 Moto 360	 LG G Watch R	 Pebble	 Chronos
Tech Spec	OMAP3 600MHz (SC), 8GB Flash, 256MB RAM	Exynos 800MHz (SC), 4GB Flash, 512MB RAM	Snap-dragon 400 1.2GHz (SC), 4GB Flash, 512MB RAM	OMAP3 1GHz (SC), 4GB Flash, 512MB RAM	Snap-dragon 400 1.2GHz (SC), 4GB Flash, 512MB RAM	ARM Cortex M3 80MHz (SC), 512 KB Flash, 128KB RAM	MSP430 160MHz (SC), 32KB Flash, 4KB RAM
Connec-tivity	WiFi, BT	BT	WiFi, BT	BT, (WiFi)	BT, (WiFi)	BT	433 MHz Band
Sensors	Ac-celerome-ter, Magne-tometer	Ac-celerome-ter, Heart Rate, Camera	Ac-celerome-ter, Heart Rate, Camera	Ac-celerome-ter, Heart Rate	Ac-celerome-ter, Heart Rate	Ac-celerome-ter	Ac-celerome-ter
Battery life	8 - 12 hrs	6 - 8 hrs	6 - 8 hrs	6 hrs	8 hrs	7 days	10 days
OS	Android	An-droid/Ti-zen	Tizen	Android Wear	Android Wear	Custom Free RTOS	Custom

Table A.2: Smart Watch Overview

A.3 Data Samples Details

In the following tables all the data samples collected for the participants involved in the experiment of this work are shown. They are divided by the types of activities that have been recorded and the time when they have been recorded resolved by week. Details and a discussion regarding the collection process and interpretations of certain values are presented in the text. The values in brackets show how often the corresponding activity has been selected in the concerned period.

An elaborate discussion of the values presented in the following tables is provided in Section 5.1.1.

A.3.1 Participant 1

	w24	w25	w26	w27	w28	w29	w30	w31
break		16312(21)	32259(25)	24562(23)	12046(15)	28696(27)	17754(20)	14516(14)
car_driving		80646(13)	194013(19)	165745(20)	145203(32)	131153(20)	79714(18)	36255(4)
eat		5449(7)	17091(14)	5850(6)	6119(9)	17450(14)	11330(12)	9174(6)
education								
family_care		1456(1)	33494(17)	25657(17)	53514(22)	63571(36)	53988(27)	18221(7)
household				3821(1)	8871(5)	23601(13)	39658(14)	2025(1)
hygiene	66(1)	19435(28)	27181(34)	31318(32)	28282(24)	35118(39)	31921(38)	30174(23)
locomotion		11800(7)	27439(9)	43482(9)	7212(10)	7227(4)	5397(5)	11102(3)
meet		1133(2)	5177(5)	6876(2)	13160(3)	7005(1)	116(1)	
none		15154(11)	49192(27)	19732(14)	21466(17)	26693(15)	17465(13)	3955(2)
read								
shop		2222(4)	2624(2)	1844(2)	9722(6)	10019(5)	8045(7)	817(1)
sleep		2551(2)	6390(7)	2854(4)	8596(4)	9944(6)	5481(5)	4631(4)
socialize		10484(5)	2324(3)	8039(1)	7929(4)	8897(3)	5847(2)	976(1)
travel					77(1)			
tv		137(3)			1132(2)	1492(3)	337(1)	
work	169(1)	54344(47)	75259(49)	61982(39)	45951(32)	338785(26)	47496(29)	27802(24)
TOTAL	235	221123	472443	401762	369280	404651	324549	159648

Table A.3: Activity Samples of Participant 1

	w36	w38	w39	w40	w41	w42	w43	w45
break		7941(8)	14972(10)	37039(30)	48682(33)	52406(33)	24775(26)	13(1)
car_driving	68383(8)	55169(9)	124091(19)	80108(11)	74624(9)	117776(13)	20589(10)	332(2)
eat	911(1)	3592(5)	5081(5)	6320(8)	20624(11)	10890(10)	3877(7)	
education								417(1)
family_care	3045(2)	13958(5)	10767(5)	19712(11)	13924(8)	18747(8)		
household	28(1)			14333(5)	43092(10)	21306(7)	9492(8)	
hygiene	4597(1)	8592(9)	12608(14)	26926(27)	428104(39)	26711(29)	26475(28)	184(1)
locomotion	2060(3)	21249(12)	3670(4)	29647(6)	35389(7)	27014(10)	8195(9)	
meet	88(1)	4621(4)	562(1)	957(2)	719(1)	3245(2)	2159(2)	
none	9240(4)	43718(3)	11235(6)	10722(8)	9225(5)	20256(7)	1099(2)	1018(5)
read		865(1)						
shop	2894(2)		4428(5)	3827(2)	4842(2)	2798(1)	834(4)	181(4)
sleep		7466(4)	1020(1)	4088(4)	6550(4)	1104(1)	972(1)	36(1)
socialize		5172(2)	2233(1)		1656(1)	2326(3)		6(1)
travel		3389(2)						
tv				540(1)		819(1)		21575(2)
work	2010(3)	38367(19)	33849(20)	58760(33)	70005(45)	43220(39)	39023(31)	93(2)
TOTAL	89216	214099	224516	292979	372139	348618	137490	23855

Table A.4: Activity Samples of Participant 1 (ctd.)

	w46	w47	w48	w49	w50	w51	w52	TOTAL
break	18635(15)	14949(9)	15850(13)	14085(13)	39076(18)	13198(13)		447766
car_driving	52762(11)	57395(8)	86650(15)	141891(11)	45930(6)	79946(8)		1838375
eat	3464(5)	2494(3)	3147(5)	2724(3)	4384(5)	2936(3)		142907
education								417
family_care	6815(2)	10970(4)	9284(4)		4613(1)	67(1)		361803
household	5406(4)	5079(1)			10620(3)	2679(1)		190011
hygiene	19315(17)	14485(13)	22324(17)	29018(16)	33331(25)	14508(17)		870673
locomotion	28214(6)	24317(4)	22909(5)	28925(6)	29417(7)	14245(6)		388910
meet	1773(3)	1488(1)	1989(1)	8243(3)	3551(3)	10272(6)		73134
none	6958(6)	3175(4)	20725(5)	21906(4)	4529(5)	32383(4)	37845(2)	387691
read								865
shop	11341(4)	827(1)	30387(6)	15139(4)		889(1)		113680
sleep			2481(1)					63192
socialize		2051(1)	2321(1)	12491(2)		2789(2)		75541
travel								3466
tv								27004
work	31114(27)	34083(14)	42523(23)	36698(20)	32498(31)	23613(22)	2354(2)	1139998
TOTAL	185797	171313	260590	298629	220440	197525	40199	6125433

Table A.5: Activity Samples of Participant 1 (ctd.)

A.3.2 Participant 2

	w38	w39	w40	w41	w42	w43	w44	w45	TOTAL
break_coffee				1197(1)		2176(1)			3373
car_driving		35518(7)	140164(12)	70467(7)	38246(7)	8965(5)	14317(6)	15060(5)	322737
coffee		2863(1)		1625(1)					4488
computer		10313(4)	11552(3)	11201(3)	13167(6)	23687(3)	14328(2)	7977(4)	92225
cook		1991(2)	18553(2)	2938(1)			26406(3)	21502(2)	71390
eat		9453(5)	15269(5)	4554(5)	1266(2)	113(1)	1443(1)	16260(2)	48358
education				53(1)					53
family_care			11220(3)	6437(2)					17657
household	5373(1)	24884(3)	6335(3)	13784(1)		13672(3)	5518(2)	7990(2)	77556
hygiene	2289(1)		24053(3)	1685(3)					28027
locomotion		7793(3)	1726(1)	21423(4)	18324(5)			6548(1)	55814
meet		548(1)	1457(1)	687(1)	2394(1)	18171(1)	3865(1)		10768
none			6(1)	2(1)					8
pray			580(2)	5787(1)			8487(1)	774(1)	15628
read		463(1)							463
relax_think				930(2)					930
shop					2499(2)	15061(2)	1424(1)		18984
socialize		1073(2)	10115(3)	12353(5)			1355(1)		24896
sports			7171(1)						7171
work	823(1)	5347(5)	47883(4)	3727(4)					57780
TOTAL	8485	100246	296084	158850	75896	65491	77143	76111	858306

Table A.6: Activity Samples of Participant 2

A.3.3 Participant 3

	w26	w27	w28	w29	w30	w31	w32	w33	w34
break	2249(4)	111(1)			274(1)				
car_driving	3030(4)	22084(9)	1654(2)	3279(3)	2602(3)	9167(5)	901(2)	2282(2)	1155(1)
eat			12166(1)			1417(1)	2835(1)		
hygiene	1890(1)	5444(4)	1756(1)			4715(1)			
locomotion	32137(9)	123058(25)	96208(16)	66694(12)	48524(10)	30835(8)	14385(4)	40214(9)	34128(7)
meet		664(1)	9825(1)						
none	161(1)	26(1)							
shop	488(1)	1063(1)						736(1)	
sleep	175(1)	716(3)	454(2)		604(3)	1776(3)		253(1)	445(1)
socialize	2021(1)	5816(4)	1806(2)	26753(2)	6797(1)	10275(2)	14830(1)		1007(1)
tv	798(4)	228(1)							
work	2534(6)	38926(13)	42668(9)	39315(9)	56523(7)	31848(6)		22510(8)	12904(5)
TOTAL	45483	198136	166537	136041	115324	90033	32951	69995	49639

Table A.7: Activity Samples of Participant 3

	w35	w36	w37	w38	w39	w40	w41	w42	w43	TOTAL
break			78(1)							2712
car_driving	3659(5)	874(3)	2004(7)	2223(4)						54914
eat										16418
hygiene										13805
locomotion	32692(1)	7086(6)	15998(10)	26346(11)	1364(1)	14943(13)	2245(3)		726(1)	587583
meet	1568(1)	976(3)		2399(5)			4135(3)		1717(3)	21284
none		215(1)								402
shop										2287
sleep		140(1)								4563
socialize										69305
tv										1026
work	36770(7)	37871(8)	30524(9)	22588(14)	8783(2)	35808(14)	3257(4)	3465(1)	8153(3)	434447
TOTAL	74689	47162	48604	53556	10147	50751	9637	3465	10596	1208746

Table A.8: Activity Samples of Participant 3 (ctd.)

Declaration of Authorship

English

I certify that the work presented here is to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other University.

The present diploma thesis is identical with the electronically transmitted text document.

Deutsch

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, June 2015 Peter Halbmayer

Bibliography

[1] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2014.

[2] Marco Altini, Julien Penders, Ruud Vullers, and Oliver Amft. Combining wearable accelerometer and physiological data for activity and energy expenditure estimation. In *Proceedings of the 4th Conference on Wireless Health*, WH '13, pages 1:1–1:8, New York, NY, USA, 2013. ACM.

[3] David Bannach, Oliver Amft, and Paul Lukowicz. Rapid prototyping of activity recognition applications. *IEEE Pervasive Computing*, 7:22–31, 2008.

[4] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive Computing*, volume 3001 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2004.

[5] Gerald Bieber, Thomas Kirste, and Michael Gaede. Low sampling rate for physical activity recognition. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '14, pages 15:1–15:8, New York, NY, USA, 2014. ACM.

[6] Gerald Bieber, Thomas Kirste, and Bodo Urban. Ambient interaction by smart watches. In *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '12, pages 39:1–39:6, New York, NY, USA, 2012. ACM.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[8] Marko Borazio and Kristof Van Laerhoven. Improving activity recognition without sensor data: A comparison study of time use surveys. In *Proceedings of the 4th Augmented Human International Conference*, AH '13, pages 108–115, New York, NY, USA, 2013. ACM.

[9] Marko Borazio and Kristof Van Laerhoven. Using time use with mobile sensor data: A road to practical mobile activity recognition? In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, pages 20:1–20:10, New York, NY, USA, 2013. ACM.

[10] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.*, 46(3):33:1–33:33, January 2014.

[11] Jose M. Chaquet, Enrique J. Carmona, and Antonio Fernández-Caballero. A survey of video datasets for human action and activity recognition. *Comput. Vis. Image Underst.*, 117(6):633–659, June 2013.

[12] Liming Chen and Chris Nugent. Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems*, 5(4):410–430, 2009.

[13] Liming Chen and Chris D. Nugent. Ontofarm: An ontology-based framework for activity recognition and model evolution. *ERCIM News*, 2011(87), 2011.

[14] Heng-Tze Cheng, Martin Griss, Paul Davis, Jianguo Li, and Di You. Towards zero-shot learning for human activity recognition using semantic attribute sequence model. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 355–358, New York, NY, USA, 2013. ACM.

[15] Heng-Tze Cheng, Feng-Tso Sun, Martin Griss, Paul Davis, Jianguo Li, and Di You. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 361–374, New York, NY, USA, 2013. ACM.

[16] G. Costante, L. Porzi, O. Lanz, P. Valigi, and E. Ricci. Personalizing a smartwatch-based gesture interface with transfer learning. In *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pages 2530–2534, Sept 2014.

[17] Merijn de Bakker. Automatic classification of bird behaviour on the basis of accelerometer date. bachelor thesis. Master's thesis, 2011.

[18] Anind K. Dey, Gregory D. Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16(2):97–166, December 2001.

[19] Zhongli Ding and Yun Peng. A probabilistic extension to ontology language owl. In *HICSS*, 2004.

[20] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, Nov 2001.

[21] Enrique Garcia-Ceja and Ramon Brena. Long-term activity recognition from accelerometer data. *Procedia Technology*, 7(0):248 – 256, 2013.

[22] Dawud Gordon, Jürgen Czerny, and Michael Beigl. Activity recognition for creatures of habit. *Personal Ubiquitous Comput.*, 18(1):205–221, January 2014.

[23] Peter Halbmayer, Gerold Hoelzl, and Alois Ferscha. A dynamic service module oriented framework for real-world situation representation. In Dan Tamir David Musliner, Elena Troubitsyna, editor, *The 6th International Conference on Adaptive and Self-Adaptive Systems and Applications, ADAPTIVE 2014, May 25 - 29, Venice, Italy*, pages 79–84. IARIA, May 2014.

[24] Gerold Hoelzl, Marc Kurz, and Alois Ferscha. Goal processing and semantic matchmaking in opportunistic activity and context recognition systems. In *The 9th International Conference on Autonomic and Autonomous Systems (ICAS2013)*, pages 33–39, March 2013.

[25] Reuben Kirkham, Aftab Khan, Sourav Bhattacharya, Nils Hammerla, Sebastian Mellor, Daniel Roggen, and Thomas Ploetz. Automatic correction of annotation boundaries in activity datasets by class separation maximization. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, pages 673–678, New York, NY, USA, 2013. ACM.

[26] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, August 1988.

[27] J. Mantyjarvi, J. Himberg, and T. Seppanen. Recognizing human motion with multiple acceleration sensors. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 2, pages 747–752 vol.2, 2001.

[28] Ross Messing. *Human Activity Recognition in Video: Extending Statistical Features Across Time, Space and Semantic Context*. PhD thesis, Rochester, NY, USA, 2011. AAI3498317.

[29] B.J. Mortazavi, M. Pourhomayoun, G. Alsheikh, N. Alshurafa, S.I. Lee, and M. Sarrafzadeh. Determining the single best axis for exercise repetition recognition and counting on smartwatches. In *Wearable and Implantable Body Sensor Networks (BSN), 2014 11th International Conference on*, pages 33–38, June 2014.

[30] Jorge Luis Reyes Ortiz. *Smartphone-Based Human Activity Recognition*. Springer Publishing Company, Incorporated, 2015.

[31] Heather S. Packer, Gustavo Buzogany, Daniel Alexander Smith, Laura Dragan, Max Van Kleek, and Nigel R. Shadbolt. The editable self: A workbench for personal activity data. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '14, pages 2185–2190, New York, NY, USA, 2014. ACM.

[32] Kurt Partridge and Philippe Golle. On using existing time-use study data for ubiquitous computing applications. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, pages 144–153, New York, NY, USA, 2008. ACM.

[33] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys Tutorials, IEEE*, 16(1):414–454, First 2014.

[34] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3*, IAAI'05, pages 1541–1546. AAAI Press, 2005.

[35] Daniele Riboni and Claudio Bettini. Cosar: Hybrid reasoning for context-aware activity recognition. *Personal Ubiquitous Comput.*, 15(3):271–289, March 2011.

[36] Daniele Riboni, Linda Pareschi, Laura Radaelli, and Claudio Bettini. Is ontology-based activity recognition really effective? In *PerCom Workshops*, pages 427–431. IEEE, 2011.

[37] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Marco Creatura, and José del R. Millán. Collecting complex activity data sets in highly rich networked sensor environments. In *Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS), Kassel, Germany*. IEEE Computer Society Press, June 2010.

[38] Daniel Roggen, Kilian Förster, Alberto Calatroni, and Gerhard Tröster. The adarc pattern analysis architecture for adaptive human activity recognition systems. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–18, 2011.

[39] Saguna Saguna, Arkady Zaslavsky, and Dipanjan Chakraborty. Complex activity recognition using context-driven activity theory and activity signatures. *ACM Trans. Comput.-Hum. Interact.*, 20(6):32:1–32:34, December 2013.

[40] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[41] Johannes A. Stork, Luciano Spinello, Jens Silva, and Kai O. Arras. Audio-based human activity recognition using non-markovian ensemble voting. In *Proc. IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN'12)*, Paris, France, 2012.

[42] Edison Thomaz, Vinay Bettadapura, Gabriel Reyes, Megha Sandesh, Grant Schindler, Thomas Plötz, Gregory D. Abowd, and Irfan Essa. Recognizing water-based activities in the home through infrastructure-mediated sensing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 85–94, New York, NY, USA, 2012. ACM.

[43] DanielT. Wagner, Andrew Rice, and AlastairR. Beresford. Device analyzer: Understanding smartphone usage. In Ivan Stojmenovic, Zixue Cheng, and Song Guo, editors, *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, volume 131 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 195–208. Springer International Publishing, 2014.

[44] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers Inc., 2005.

[45] Robert Xiao, Gierad Laput, and Chris Harrison. Expanding the input expressivity of smartwatches with mechanical pan, twist, tilt and click. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 193–196, New York, NY, USA, 2014. ACM.

[46] Juan Ye, Simon Dobson, and Susan McKeever. Situation identification techniques in pervasive computing: a review. Pervasive and Mobile Computing, 2011.

[47] Juan Ye, Graeme Stevenson, and Simon Dobson. Usmart: An unsupervised semantic mining activity recognition technique. *ACM Trans. Interact. Intell. Syst.*, 4(4):16:1–16:27, November 2014.

[48] Chun Zhu and Weihua Sheng. Motion- and location-based online human daily activity recognition. *Pervasive Mob. Comput.*, 7(2):256–269, April 2011.