Tracking of Human Motion over Time

Pijl, M.J.

Publication date:
2016

Document Version
Publisher's PDF, also known as Version of record

Tracking of human motion over time

Marten Pijl

Tracking of human motion over time

ISBN: 978-94-6233-492-2

Photo by Eadweard Muybridge

The work described in this thesis has been carried out at the Philips Research Laboratories in Eindhoven, the Netherlands, as part of the Philips Research programme.

Tracking of human motion over time

Proefschrift ter verkrijging van de graad van doctor
aan Tilburg University
op gezag van de rector magnificus,
prof. dr. E. H. L. Aarts,
in het openbaar te verdedigen ten ovenstaan van een
door het college voor promoties aangewezen commissie
in de aula van de Universiteit
op woensdag 14 december 2016 om 14.00 uur

door

Marten Jeroen Pijl

geboren te Noordhorn.

Promotores: prof. dr. E. H. L. Aarts
prof. dr. M. M. Louwerse
Copromotor: dr. ir. J. H. M. Korst

Overige leden van de Promotiecommissie:
prof. dr. ir. C. M. Jonker
prof. dr. ir. J. J. Lukkien
prof. dr. E. O. Postma
prof. dr. W. Van Petegem
prof. dr. B. E. R. de Ruyter

Contents

1

Introduction

This thesis concerns the analysis of human motion through sensors placed on the body or in the environment. It is our ability to move around and interact that allows us to have an impact on our environment; it allows us to shape and alter our environment, and to address our needs through interaction. As a result, there is a plethora of information to be unlocked by measuring and interpreting human motion, ranging from low-level mechanical abilities to high-level behavioral intentions. Low-level measurements might include the speed and distance of a single footstep, while high-level interpretations might extend activities such as cooking, or running as an expression of our intent to stay healthy. A lot of the information locked away in our movements relates to our health - insufficient movement can lead to health problems, while injury or illness affects how we move.

Unlocking this information is not trivial however, as we need to rely on the interpretation of sensor measurements. While there is a wide variety of sensors that can be applied to measuring human movements, it is rarely possible to measure the desired property (e.g., an activity, gait, or posture) directly. Instead, we rely on the interpretation of acceleration signals, signal strengths, and the like. In addition, sensors are affected by measurement noise or false readings. As a result, obtaining meaningful interpretations of human motion from sensor data such as acceleration data or camera images is a major challenge in the field of human motion tracking.

A major part of addressing this challenge is the use of machine learning algorithms to make sense of the data. Algorithms that take the ordering of the sensor data

into account (such as hidden Markov models and Kalman filters) are of particular interest here, as sensor data is generally ordered in time. Machine learning algorithms can help in recognizing patterns in data, or distinguishing between different activities or movements.

In addition to machine learning techniques, signal analysis also plays a role in processing the raw sensor data. This includes removing noise from the sensor measurements, but also the extraction of features which can be used as input for machine learning algorithms. In the remainder of this thesis, we will use techniques from fields such as machine learning, data science and signal processing to discuss methods and applications of tracking and interpreting human motion. In particular, within this thesis we aim to address the following main research question:

> *How can segmentation, classification, and regression be applied to problems that involve the tracking and interpretation of human motion?*

To address the research question, we first detail a model based on the three components of segmentation, classification and regression. Note that we do not require each of these three elements to be present in every problem addressed by the model; for instance, many problems require either a classification approach or a regression approach.

The model for motion tracking will be introduced in more detail in Section 1.6. Here, we introduce the segmentation, classification, and regression components of the main research question as three optimization problems that are often encountered in human motion tracking. In the remaining chapters of this thesis, we will describe how these problems can be addressed in the context of a number of practical applications of human motion tracking. The individual applications are briefly introduced in Section 1.8; in each chapter, we will address one or more of the components of the main research question.

To address the problems of segmentation, classification and regression for a number of applications in human motion tracking, we formulate three research questions related to the main research question. Through these related research questions, we aim to in turn address the main research question of this thesis.

- Can activities of daily living (ADL) be unobtrusively tracked and recognized?
- Can analyzing the behavior of people trying to be more physically active help predict if they will drop out of a lifestyle physical activity program?
- Is it possible to determine (psycho)motor skills such as gait accurately using wearable sensors?

These related questions were in part inspired by real-life applications of human motion tracking in the lifestyle and healthcare domains during the work on this thesis. Within the context of the main research question, we will address these related questions in the thesis' upcoming chapters. In the remainder of the introduction, we will

first provide a background and context to the field of human motion tracking, introduce commonly applied methods and techniques, introduce a model for motion tracking and discuss, and finally discuss the challenges of the field.

1.1 Background

Whether it is a person's health, mood or state of mind, there is a lot that can be learned simply through observing someone's movements. Increasingly, the interpretation of motion is carried out by computational intelligence systems[1] that can be worn on the body, integrated into the environment or in existing devices like smartphones. Advantages of using computationally intelligent systems include that they can be ubiquitous, minimally obtrusive, and detect subtle differences which can be hard to see with the naked eye. The challenge for such systems is to process the raw sensor data, and arrive at meaningful interpretations.

While measuring human motion or activities is not an entirely novel topic, improvements to sensor hardware over the last few decades have certainly made it a more practical one. This is in large part due to improvements in miniaturization and power consumption, in particular related to the continued development of microelectromechanical systems (MEMS) and their widespread use in sensor applications [Shaeffer, 2013]. Miniaturization of sensors means that sensors such as accelerometers and GPS receivers can now be fitted into smaller devices, or are more easily integrated into existing devices. The best example of this is probably the smartphone - it is rare to find one which does not at least include an accelerometer, GPS and WiFi receiver. Sensors not worn on the body, such as ultrasound or pressure sensors, are easier to integrate into the environment (that is, they are less obtrusive and require less space). In all, miniaturization has led to a significant reduction of the obtrusiveness of sensors as well.

Likewise, improvements in power efficiency imply that sensors can operate for longer periods of time without recharging or replacing the battery, or can operate at higher sample frequencies than was previously practically feasible. In some cases, power consumption has decreased sufficiently that sensors which were previously not considered for continuous measurements can now feasibly be used - an example of this is a GPS receiver. In addition, improved power efficiency also benefits miniaturization, as the size of the battery is often a major contributor to the size and weight of a device.

At the same time, society faces a significant number of healthcare challenges. Many of these stem from an increasingly ageing population, leading

[1] While there is no universally agreed-upon definition of computational intelligence, the field generally focuses on problems that can be solved by humans, but are (traditionally) hard to solve using computational techniques; for example, problems that are not mathematically well-defined, or are difficult to model statistically or axiomatically.

to increasing costs for elderly care, with a smaller working population to support these costs [WHO, 2011]. Similarly, people increasingly maintain unhealthy lifestyles [Lim et al., 2010], which can result in conditions such as obesity and cardiovascular disease. Here, behavior monitoring can play an important role in assisting with early diagnosis or through monitoring a person's physical and possibly mental state.

Despite these challenges, the average life expectancies continue to increase [Mathers et al., 2015]. As we, as a society, grow older, there is the expectation that the amount of time we are able to retain our independence and maintain a high quality of life will increase as well. In particular, there is the desire for the elderly to remain at home, avoiding institutionalization, without feeling that they must rely on others for their day-to-day activities. To achieve this, it is important to develop new ways to support the elderly in need of assistance living at home, as well as their caregivers [Stefanov, 2004].

Apart from assistance, another aspect of remaining healthy and retaining a high quality of life is timely diagnosis and efficient treatment of diseases, or detecting and reducing the risk of certain illnesses and conditions. Examples include monitoring or estimating fall risk, cognitive decline, or detecting when someone is not eating well. Here, computationally intelligent systems can help in monitoring for these risks and conditions while maintaining the person's privacy as much as possible.

In addition, healthcare systems are under pressure due to the increase in people leading unhealthy lifestyles, typically caused by a lack of sufficient physical exercise [Stevens et al., 2012]. As mentioned, this can lead to a slew of other conditions, most notably those due to obesity and cardiovascular disease [Must et al., 1999; Thompson, 2003]. There is also evidence suggesting that for elderly, maintaining a level of physical activity may be beneficially impact cognitive ability [Kelly et al., 2014]. Lifestyle activity programs exist to assist participants in changing their habits and leading a more healthy lifestyle, but even with this additional assistance, staying motivated is often a major challenge for participants.

1.2 Monitoring human motion

Human motion and activity can be measured and interpreted at a number of levels. At its most basic level, we can measure properties of a given behavior when it occurs: how fast do we walk, how long does it take us to stand up, and so on. At a higher level we could determine to which goal these activities contribute: is someone walking to the kitchen to get some food, or are they lost? Finally, we can try to make inferences about the mental state of a person, such as their current motivation or mood.

In this thesis, we attempt to address topics at both the basic and higher levels of behavior monitoring. Analyzing walking activity is an example of a low-level behavior. For most people, walking is our primary method of mobility within our

environment, and in many cases enables us to perform higher level behaviors. Statistics such as when and how far we are walking are therefore of interest not only for describing walking behavior itself, but can also help provide insight into higher level behaviors. That is not to say that investigating walking behavior has no merits on its own - walking behavior is often related to various health aspects ranging from leading a healthy lifestyle to cognitive decline.

At a higher level, we can consider activities of daily living (ADL), such as dressing, eating, and so on. Such activities are often composites of several lower-level activities, and as such are often difficult to capture by any single feature. For example, an activity like preparing a meal might be performed very differently by two different persons, or even by the same person on different days.

Finally, at the highest level of interpretation we can use measured behavior as a means to glean understanding of the behavioral intentions of people, which drive their observed behaviors. This often comes down to finding quantifiable values regarding a person's behavior, such as the probability they will perform a certain activity, or a prediction of how often a certain behavior is performed in a period of time. This generally involves observing (aspects of) a person's behavior over a longer period of time, at least compared to the timespan of observing a single activity only.

1.3 Sensing

When it comes to monitoring human motion, there is a wealth of sensors available that offer a variety of uses for this purpose. Broadly speaking, the set of sensors can be divided into two categories: environmental sensors and body-worn sensors. Each category of sensors typically has its own benefits and disadvantages. As a result, the choice of sensor(s) for a particular behavior monitoring application should not only be determined based on the behavior to be monitored, but also on the context of the application itself.

Environmental sensors. The first group, environmental sensors (also called ambient, pervasive, or ubiquitous sensors), consists of sensors which are placed within an environment such as a person's home, workplace, or car. The environmental sensors can then monitor a user within (most of) this environment. In some cases, a single sensor is used, but more commonly a network of sensors, is required to adequately cover the entirety of the environment. Examples of environmental sensors include cameras, microphones, ultrasound range finders, or simple pressure switches.

Body-worn sensors. The second group consists of body-worn (or wearable) sensors; as the name suggests, these are sensors that are worn on or attached to the body of the user. Examples of this include inertial measurement units, accelerometers, and heartrate monitors. Body-worn sensors are usually worn or attached as part of a unit which not only contains the sensor, but also a microprocessor, power source,

any additional sensors, and other electronical components such as a wireless interface or memory - such a unit is often referred to as a sensor platform. Due to their portable nature, these sensor platforms are typically battery powered, and need to be recharged at regular intervals. The frequency of the recharging cycle depends on the power efficiency of the sensor platform in question.

Comparing the two groups of sensors, we can identify three major advantages of environmental sensors compared to wearable sensors. First, environmental sensors tend to be less obtrusive compared to wearable sensors as they are placed in the environment rather than on the body. It should be noted though that obtrusiveness in large part depends on the sensor platform in question; the size (or formfactor), weight and wearing position can all significantly impact the obtrusiveness of the platform.

Second, environmental sensors have the option of connecting directly to the power grid, eliminating the need for recharging. Third, environmental sensors do not depend on the user wearing any special devices, and hence there is no risk of, for instance, the user forgetting to put on the device in the morning. When using wearable sensors, the user must generally remember to charge and wear the sensors. It should be noted that some environmental sensors like RFID systems do require the user to wear a token for identification.

This also touches upon one of the three major disadvantages of environmental sensors compared to wearable sensors: environmental sensors often have trouble distinguishing between different people, or more generally, distinguishing between the information created by different people. This can lead to odd conclusions in some cases, for instance, in cases where a second person was not anticipated, a system may conclude that the user is simultaneously in the living room and the bathroom. However, when the objective is tracking everyone in the environment rather than a specific person, the fact that environmental sensors are not bound to a single user could also be seen as an advantage. Otherwise, coping with this issue requires either using some sort of token to be worn by the user(s), or using algorithms to try and distinguish different users (using for example face or speech recognition).

A second disadvantage of environmental sensors is often the difficulty of installation, particularly if multiple sensors are involved. These sensors often need to be securely fitted into an existing environment, where their functioning is dependent on correct placement. In addition, calibration steps may be required after installation of the sensors in the environment. A final disadvantage is that environmental sensors are obviously not able to monitor a user outside of the sensors' observable environment, whereas a wearable sensor will simply follow the user as they move about.

Hybrid sensors. Arguably, some sensors can be considered hybrids in that they have properties consistent with both groups. An example is wireless beacons, which work through triangulation of signal strengths of the wireless connections of multiple beacons. Such a system both requires several beacons to be installed in the

environment, as well as an active receiver to be worn by the user. Technically speaking, GPS can be considered to belong to this category, even though in this case the environment spans the entire world (as this is the environment covered by the GPS satellites). Sensors in this group typically share disadvantages of both environmental sensors and body-worn sensors, but generally offer some other advantages to make up for this.

For some sensors, it can be debated whether they are hybrid sensors or not. This is particularly the case for some environmental sensors. For instance, a system of RFID sensors generally requires a tag to be placed somewhere on the body. However, this tag can be passive - that is, they do not create any sensor measurements[2]. In the case of wireless beacons, an active unit placed on the body is required, making such sensor systems a more compelling case as a hybrid sensor.

In addition, some sensors can be fit into both the environmental or wearable category, depending on their application. An example of this are microphones; these can either be placed in the environment to detect specific sounds (e.g., running water in the bathroom, conversation in the living room), or can be worn on the body, for instance to detect heart rate through acoustics.

In Table 1.1, a list is provided of sensors that may be used in the field of human motion tracking. It should be noted however that this list is far from exhaustive; a far larger number of sensors can potentially be employed in this field, or at least to certain aspects of it. Furthermore, we considered the inclusion of implantable sensors in the table to be out of scope. The sensors in the table have been broadly categorized based on their common applications, as either 'movement and location', 'physiological parameters', or 'environment and interaction'. Some of the sensors in the table have their type listed as 'both'; this indicates that depending on the application, the sensor may be used as either a wearable sensor, or as an environmental sensor. Table 1.1 has primarily been based on the works of Bonato [2010], Lara and Labrador [2013], Logan et al. [2007], Pantelopoulos and Bourbakis [2009], Patel et al. [2012], Suryadevara and Mukhopadhyay [2012], Tao et al. [2012], Luštrek and Kaluža [2009], and Ye, Dobson, and McKeever [2012].

Note that Table 1.1 is provided here to provide some background on the common sensors and modalities one can expect to encounter in the field of human motion tracking; as this thesis focuses primarily on the application of methods and algorithms (derived from, for example, the fields of machine learning, signal processing, and data analytics) to problems of human motion tracking, a critical comparison of the abilities, advantages and disadvantages of the individual sensing modalities is out of the scope of this work.

[2]Strictly speaking, a passive RFID tag refers to the fact that the tag does not require its own power source.

Sensor	*Modality*	*Type*
Movement and location		
Accelerometer	Motion (acceleration)	Wearable
Gyroscope	Motion (rotation)	Wearable
Magnetometer	Motion (rotation)	Wearable
Inertial measurement unit (IMU)	Location, motion	Wearable
Flexible Goniometer	Motion (angular changes)	Wearable
Electromagnetic tracking system	Motion	Hybrid
Global positioning system (GPS)	Location, motion	Hybrid
Camera	Video	Environmental
Infrared sensor	Location, motion	Environmental
Ultrasound sensor	Location, motion	Environmental
Radio / WiFi beacons	Location, motion	Hybrid
Microphone	Audio	Both
Physiological parameters		
Piezoelectric strap / patch	Heart rate, respiration, motion	Wearable
Spirometer	Respiration	Wearable
Arm cuff-based monitor	Blood pressure	Wearable
Photoplethysmography (PPG)	Heart rate, respiration	Wearable
Pulse oxymeter	Oxygen saturation, heart rate	Wearable
Galvanic skin response (GSR)	Perspiration	Wearable
Electrocardiogram (ECG)	Heart rate	Wearable
Phonocardiograph	Heart sounds	Wearable
Electroencephalogram (EEG)	Brain activity	Wearable
Electromyogram (EMG)	Activity of skeletal muscles	Wearable
Glucose meter	Blood sugar levels	Wearable
Environment and interaction		
Thermometer	Temperature (skin or ambient)	Both
Hygrometer	Humidity	Environmental
Barometer / altimeter	Pressure (air)	Both
Photodetector	Light	Both
Switch sensor	Use of objects	Environmental
Motion sensor (accelerometer)	Use of objects	Environmental
RFID tag	Use of objects, location	Environmental
Current sensor	Electrical current	Environmental
Pressure sensor / mat	Use of objects, location	Environmental
Flow meter	Flow of water or gas	Environmental
CO_2 sensor	Levels of CO_2	Environmental

Table 1.1: List of common sensor modalities used in human motion tracking, including the modalities commonly measured by the sensor, the sensor type (wearable, environmental, both, or hybrid). Note that the table is not intended to provide an exhaustive list of sensors and modalities that might be used in human motion tracking.

1.4 Privacy and ethical considerations

Whenever tracking sensors are used to record data from people, it is important to consider what effects this could have on their privacy. Often, the data recorded can hold explicit or implicit clues to their identity, lifestyle, and so on. This is obvious in the case of for instance audio or video recordings, but other information such as the places someone visits (for example through GPS, the ethics of which have been discussed by Michael, McNamee, and Michael [2006]) or movement data from accelerometry can raise privacy concerns for users. Gait information, like other biometric information such as retina scans and fingerprints, can be used for identification, as described by Iwama et al. [2012], and as such can potentially be harmful to a person's privacy. Even if, for example, fears of widespread gait identification systems seem unfounded for the foreseeable future, as discussed by Boulgouris, Hatzinakos, and Plataniotis [2005], the fact that such concerns exist may impact users' willingness to adopt a certain solution or technology.

When dealing with any kind of personal or sensitive information, minimizing the impact on privacy is important, for instance by:

- Ensuring that any sensitive data is handled securely, in particular with regards to storing and transmitting of the data. This also includes removing any data which is no longer required for the application to function. Particular care must be taken with cloud storage in this respect, as it is often unclear where exactly the data is stored, and whether or not it is completely removed from the cloud.
- Processing sensitive data on the sensor platform itself. For example, if video data can be analyzed and removed, and only a person's activity information is transmitted, this reduces the impact on privacy. In some cases, privacy issues can be avoided altogether in this fashion (although this does not mean it is perceived as such by the users).

In the end, when considering the use of any system that may have an impact on privacy, it is important to weigh these concerns against the potential benefits for the users whose privacy is impacted. For instance, if a system allows elderly to live in their own home for a longer period of time, they could perceive this as a benefit that offsets a loss of privacy.

Ideally, it is the users themselves who need to decide whether the benefits outweigh the disadvantages, rather than the decision being made by a third party. In some cases, this can cause a dilemma when the users of the system are no longer able to make informed decisions themselves, as can be the case for people with dementia, for example. Such cases show that there is a need for ethical considerations and discussion in the field of human motion tracking. This also applies to those who can make informed decisions; as mentioned by McNamee [2005], the use of tracking technology can simply become a habit over time, with users no longer considering the potentially harmful effects as they become blasé about the technology's presence.

1.5 Analysis of data ordered in time

Regardless of the sensors used, the output generally consists of time-ordered data: a series of data points ordered by successive points in time. Usually, the data points are spaced at fixed time intervals, determined by the sample rate of the sensor. Techniques which are common in machine learning and pattern recognition are also often applied to this type of data - due to its time-ordered nature, techniques which are able to take the ordering into account are of particular interest here.

An important distinction here is between techniques using online processing and techniques using offline processing. In online processing, data can be included in a piece-wise fashion, and the analysis is continuously updated. For offline processing, the entire set of data over which the analysis is performed must be available before any analysis can be done. For example, the hidden Markov model and Kalman filter mentioned in Section 1.5.1 are examples of online methods, while the three algorithms mentioned in Section 1.5.2 are offline.

Prior to any analysis, there is often a process of feature extraction. Here, features are values derived from the raw signal data; these can be the raw values themselves, but also aggregate values such as the mean or variance of the raw signal values, or values obtained after a transformation of the raw signal (for example through a low-pass filter). Feature selection is often applied to reduce the impact of noise in the original signal, to remove redundant or unneeded information, or to reduce the overall dimensionality of the data. Which features are appropriate depends on the application, and feature extraction is often one of the challenges when applying the analysis of time-ordered data to a certain problem.

In this section, we discuss a number of techniques commonly used for the analysis of time-ordered data based on statistical models, machine learning, frequency analysis, and signal processing. We will briefly summarize each category before discussing them in more detail. It should be noted that the categorizations made here are primarily intended to illustrate common techniques in the analysis of time-ordered data, and as such, the different categories are not free of overlap; for example, frequency analysis is often seen as part of signal processing[3], and statistical models can employ techniques from machine learning. At the end of this section, we discuss a few notes on model transparency; that is, how easy it is to understand a model's behavior.

Statistical models. One way to take advantage of the time-ordered nature is through the use of statistical models such as (hidden) Markov models and Kalman filters. These models, also referred to as state-space models, have in common that they maintain a model state which is updated after each time step. How the model

[3]More precise names for these categories might be 'frequency and wavelet analysis' and 'time and space domain signal processing', respectively.

state changes is determined by the current state, the properties of the model, and the newly observed data point. In the case of a hidden Markov model for instance, a new state is chosen using a given transition probability distribution. This transition probability distribution depends on the current state of the model; generally, each state has its own probability distribution parameters.

State-space models can be used for regression and classification; regression refers to determining some numeric value, such as walking speed. Classification refers to distinguishing between a number of distinct possibilities, for example, between eating, walking, sleeping. Regression and classification can be achieved by observing the sequence of model states or examining the likelihood of a given model matching the sequence of time-ordered data. Another use of such models is forecasting; by generating additional model states, we can get an estimate of future data points. The forecasts are less likely to be accurate the further they are projected into the future however.

Machine learning. Machine learning is a subfield of computer science that encompasses the creation and study of algorithms that can learn from previous data (also called training data), and can then provide predictions or categorizations of new data. The statistical models described above are examples of machine learning algorithms, and as such could have been included here; due to their focus on time-ordered data however, it is worthwhile to discuss them separately in this context. While many machine learning algorithms do not share this focus, they can still be applied in the context of human motion tracking, especially when the ordering of the data is less important or otherwise accounted for. Common machine learning techniques, which we will discuss in more detail below, include neural networks, naive Bayesian classifiers, and nearest neighbor classification.

Frequency analysis. Another way to look at time-ordered data is through frequency analysis, which is often useful to find recurring patterns over time in the data. Common approaches include fast Fourier transforms (FFT) and wavelet analysis, which show the dominant frequencies of the data. Wavelet analysis is somewhat more involved than FFT, but can also show how the frequency spectrum changes at different time intervals. Frequency analysis is particularly useful when dealing with repetitive motions such as walking.

Signal processing. As we are often dealing with raw sensor data, various signal processing techniques are useful for obtaining relevant features from the data. High or low-pass filtering is often used as a first step to remove certain frequencies from the data set. This is because high or low frequencies are often related to noise, or because for example certain motions stay within the boundaries of some frequency band. Other scene analysis techniques are also useful for preprocessing or extracting features, such as obtaining the derivative of a signal or the medians for a certain window size.

1.5.1 Statistical models

Statistical models generally refer to models that describe the way data was generated through a set of probability distributions. More precisely, a statistical model is a mathematical model for which some variables do not have fixed values, but instead are given through probability distributions. Often, the parameters of these probability distributions are unknown, and have to be estimated based on the data and prior assumptions. A statistical model can also be described as a stochastic model (a model with one or more random components) that depends on a set of model parameters. As a result, a statistical model is a non-deterministic model[4]. For the modeling of data ordered in time, a type of statistical model called a state-space model is often used, with notable examples including the hidden Markov model and the Kalman filter.

Specifically, a state-space model is a mathematical model that models a process as a set of one or more process states. State-space models can be deterministic, but generally include stochastic variables defined by a set of parameters, and as such tend to fall under the umbrella of statistical models[5]. In most state-space models, both the process outputs (or observations) and the process states are modeled. The probability of a given observation is influenced by the model's current state. One of the best-known state-space models is the hidden Markov model, which uses a finite number of model states and observations, with a distinct set of observation probabilities for each state.

In short, the hidden Markov model as introduced by Rabiner [1989] consists of n distinct states $U = \{U_1, U_2, \cdots, U_n\}$, and m distinct observations $V = \{V_1, V_2, \cdots, V_m\}$, where observations are sometimes also referred to as outputs or emissions. At each discrete time step t, the model is assumed to be in one of the n possible states; the state at time t is referred to as q_t. The model state cannot directly be observed - hence the term 'hidden'. At every time step, the model changes to a new state (which may be the same as the current state), and a new observation is emitted. The probabilities of the new state and observation are determined by the transition and observation probability functions a and b respectively:

$$a_{ij} = P(q_{t+1} = U_j | q_t = U_i)$$
$$b_j(k) = P(V_k | q_t = U_j)$$

An important property of the hidden Markov model is that the transition and observation probability functions rely only on the most recent model state - this is called

[4] However, this does not necessarily mean that the modeled process must be non-deterministic as well.

[5] Admittedly, the terminology can be somewhat confusing. In short, a state-space model with stochastic variables is a type of statistical model (a stochastic model defined through parameters), which in turn in a type of mathematical model. A state-space model without stochastic variables (such as finite state machines) is a type of mathematical model, but not a statistical model - in the context of human motion tracking however, we are generally interested in the stochastic versions of the state-space model.

the Markov property. This results in computationally efficient methods which allows some influence of past observations on future model states, but may prove insufficient in situations where a longer memory is required. Examples of applications of hidden Markov models in human motion tracking include the work by Mannini and Sabatini [2012], Karaman et al. [2014], and Viard et al. [2016]. The hidden Markov model is discussed in more detail in Chapter 3.

Another common state-space model is the Kalman filter [Welch and Bishop, 1995; Meinhold and Singpurwalla, 1983]. Kalman filters are similar to hidden Markov models in the sense that they both share the concept of hidden states which are updated each time step based on the previous state, and produce observations based on their current state. Unlike the finite number of states in hidden Markov models however, a Kalman filter's hidden state can be described by a vector of real numbers, essentially allowing for an infinite number of hidden states. As a tradeoff, there is no separate probability distribution for generating observations for each state; all states use the same function for generation observations.

Given a state q and an observation y, a Kalman filter can be described as:

$$q_t = A \cdot q_{t-1} + w_{t-1}$$
$$y_t = B \cdot q_t + v_t$$

where the terms w_t and v_t represent the state and measurement noise, respectively. They are assumed to be independent and normally distributed, with covariances given by Q and R respectively. The matrices A and B have a similar role to the probability functions a and b for hidden Markov models, in that they determine how the new states and observations are generated, respectively. Applications of the Kalman filter in human motion tracking include the work by Wichit and Choksuriwong [2015], Ligorio and Sabatini [2015], and Auger et al. [2013].

Other examples of statistical models (but not state-space models) used in human motion tracking include the (Gaussian) mixture model, which uses multiple probability density functions to model the underlying statistics of the training observations, and (linear) regression models, which estimates the coefficients of a (linear) combination of features related to the desired output (or response).

In general, state-space models such as the hidden Markov model and the Kalman filter are best applied to problems where a current state will have significant impact on what the future state will be. Usually, these are problems where sequences of events follow each other in a somewhat logical (but not necessarily strict) order. In contrast, in problems where states follow each other without any pattern, or where different states cannot be distinguished, much of the advantages of such a modeling approach are lost. In these cases, it may be more beneficial to consider other machine learning methods such as naive Bayesian classifiers or support vector machines.

The reliance on a natural path through the state-space to take full advantage of the state-space models described here explains why these models have often been

applied successfully to problems in fields such as speech and video analysis; the spoken phonemes often follow a certain pattern (not every phoneme can follow any other phoneme), and similarly the position of a person in a camera recording will be constrained by how far a person can move between successive frames.

Other advantages of state-space models include that the models are fairly easy to interpret once created, and that expert knowledge about the process observed can easily be applied to the model by for example setting the number of states or by defining the initial state transition probabilities. However, state-space models also tend to have large parameter spaces that need to be estimated, which can be a problem if there is little knowledge about the modeled process available. In addition, the models are limited to some degree by the Markov property, if the current state of the modeled process depends on a sequence of previous states, rather than just the last one. There are several approaches to cope with this however, and in practice the models can often perform well even if the Markov property is not strictly adhered to.

1.5.2 Machine learning techniques

Machine learning techniques, in general, make use of past observations to derive inferences about the observed data, and as such can make predictions or classifications about any new observations. The observations used to derive inferences are often referred to as training examples, as they are used to 'train' the machine learning models. In machine learning, a distinction is often made between supervised and unsupervised learning. Supervised learning methods make use of labeled training data; that is, they require information regarding the class or desired output of each training example. In contrast, unsupervised methods do not require such input, and make classifications based on other aspects of the data, for example through similarity metrics. While the entire range of machine learning algorithms is extensive to say the least, we will here briefly discuss three of the more common (supervised) algorithms used in the field of human motion tracking. These include neural networks, naive Bayesian classification, and the nearest neighbor algorithm.

The first of these, neural networks, have previously been applied to movement pattern analysis and activity recognition [Bataineh et al., 2016; Toshev and Szegedy, 2014; Du, Wang, and Wang, 2015]. Neural networks are loosely based on the interconnected neurons in the brain; each neuron in a neural network accepts weighted inputs from other neurons, and the weighted sum of these inputs determines whether or not the neuron 'activates'. While there are many types of neural networks, as for example described by Lippmann [1987], one of the more commonly used models is the multilayer perceptron. In a multilayer perceptron, neurons use non-linear activation functions, which are often sigmoids. A commonly used sigmoid is for example

$$f(\alpha) = \frac{1}{1 + e^{-(\alpha - \theta)}}.$$

Here, $f(\alpha)$ is the output of the neuron, α is the weighted sum of inputs, and θ is an internal threshold. The idea behind the non-linear activation functions is to provide an output close to either 0 (or -1) and 1 for most values of α. Multilayer perceptrons are structured into at least three layers of neurons, with each layer of neurons providing the inputs for the neurons in next layer. Multilayer perceptrons are fully connected; that is, each neuron takes (weighted) inputs from each neuron in the previous layer. Furthermore, we can distinguish between the input layer (which represent feature values), one or more hidden layers, and the output layer (representing output values or classes), in that order. The weights of a neural network can be adjusted based on training examples through a procedure called back-propagation [Lippmann, 1987]. Apart from movement pattern analysis and activity recognition, neural networks have historically seen much use in pattern recognition applications for audio and video. Over time, other methods have been developed (such as support vector machines), although there has been renewed interest in back-propagation methods recently due to the development of deep learning.

The reason for this renewed interest is the excellent performance of deep learning methods on many machine learning problems considered to be very difficult (such as image recognition and handwriting recognition). However, at the time of writing it is still difficult to conclude whether similar advancements will be made on problems related to human motion tracking, although encouraging results exist [Ronao and Cho, 2016]. A second strength of deep learning techniques is that they inherently solve the feature selection problem; the learning process automatically determines which features are important and which can be ignored. However, deep learning still suffers from the classic neural network problem that the resulting model is a virtual black box (see Section 1.5.5). In addition, deep learning methods currently require specialized hardware for most problems due to their computational demands, and generally require large sets of data to achieve their state of the art performance.

The second method discussed here, the naive Bayesian classifier, described by for example McCallum and Nigam [1998], is a relatively straight-forward probabilistic classifier based on Bayes' theorem. Applications of the naive Bayesian classifier to human motion monitoring include the work of Urwyler et al. [2015], Valle, Varas, and Ruz [2012], and Preis et al. [2012]. The 'naive' part of the name refers to the strong assumption that each feature is conditionally independent. This assumption rarely, if ever, holds in practice. However, the naive Bayesian classifier often performs well on real-world problems despite this assumption.

Using Bayes' theorem, the naive Bayesian classifier estimates the conditional probability $P(C_i|x)$ of a certain output or class $C_i \in C$ given a set of input features denoted by x as

$$P(C_i|x) = \frac{P(C_i)P(x|C_i)}{P(x)},$$

where $P(C_i)$ is the prior probability of C_i, $P(x|C_i)$ is the probability of feature set x occurring for C_i, and $P(x)$ is the prior probability of x. In practice, the denominator $P(x)$ is often omitted, as we are generally interested in the relative probabilities between outputs or classes, and since $P(x)$ does not depend on C, the term becomes constant.

The strength of the naive Bayesian classifier is that it is both simple and powerful [Mitchell, Monaghan, and O'Connor, 2013]; even if features are not conditionally independent, the method often produces good results. As a result, the model is widely applied in a variety of contexts. Even so, naive Bayesian classifiers may not be the best choice when there are strong interactions between the model features.

The third method, the nearest neighbor algorithm, is arguably one of the simplest classifiers in machine learning: a new observation is assigned the class of its closest previously observed neighbor. Determining which previous observation is the closest is generally based on the Euclidian (or L2) distance, although other distance metrics such as city block (or L1) distance or Hamming distance can be used. Often, the more general version of the nearest neighbor algorithm is used, the k-nearest neighbor algorithm. Here, the class is assigned based on the k nearest neighbors, usually through a majority vote[6].

The nearest neighbor algorithm is easy to implement and can fit complex problem spaces, but does have some disadvantages. These include being dependent on the scaling of the various features, the requirement to retain all previous observations, and potentially high computational requirements for high numbers of past observations and features [Kaghyan and Sarukhanyan, 2012]. Even so, the algorithm can perform well in practice, as has been successfully applied to human motion tracking in for instance the work by Vögele, Krüger, and Klein [2014], Gupta and Dallas [2014], Kaghyan and Sarukhanyan [2012], and Mezghani et al. [2008].

Other machine learning algorithms of note in the field of human motion tracking include support vector machines (SVM) and decision trees, keeping in mind that many more algorithms exist that can be of use. Support vector machines achieve non-linear separation of a feature space by mapping the feature space to a higher dimensionality, and making a linear separation in this hyperspace. Decision trees construct a number of binary decision rules based on the data features, with each decision for a new observation leading either to the 'left' or 'right' of the tree, until a final node (called a leaf) is reached which assigns its class to the observation.

[6] For $k > 1$, this does introduce the additional problem of having to deal with tied votes. Fortunately, (or perhaps unfortunately) there are many solutions to break ties; for example, selecting a class at random, reverting back to 1-nearest neighbor, or basing the decision on the average distances for each class. Of course for binary classes, the issue of ties can simply be avoided by choosing k as odd.

1.5.3 Frequency analysis

One of the most common tools for frequency analysis (also referred to as spectral analysis) is the fast Fourier transform [Cooley and Tukey, 1965; Cochran et al., 1967]. The fast Fourier transform describes a set of algorithms for transforming discrete data in the time domain to the frequency domain - this is particularly useful for discovering periodic components in the sensor data, and in showing the relative strengths of these components. In motion analysis, periodic elements are fairly frequent; examples include the footsteps in walking, the days of the week in levels of physical activity, and so on.

One of the downsides of Fourier transform is that the transformation to the frequency domain removes any temporal resolution from the data. For example, if a sequence of periodic measurements slowly changes its frequency over time, this is not reflected in the frequency domain. This is one of the shortcomings that another popular frequency analysis technique, wavelet transform, attempts to address, as for example described by Torrence and Compo [1998]. Here, the signal is multiplied by a wavelet, a wave-like oscillating function, at multiple scales and transposes. As a result, frequency information can be displayed in multiple time intervals, at multiple frequency bands. Due to the nature of the transforms, lower frequencies have higher frequency resolution, but poorer time resolution, compared to higher frequencies. The results of the wavelet transform also depend on the choice of wavelet (for which there are many options), and which wavelet to choose is not always obvious.

1.5.4 Signal processing

Signal processing techniques are often used as a preprocessing step, or in other words, for cleaning up the (sensor) data. In particular, filtering techniques are often applied, with high-pass and low-pass filters being arguably the most common. These types of filters are used to remove unwanted frequency components from sensor data; as the names suggest, high-pass filters only allow frequencies above a certain threshold to pass, and low-pass filters only allow frequencies below a certain threshold to pass. Combining the properties of a high-pass and low-pass filter yields a band-pass filter, which removes all frequencies not within a certain band. In practice, filters are not ideal, and as such settle for suppressing certain frequencies, while attenuating others. Apart from band-pass type filters, other filters can for example include derivative filters, which estimate a derivative of a signal, or squaring filters, which return the signal squared in a point-wise fashion. Filters are processed in an online fashion; that is, signal values are added to the filter sequentially, ordered in time.

A major distinction in filters is between finite impulse response (FIR) and infinite impulse response (IIR) filters. The difference between these two types of filters is that for a certain value (or impulse) entering the filter, that particular value will eventually have no impact on the filter output (or response) in the case of a FIR filter, while

that value will always have some impact on the outcome of the filter in the case of an IIR filter (although the impact may asymptotically approach zero over time). FIR filters can generally be implemented through convolution, as further discussed in Section 5.2.3. Sliding window filters such as moving window integration are an example of FIR filters. Examples of IIR filters include the moving average filter, and the Kalman filter[7] discussed in Section 1.5.1.

1.5.5 Notes on model transparency

An often not discussed aspect of various modeling techniques is that of model transparency. Model transparency refers to the ease with which one can observe the behavior or inner workings of a model - in other words, how easy it is to 'see what the model is doing'. There are several benefits to having a high model transparency: first, it becomes more straightforward to determine the model's behavior, for example by observing which aspects are modeled incorrectly and why. Second, one might have more confidence in a model if one can reason about why it makes certain decisions based on its inner workings.

In health-related applications model transparency can be particularly important for this reason; for example, if a model indicates that a person might not be able to live alone unassisted anymore, healthcare professionals and caregivers might expect some explanation as to why the model believes this to be the case. While human adjudication should always be applied when making such decisions, caregivers might be more receptive to review the current situation if the model can provide a solid reason as to why they should do so. In many cases, the acceptance of such a model could be dependent on the level of transparency that the model can provide.

Transparency can vary considerably between different modeling techniques. Neural networks, in particular, are well-known for being highly opaque (or as it is often called, being a black box). While it is technically possible to examine the weights of the individual neurons, the fact that every neuron in each layer is connected to every neuron in the next layer makes it highly complicated to derive the model's inner workings. In contrast, linear regression is often seen as a highly transparent model, as the coefficient values directly indicate the contributions of the individual features to the overall response of the model. A hidden Markov model could be argued to be somewhere in the middle - while it consists partly of hidden states, the model can only be in one state at a time, making the modeling process much easier to examine compared to the interconnected neural networks.

In all, each modeling technique can be argued to provide a certain level of transparency, based on the modeling approach used and the envisioned complexity of the model (such as the number of hidden states in a hidden Markov model). For any

[7] The Kalman filter, as the name suggests, can act as an averaging or smoothing filter. As the Kalman filter is generally used for process modeling, however, it is more appropriately listed as a statistical modeling technique.

given application, it is important to consider how much transparency would be expected from a model, and how much a prospective modeling technique could provide.

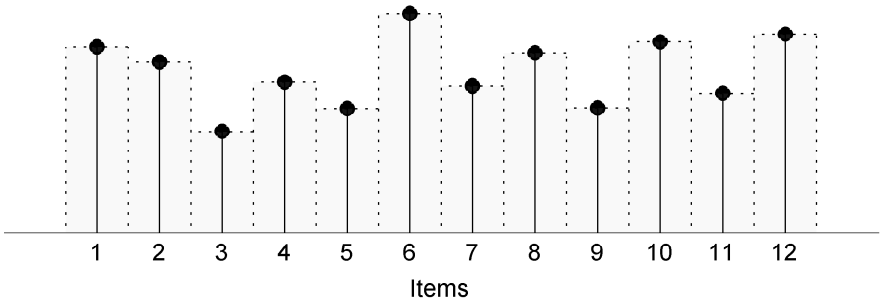
1.6 A model for motion tracking

In this section we introduce a model for analyzing human motion over time. More generally, this model can be applied to analysis problems in other fields as well, provided they also involve data with a time component. Making use of this model, three common problems are outlined below: segmentation, classification and regression. Segmentation refers to dividing a sequence of data into a number of smaller segments based on some criterion, while classification involves assigning a sequence of data to one of a number of distinct classes, and regression involves assigning a continuous (often real-valued) value to a sequence. These problems are described in more detail below.

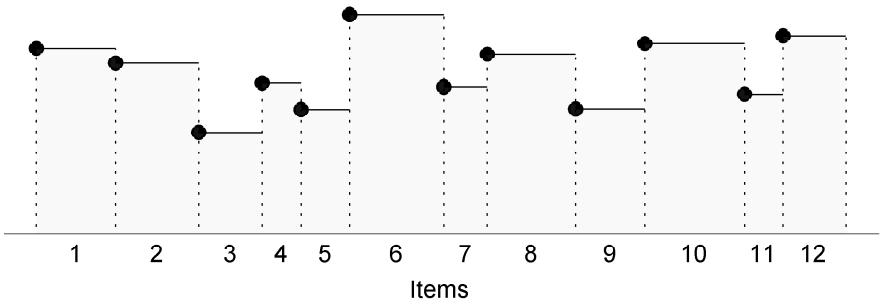
Before introducing the model, we first briefly discuss how sensors record data over time. In the majority of cases, sensors produce a series of discrete measurements as a representation of a continuous process. That is, the actual process measured by the sensor can be represented as some continuous function, yet the sensor itself only records samples of this function at certain intervals. An example is an accelerometer; acceleration continuously acts on the sensor, but is only recorded in a finite number of samples each time interval, determined by the sensor's sample rate. How well this discrete set of samples captures the original process depends on both the process itself and the sample rate - as is well known, any spectral components higher than the Nyquist frequency are likely to cause aliasing and loss of information in the discrete signal [Ifeachor and Jervis, 2002].

While the values of the actual process we are measuring are unknown to us outside of the discrete measurement samples that were recorded, and barring any knowledge or assumptions regarding the process itself, the best estimate of the actual value at an arbitrary time is the value of the closest measurement. As a result, we can view a series of measurements as a sequence of items with a certain value and duration. If the sampling interval is fixed (which it commonly is), and the measurements are equidistant, the durations for each item will be the same. An example of such a sequence of equidistant measurements is shown in Figure 1.1a.

An alternative approach to sampling sensor data is to only record a measurement when the value of the actual process changes. This is essentially an event-based approach; a measurement is recorded whenever the state of the sensor is changed. Clearly, this is not practical for a continuous process, as values tend to differ for even the smallest of intervals, but this approach can be applied to processes of a binary or categorical nature. An example is a simple contact switch: rather than reporting if the switch is pressed at every time interval, a recording can be made every time the switch is pressed or released. These type of measurements also fit with the idea of a



(a) Sequence of discrete measurement samples with a fixed sampling interval.



(b) Sequence of measurement samples obtained through an event-based sampling approach.

Figure 1.1: Examples of measurement sample sequences obtained using either a fixed sampling interval, or an event-based sampling approach.

sequence of items with a certain value and duration, the duration being determined by the time until the next event. An example of such a sequence is shown in Figure 1.1b.

Formally, we can describe a series of measurements as a sequence S of items $i = 1, 2, \ldots$, where each item consists of a duration w_i and a value h_i, with $w_i > 0$ for all i. Here, h_i can be a scalar, or a vector of measurements that were recorded simultaneously. Note that sequences of items need not necessarily represent sequences of raw measurement values, but may represent a sequence of one or more features derived from the original measurements. In theory, a sensor can keep recording indefinitely, resulting in an infinite number of items. In practice, data is recorded for only a certain amount of time, or we are only interested in a certain part of the data that is recorded. We define such a part of the measurement data as a *segment*.

Definition 1.1 (Segment). A segment j of a sequence S of items $i = 1, 2, \ldots$, where an item i consists of a duration w_i and value or height h_i, is defined as the interval $\sigma_j = [s_j, e_j)$. Any item i for which $[\sum_{l=1}^{i-1} w_l, \sum_{l=1}^{i} w_l) \cap \sigma_j \neq \emptyset$ is at least *partially covered* by the segment j. An item is considered *fully covered* by a segment j if it is entirely within the interval σ_j. If the boundaries s_j and e_j of a segment both coincide with item boundaries, the segment is said to be *aligned*. □

In most cases, it is convenient to choose segments such that they are aligned, that is, items are either fully inside the segment, or fully outside of it. Segments cannot always be selected in such a manner however, especially when segments are generated through some automated process, such as in the case where segments last exactly one day. In such cases, one possibility is to split items into multiples with new durations and the same value, so that for the new set of items, the alignment requirement is preserved.

Often, we want to select segments such that they provide some meaningful division of the data with regards to the problem we wish to analyze. For example, when counting the number of footsteps of a person over the day, we may wish to partition the data into segments of walking data, and segments containing other activities. If there is no ground truth available to determine these segments, we can turn to machine learning techniques to determine these data segments. In general, we want to solve the problem of finding a segmentation of our data set which is optimal with regards to some error criterion.

Problem 1 (Segmentation problem). Given a sequence S of items $i = 1, \ldots, n$, find a partitioning of the items i in S into non-overlapping, contiguous segments $j = 1, 2, \ldots$ that completely cover all items, such that $\sum_j E_j$ is minimal, where E_j is an error criterion for segment j. □

The nature of the error criterion depends on the application. A very simple example of an error criterion is based on the L2 norm; if we let μ_j be the weighted mean of the items in segment j, the L2 error is given by $\sum_{i \in j} w_i (h_i - \mu_j)^2$. A definition for the more general Lp error criterion is given in Section 2.2. For the sake of convenience, we assume here that segment j is aligned. In practice, the error criterion can be much more complicated, and can be related to a probability estimate or the fit of a model. It will therefore not always be possible to find the optimal segmentation in reasonable time. In this case, approximation techniques can be considered. A common variation of the segmentation problem is the k-segmentation problem, which is discussed in more detail in Chapter 2. Here, the aim is to partition S into exactly k segments.

A second problem is the classification problem, where we want to assign one of a distinct set of classes $C = \{c_1, \ldots, c_l\}$ to a segment. For example, in human activity recognition, classes can be eating, sleeping, watching television, and so on. To assign classes to segments of data, we often create a function or inference model I

which derives a probability or confidence for each class in C based on a segment of data σ_j, given by $p(c_i|\sigma_j) = I(\sigma_j)$. The inference model is often created through machine learning algorithms, generally by training the model using a separate data set. Many algorithms do not provide a probability measure for each class directly, although some, like hidden Markov models, do. In most cases though, it is possible to adapt these algorithms to provide a probability or confidence measure.

Usually, the inference model does not operate on the raw data directly, but rather on a set of features derived from the data. In this case, we can represent the classification process as $p(c_i|X) = I(X)$, where X is the set of features derived from segment j. The behavior of an inference model is often controlled by a set of model parameters, ρ. These parameters are sometimes included in the classification model, yielding $p(c_i|X,\rho) = I(X,\rho)$, although like above, they are often left out of the equation and assumed implicitly.

The estimated class y_j that is assigned to a data segment is generally determined as the class with the highest probability score, $y_j = \arg\max_{c_i} p(c_i|\sigma_j)$. The aim is to find an inference model such that $y_j = C_j$, where C_j is the actual class of segment j.

Problem 2 (Classification problem). Given a set Σ of one or more data segments $\sigma_j = [s_j, e_j)$ with class $C_j \in \{c_1, \ldots, c_l\}$, find an inference model I such that the estimated class y_j of σ_j, given by $y_j = \arg\max_{c_i} p(c_i|\sigma_j)$ and $p(c_i|\sigma_j) = I(\sigma_j)$ is the same as the actual segment class, $y_j = C_j$, for all segments $\sigma_j \in \Sigma$. □

The problems of classification and segmentation often occur together as a combined problem; for example, when we want to find segments of walking data, we want to find a segmentation of the data based on a classification outcome. Here, the segmentation error criterion can be directly linked to the probability metric of the inference model. In practice, it is generally not possible outside of trivial cases to find an inference model that correctly classifies every possible data segment. Some accuracy measure is therefore often established as an indicator of the performance of a given inference model. This can simply be based on the number of errors made by the inference model on a given set of data, but more complicated metrics exist as well.

To obtain measures of accuracy representative of the likelihood of correctly classifying a previously unseen data instance[8], the training set / test set paradigm is generally used. Here, the available data is divided into a training set that is used to build the model, and a test set that is used to evaluate the model performance, where the training set and test set do not overlap. A further alternative, building on the training set / test set paradigm is n-fold cross validation, where the data is divided into n (roughly equal-sized) non-overlapping folds. In n rounds, each of the folds is used as

[8] In other words, a measure of accuracy that is not biased due to possible overfitting of the data.

a test set, with the remaining folds forming a training set. The n-fold cross validation accuracy is then computed as the weighted average of the accuracies of the individual rounds.

The third problem, regression, is similar to the classification problem, but differs in that where classification aims to find one of a distinct set of classes, the regression problem attempts to assign a continuous numerical, or at least ordinal, value based on a sequence of data[9]. This value can be anything from walking distance or heart rate, to the health of a person expressed in a (set of) numerical value(s). Unlike classification, where there are a finite number of classes to choose from, in regression there can potentially be an infinite number of regression values. The general regression problem can be divided into the specific cases of time-point regression, where a value is assigned to each time $t \in T$, and segment regression, where a value is assigned to each segment.

For a regression model F and segment j, we can derive a value z_j through $z_j = F(\sigma_j)$. Similar to classification, features are often derived first from the raw data to use in the regression model. For regression, machine learning techniques are often used as is the case for classification, although regression usually requires a different set of algorithms. Examples of common regression techniques include simple linear regression and Kalman filters.

Problem 3 (Regression problem). Given a set Σ of one or more data segments $\sigma_j = [s_j, e_j)$ with property Z_j, find a regression model F with $z_j = F(\sigma_j)$ such that $z_j = Z_j$, for all segments $\sigma_j \in \Sigma$. □

As is the case for the classification problem, finding a regression model where this holds for all segments is rarely possible in practice. Usually, some error criterion is used to determine the performance of a regression mode, such as the least squares method.

1.7 Challenges of human motion tracking

In this section, we discuss a number of the challenges commonly faced when attempting to address a problem in the field of human motion tracking.

Human variability. In the field of human motion tracking, arguably one of the biggest challenges stems from the source of our measurements. Different people will often perform the same task in a different manner, and even for the same individual, performing the same task twice hardly guarantees an identical result. As a consequence, measurements obtained from tracking humans are often subject to a large

[9] Although classification and regression are related, it is worthwhile to point out that they are both fundamentally different problems: regression is concerned with estimation of values on a continuous (or ordered) scale, while classification is concerned with membership of one of a set of nominal classes. As such, the regression problem is not a generalization of the classification problem.

amount of variance. As discussed by Stergiou and Decker [2011], this variability is present in all biological systems. Rather than being the result of movement errors, a certain level of variance avoids movements becoming too rigid or too chaotic.

Apart from the natural variability in how actions are performed, we may also see variation due to the wearing position of a sensor (for example, a change of pants can influence the measurements obtained from a mobile phone), or due to changes in habits of a person over time. Naturally, changes in a person's health or even mood can also affect how actions are performed, and can be an additional source of errors over time.

In terms of variability when tracking humans performing actions, Sheikh, Sheikh, and Shah [2005] identify (in the context of camera images) three important sources: viewpoint (camera position with regard to the scene), execution rate (speed of performing an activity), and anthropometry (personal characteristics such as height or gender). While these apply specifically to camera recordings, we can generalize these sources of variance by extending the 'viewpoint' to include sensor (wearing) position and orientation, and extending 'execution rate' to include not only the speed, but also the order in which actions are performed (which may include components that are entirely missing in some cases). When extended as such, these three sources arguably capture much of the variation observed when tracking human motion.

As a result, it is important to employ methods that are robust to high levels of variation in the data observed. One of the benefits of the type of methods described in Section 1.5 is their ability to generalize from observed data. Methods that fail when a strict ordering of events is not adhered to, such as finite state machines, are generally not recommended for applications in this field, at least not without some additional reasoning to cope with an unexpected sequence of events.

Due to the large amounts of variability we can expect in our measurements, it is generally not possible to create an application that is 100% accurate in tracking a user's movements. As such, it is important to consider the cost of failure of an application, or in other words, the consequences of misclassification. For example, a vital signs monitoring system that calls emergency services every other day due to misclassification is unlikely to be accepted by its users. In contrast, a step counter that misses a few steps each day is likely to be acceptable for all but the most demanding of applications.

However, the cost of failure is not simply an equation based on the accuracy of the application in question; measures can be taken to mitigate these costs. For example, the vital signs monitor mentioned could include a wearing detection module that will prevent alarms when the monitor is not worn. In addition, the monitor could include a button that allows the user to cancel an alarm for some time prior to the actual alarm being created.

Changes in conditions or environments. When recording data over time, the conditions in which the measurements are recorded may vary or change. This may include changes in lighting conditions over the day, pieces of furniture being moved inside the home, or changes in the ambient temperature due to changes in season. Any system that is to be used in practice needs to be able to cope with such changes, either automatically or through some assisted recalibration procedure. In addition, systems that are intended to be used within a certain environment (such as a person's home) need to take into account that these environments may differ from user to user. This is generally less of a problem when employing wearable sensors, but is often the reason that systems based on environmental sensors require specific installation or calibration procedures to ensure that the system will work properly (or at all).

To deal with changes in conditions or environments, several techniques have been devised over the years. Specifically, the issue of changing lighting conditions is one that is very prevalent in computer vision applications. A common technique in this field is the use of background subtraction [Sorbral and Vacavant, 2014; Godbehere, Matsukawa, and Goldberg, 2012]; distinguishing between moving objects (foreground) as opposed to a non-moving background. By maintaining a background model that can be adjusted over time, changes in lighting conditions can be compensated for. As discussed by Sorbral and Vacavant [2014], there is a multitude of algorithms available to perform background subtraction.

Outside of the field of computer vision, the counterpart to background subtraction is sometimes referred to as a baseline model. The aim of a baseline model is to learn over time the measurements normally observed during some neutral state, often when the user is at rest. Often, this baseline is modeled as some probability distribution of measurement values [Kocielnik et al., 2015].

Missing data. Due to the challenges listed above, we can often find ourselves in a situation where the recorded measurements contain missing data. Missing data can occur in human motion tracking for a number of reasons, including human error (forgetting to wear a sensor or to switch on the system), actions being performed outside of the sensor's range of view, or artifacts due to motion or loss of contact with the skin (particularly for sensors measuring physiological signals).

The problem of missing data is not unique to the field of human motion tracking, and as such, a number of approaches exist [Saar-Tsechansky and Provost, 2007]. However, there seems to be no single best approach overall, and as a result the choice of how to handle missing data is often problem-dependent. The most common approaches are deletion (ignore data with missing measurements), imputation (replace missing values with new values, often chosen through some distribution), and the use of algorithms that are robust to missing data (for example, ensemble methods consisting of multiple partial models).

Taking advantage of the fact that human motion data is generally ordered in time, missing measurements can also be approached as a filtering problem; we can impute the missing measurements with a value based on measurements close to the missing data point in time. Naturally, this will only work if the missing data points are distributed somewhat randomly timewise; if there are no actual measurements close in time, this approach is unlikely to succeed.

Feature selection. Another challenge not unique to the field of human motion tracking, yet often encountered, is that of feature selection (also called feature engineering). Particularly when trying to assess higher-level activities, it is often not clear what measurement features to use; generally, the inclusion of too many features in a classification or regression method will lead to overfitting and reduced performance [Gheyas and Smith, 2010].

There are numerous methods to help determine the predictive power of a given feature; these include statistical testing, measuring information gain, analysis of model coefficients or weights, automated feature selection methods such as stepwise selection for linear regression, or dimensionality reduction methods such as principal component analysis [van der Maaten, Postma, and van den Herik, 2009]. As all of these methods are by necessity based on some set of prior assumptions however, they generally only provide a part of the whole picture. It is therefore recommended to try several of such methods when taking this approach[10].

Another approach is to select a classification or regression algorithm that automatically performs feature selection as part of its learning phase. Examples of such algorithms include LASSO regression, decision tree-based methods such as random forests and extreme gradient boosting, and deep learning. The advantage of these methods is that the feature selection problem can be avoided. However, there is a price to pay for this, often in terms of requiring more data points to achieve comparable results to other methods.

Other challenges. Other challenges that are often encountered in the field of human motion tracking include distinguishing between the people (and sometimes pets), challenges related to user adherence, and user acceptance due to obtrusiveness or privacy. The challenge of distinguishing between people is particularly prevalent for applications of environmental sensors, where it is often difficult to distinguish between the user of the application and other people such as family members. Common solutions include tags or devices worn by the users to identify them to the system, or the use of computer vision techniques to identify a user. In addition, designing the application for a specific population (such as elderly living by themselves) can help in mitigating this challenge as well.

[10] For example, in Chapter 4, we employ information gain, statistical analysis, and genetic programming for feature selection.

When using an application over a longer period of time where a degree of user input is required, adherence can become a challenge, as users may provide input less frequently over time, or stop doing so altogether. The topic of user adherence is a field of its own, and as such a full discussion on how to keep users engaged is out of the scope of this thesis.

Similarly, when designing any application in the field of human motion tracking, where we often measure personal information, or place sensors in a personal space, user acceptance due to privacy or sensor obtrusiveness should be considered. As discussed in Section 1.4, it is important for user acceptance that the perceived benefits outweigh the burdens placed on the user by the system. As with the topic of adherence, a full discussion of user acceptance strategies is beyond the scope of this introduction, as these topics are too broad to be captured in a few paragraphs.

1.8 Thesis overview

The remainder of this thesis consists of two parts: the first part concerns the theory of human motion tracking, specifically on the problem of the segmentation of time series, as introduced in Section 1.6. This part is discussed in Chapter 2, where we expand upon the model of Section 1.6, and show various properties of optimal segmentations. The second part of the thesis consists of a number of applications of human motion tracking, which are discussed in Chapters 3, 4, and 5. The applications considered in these chapters are: the recognition of activities of daily living (ADL), prediction of dropout in a lifestyle physical activity program, and cadence estimation. While we will introduce the individual applications in more detail below, they all share the common thread of measuring and tracking the users' movement through the environment, and making an interpretation of their behavior or state based on the measurements obtained. In the remainder of this section, we will give a brief introduction to each of the chapters. All studies described in these chapters (and by extension, in this thesis) have been performed in accordance with the ethical standards and procedures within Philips Research.

In Chapter 2, we expand upon the model introduced in Section 1.6 and examine the problem of segmentation of time series. Specifically, the k-segmentation problem is discussed; this is the problem of segmenting a sequence S into exactly k non-overlapping, contiguous segments, such that the total error of a given error criterion is minimized. The chapter includes a formal definition of the k-segmentation problem, and the dynamic programming algorithm for solving it in polynomial time under the condition that all segment boundaries coincide with item boundaries; the alignment requirement. We will prove in Chapter 2 that even if the alignment requirement does not hold, the dynamic programming algorithm can solve the k-segmentation problem optimally in polynomial time for the Lp error criteria. In addition, we further refine this result by showing that for the L1 and L2 error criteria, the segmentation error

as a function of the coverage of an item contains at most a single maximum, and no other stationary points.

In Chapter 3, the first research question is addressed: can ADLs be unobtrusively tracked and recognized? ADLs such as eating, cleaning, and so on, play an important role in self-care. As such, whether a person can perform ADLs successfully can be a strong indicator of their ability to live independently. In this chapter, a method is presented for tracking ADLs in a kitchen environment using information from a single camera and microphone mounted on the ceiling. Hidden Markov models are introduced as a means to detect ADLs by modeling each activity as a separate hidden Markov model. We then examine the effectiveness of this approach using data recorded in a home-like setting, and investigate the benefits of using a combination of video and audio data, compared to using only a single modality.

The second research question is discussed in Chapter 4; can analyzing the behavior of people who try to be more physically active help predict if they will drop out of a lifestyle physical activity program? In this chapter, we make use of a large database of participants of a lifestyle physical activity program to predict which participants are likely to drop out of a twelve-week program before the program is complete. The lifestyle physical activity program aims to slowly increase the daily levels of physical activity of the participants over twelve weeks by offering feedback using a wearable activity monitor, and coaching support. We examine how well the database features are able to distinguish between adherent participants and dropouts, and introduce a genetic programming approach towards classification of dropouts one week prior to their dropout day.

Chapter 5 addresses the topic of cadence estimation using a wearable accelerometer; that is, the estimation of the number of steps per minute that a person makes while walking. Cadence is an important characteristic of a person's gait; changes in gait can often be indicative of injury or disease, and as such can provide important information regarding a person's health. In addition, cadence estimation is highly related to step detection, the cornerstone of many pedometers. In this chapter, we aim to address the final research question: is it possible to determine (psycho)motor skills such as gait accurately using wearable sensors? To do so, four cadence estimation algorithms for a wearable tri-axial accelerometer are investigated, for both a wrist-worn position and a pendant (torso) position. The algorithms are investigated at a variety of different walking speeds, both while walking on a treadmill and under free-walking conditions.

Finally, a summary of this thesis and the results that have been obtained are provided in Chapter 6.

2

Sequence segmentation

2.1 Introduction

In this chapter, we address the first component of the model for human motion tracking described in the main research question, and defined in Section 1.6; the segmentation of time ordered data into meaningful segments. Often, segmentation is required to separate the different types of activities within the continuous stream of time-ordered sensor measurements, or to distinguish between activities of interest and the background activities that are outside of the application scope. While some algorithms can perform the segmentation step implicitly, in many cases a segmentation step is required. For small data sets segmentation can be performed manually, but for many data sets, we have to rely on algorithmic methods such as described here.

The topics discussed here can be applied to most types of time ordered data, not just those regarding human movement. Examples include the analysis of DNA or protein sequences in the field of genomics [Azad et al., 2002; Churchill, 1989; Gwadera, Gionis, and Mannila, 2006], and various data mining applications such as text segmentation [Ge, Pratt, and Smyth, 1999] and weather prediction [Gedikli, Aksoy, Unal, and Kehagias, 2010; Kehagias, Nidelkou, and Petridis, 2006]. Ideally, a good segmentation should capture segments that are of particular interest for a given application, such as segments that match a certain activity, a certain time of day, and so on.

When collecting data for longer periods of time, it can in many cases be expected that the state of the person (or object) being recorded can change over time. Here, the state refers to a set of contexts that are relevant to the intended subject of analysis. For example, when the subject of analysis is measuring a person's gait (e.g., walking speed), their state may reflect whether they are walking, running or sitting down. Often, it is desirable that segment boundaries coincide with changes in state. In some cases, only measurements taken during particular states will be of interest, or measurements will be subject to different interpretations depending on the underlying state. In other cases, the fact that a state change has taken place may be relevant in itself.

As such, it is often not just the subject of the analysis or interpretation of time ordered measurement data that is important (e.g., how fast was a person walking?), but also finding the start (and end) of relevant data segments (e.g., when were they walking?). This is especially true if prior knowledge of the state the person is in at a given time is unavailable, or if it is unclear what states would be relevant to the question at hand. The problem of finding appropriate segments in a series of data is known as the segmentation problem, sometimes also referred to as change-point detection.

The k-segmentation problem can be seen as a specific case of the more general segmentation problem. In particular, the k-segmentation problem seeks to optimally partition a sequence of items into k separate segments - a formal definition is provided in Section 2.2. Both the segmentation problem and the k-segmentation problem have been widely discussed in the literature [Terzi, 2006; Keogh et al., 2001; Bingham, 2010; Lovric, Milanovic, and Stamenkovic, 2014]. The appropriateness of a given k-segmentation can be determined by defining an error criterion based on a representative of the given segment (e.g., the segment mean) and the items that make up the segment. A common example is the L2 error criterion, where the error is determined as the cumulative Euclidian distance between the segment items and the segment mean. A k-segmentation which minimizes the total error of all segments is called an optimal k-segmentation.

It can be argued that the segmentation problem is related to (or even a special case of) the well-known clustering problem. In the clustering problem, a set of items is partitioned in such a way that items with similar values are grouped together, according to some similarity criterion. In the segmentation problem, it can be argued that a similar partition is made, based not only on the similarity of the items themselves, but also on their temporal similarity. A similar comparison is made by Aghabozorgi, Shirkhorshidi, and Teh [2015], where the authors refer to this type of clustering as time point clustering, although they note that in contrast to segmentation, not all items need to be assigned to clusters in time point clustering.

Broadly speaking, solutions to the k-segmentation problem can be categorized as either approximate solutions, or as exact solutions, the latter generally involving a dynamic programming approach. Bingham [2010] provides a good overview of many solutions and variations to the segmentation problem. For the general segmentation problem, a further distinction can be made between online and offline algorithms [Keogh et al., 2001]. Here, online algorithms deal with the segmentation of a stream of data, while offline algorithms operate on a full, finite-length sequence. As finding an appropriate k-segmentation is challenging when the sequence length is unknown, the k-segmentation problem generally deals with offline sequence segmentation.

Solutions to the k-segmentation problem. As was (to the best of our knowledge) first described by Bellman [1961], an exact solution to the k-segmentation problem can be found in $O(n^2k)$ time for a sequence containing n items of unit duration[1]. While this is sometimes seen as insufficient for, for instance, data mining problems[2], a number of suggestions for improvements to the running time of dynamic programming approaches have been made, while maintaining the optimality of the resulting k-segmentation [Kehagias, Nidelkou, and Petridis, 2006; Gedikli et al., 2010; Tatti, 2013]. This is usually achieved through pruning techniques; that is, paths in the dynamic programming scheme that lead to sub-optimal solutions are identified and abandoned before they are fully explored.

Approximate solutions to the k-segmentation problem are generally used when the $O(n^2k)$ time complexity of the exact method is too big a limitation for the problem at hand, and a lower running time is required; often this is the case when the length or the number of sequences to be processed is extremely large. Approximate solutions often find applications in the fields of data mining and big data analytics as a result. Common approximate solutions are the top-down approach where large segments are repeatedly split in a greedy fashion, or the bottom-up approach where initially each point is its own segment, and segments are merged greedily [Keogh et al., 2001; Lovric, Milanovic, and Stamenkovic, 2014]. The top-down algorithm works by initially considering the entire sequence as a single segment, and considering every possible segmentation boundary, selecting the segmentation boundary which minimizes the total error. This process continues recursively until k segments are found. The bottom-up algorithm works similarly, by recursively merging single-item segments.

[1] The notation used in $O(n^2k)$ is called the 'big O' notation (described by for example Jones and Pevzner [2004]), and in short, indicates an upper bound on the degree of growth of the computational load of a function, based on its inputs. In the case of $O(n^2k)$, the computational load grows at most quadratically with the number of items n in a sequence, and linearly with the number of segments k.

[2] In data mining applications, segmentation is often applied to find more compact representations of long sequences, for example through linear interpolation or regression of the segment items.

Other approximation algorithms include the DNS (Divide & Segment) algorithm presented by Terzi and Tsaparas [2006], for which the authors provide an approximation error bound. In order to improve on the speed of the dynamic programming algorithm, the DNS algorithm aims to divide the segmentation problem into smaller sub-problems, and to solve these optimally using dynamic programming. As such, the sequence is first partitioned into a number of subsequences, and for each, a k-segmentation and a set of k segment representatives weighted by segment length is computed using dynamic programming. A final k-segmentation of the original sequence is then obtained by using the dynamic programming procedure on the concatenated subsequence representatives. The running time of the DNS algorithm is $O(n^{4/3}k^{5/3})$.

Variations of the k-segmentation problem. A special case of the k-segmentation problem is referred to as the k,h-segmentation problem. Here, the k segments may have at most h (with $h \leq k$) different representations, such as the segment mean or median. The k,h-segmentation problem was introduced by Gionis and Mannila [2003], where it was presented along with a number of approximation algorithms for the k,h-segmentation problem. It can be argued that the k,h-segmentation problem is a more appropriate representation in case of the classification problem, as described in Chapter 1, where each of the k segments should be classified as one of h possible classes. Alternatively, as has been remarked by Terzi [2006], the k,h-segmentation problem can be interpreted as a hidden Markov model discovery problem for a model with h hidden states, and a total of $k-1$ state changes for a given sequence of items.

Rather than segmentation on sequences of items with continuous or discrete numeric values, the segmentation problem has also been applied to symbolic sequences. In symbolic sequences, item values are represented by symbols such as letters or ordinals. State-space models, such as hidden Markov models or Markov chains, are often applied to find segmentations. As noted above, this can be interpreted as solving a k,h-segmentation problem, or a more general h-segmentation problem when the number of segments is not defined in advance. In this case, h is given by the number of states in the state-space model. Examples of such approaches to the segmentation of symbolic sequences include the work of Ge, Pratt, and Smyth [1999], who use a hidden Markov model to segment Chinese written text, and Gwadera, Gionis, and Mannila [2006], who describe the use of tree models (variable length Markov chains) for symbolic sequence segmentation. Other applications include clustering approaches to find common events or contexts [Flanagan, Mantyjarvi, and Himberg, 2002; Mannila, Toivonen, and Verkamo, 1997], or the segmentation of DNA sequences into genomes or other regions of significance [Azad et al., 2002; Churchill, 1989].

Another variant of the segmentation problem is the monotonic or unimodal segmentation of sequences, described by Haiminen, Gionis, and Laasonen [2008]. A

monotonic k-segmentation is a k-segmentation such that each segment representative has a value of at least the previous segment representative. Similarly, a unimodal k-segmentation is a k-segmentation where the segment representatives are monotonically increasing until a maximum is reached, after which the segment representatives are monotonically decreasing. The authors describe an exact algorithm for optimal monotonic or unimodal k-segmentation based on dynamic programming, with $O(n^2k)$ time complexity, as well as an approximate algorithm with time complexity of $O(n \log n)$.

Chundi and Rosenkrantz [2008] discuss the segmentation of item-set time series. Here, each item in the time series is represented as a set of discrete values (the item set) - essentially, this can be seen as a generalization of the symbolic sequence segmentation problem. An example of such a time series is a corpus of email communications, each of which may contain information regarding the sender, receivers, message subject, and so on. The contents of a segment are defined by a measure function, which for example determines that a value in the item set belongs to the segment if it occurs in at least a certain fraction of the items that make up the segment. The authors describe a number of optimal k-segmentation algorithms for item-set time series, for a number of different measure functions. The optimal k-segmentation algorithms are based on dynamic programming, and share the $O(n^2k)$ time complexity of the regular k-segmentation problem.

Considerations of time complexity. As mentioned before, it was shown by Bellman [1961] that an optimal solution to the k-segmentation problem can be found in $O(n^2k)$ time for a sequence of n items. The dynamic programming approach used to solve the k-segmentation problem is discussed later in this chapter.

A more interesting problem in terms of computational complexity is the k,h-segmentation problem, which as mentioned, is more appropriate for applications where segments are to be assigned to one of h classes. In this case, the dynamic programming approach by Bellman [1961] will no longer work, as it is not straightforward to choose the h segment representatives in advance for $h < k$.

The observations on the time complexity for the k,h-segmentation problem have already been discussed by Gionis and Mannila [2003]. They show that unlike the k-segmentation problem, the k,h-segmentation problem cannot be solved in polynomial time for sequences of dimensionality two or greater - at least, this has been shown for the L1 and L2 error criteria[3]. To see this, consider the special case of the k,h-segmentation problem, where k is equal to n, the length of the sequence to be segmented. Essentially, the ordering of the sequence is irrelevant here, as each item will end up as its own segment. As such, the n,h-segmentation problem is equivalent

[3]Here, the terms L1 and L2 (also written as L^1 and L^2) refer to the absolute value norm and the Euclidean norm, respectively; the Lp naming scheme is derived from the concept of Lp (or Lebesque) spaces.

to the well-known clustering problem. For the L1 and L2 error criteria, this corresponds to the k-median and k-means[4] problems, respectively. It is already known that these problems can be solved in polynomial time for dimensionality one [Megiddo, Zemel, and Hakimi, 1981], and that they are NP-hard for dimensionality two or greater [Megiddo and Supowit, 1984].

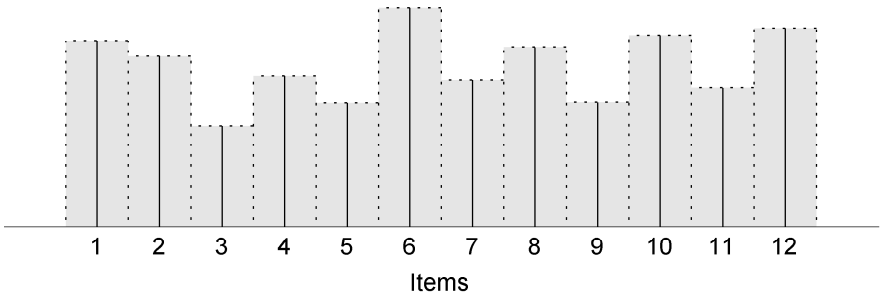
For sequences of dimensionality one, the time complexity of the k,h-segmentation problem is currently only known for a number of special cases: the n,h-segmentation problem, as described above, and the k,k-segmentation problem, which is equivalent to the regular k-segmentation problem. There currently seems to be no algorithm to solve the one-dimensional k,h-segmentation problem optimally in polynomial time for $h < k < n$. As such, the complexity of the one-dimensional k,h-segmentation problem remains an open problem.

Item duration and the alignment requirement. So far, the k-segmentation problem has only been explored for sequences of items of unit duration, or equivalently, items that are uniformly spaced - in the remainder of this chapter, the items in a sequence will be represented as consisting of given heights and durations. The difference between a sequence of (uniformly spaced) items of unit duration and a sequence of items of non-unit durations is visualized in Figure 2.1. For many applications, samples might be obtained at irregular intervals, or may retain the same value over a period of time. These cases can be modeled as each item having a certain (integer or real-valued) duration. For a sequence of items with non-unit durations, the *alignment requirement* need not hold; that is, items need not be part of only a single segment - or in other words, item boundaries need not coincide with segment boundaries.

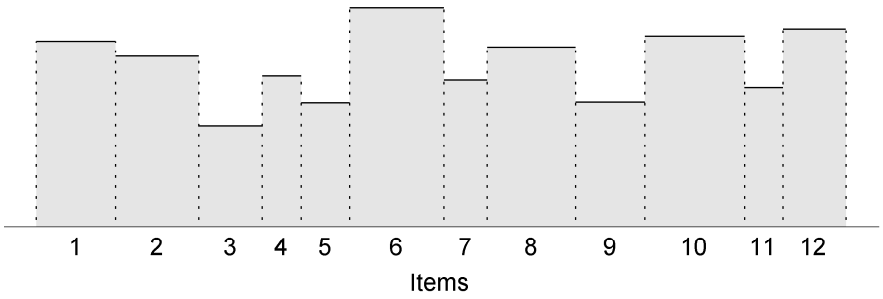
For sequences with items of non-unit duration, there is currently no guarantee that applying the aforementioned dynamic programming will still create an optimal k-segmentation, as it is limited to only exploring item boundaries and therefore adheres to the alignment requirement. This means that the algorithm may produce non-optimal segmentations for sequences of items with non-unit durations. Further, the performance of the many optimized or approximate k-segmentation algorithms which fully or partially rely on dynamic programming may also be impacted, such as the algorithms described by Tatti [2013], or Terzi and Tsaparas [2006].

In this chapter, we will show that even for sequences of items with integer or real-valued durations, or under conditions where the alignment requirement is not enforced, an optimal k-segmentation can be found through a similar dynamic programming approach. The proof for this is given in Section 2.3. In addition, we further explore the behavior of the L2 segmentation error function for segmentations that do not adhere to the alignment requirement in Section 2.4. In particular, we show that for two adjacent segments with items of integer durations, partially covering the same item, the error as a function of the coverage encounters at most a single maximum,

[4]Not to be confused with the k-means approximation algorithm.



(a) Sequence S_1 of unit duration items.



(b) Sequence S_2 of non-unit duration items.

Figure 2.1: Example of a sequence S_1 of items $i = 1,\ldots,12$ with unit durations, and a sequence S_2 of items $i = 1,\ldots,12$ with non-unit durations.

or may be monotonically decreasing or monotonically increasing. In Section 2.5, we show that this also holds for the L1 error criterion. Then, in Section 2.6, a number of examples of the behavior of the L1 and L2 error functions are shown and discussed. Finally, generalizations of the findings of Sections 2.4 and 2.5 to sequences of items with real-valued durations are discussed in Section 2.7.

2.2 Notation

Given a sequence S of items $i = 1,2,\ldots,n$, where item i is specified by its duration or width w_i, and its height h_i, a k-segmentation is a partitioning of the items i into k non-overlapping, contiguous segments $1,2,\ldots,k$ that completely cover all items. A segment j covers the items on the interval $[s_j,e_j) \subseteq [0,W)$, with $W = \sum_{i=1}^{n} w_i$. By definition, it holds that $s_1 = 0$, $e_k = W$, and $s_{j+1} = e_j$ for all $j = 1,2,\ldots,k-1$. For a visualization of such a sequence, see Figure 2.2.

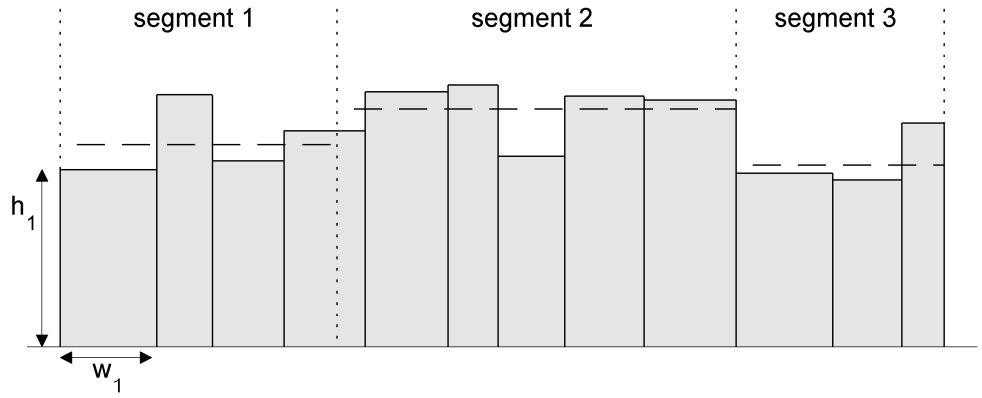


Figure 2.2: Visualization of an example (non-optimal) 3-segmentation of a sequence of 12 items with durations $w_1, \ldots, w_{12}$ and heights $h_1, \ldots, h_{12}$, divided into three segments 1, 2 and 3. The dashed, horizontal lines represent the segment means. The segment boundary between segment 2 and 3 fulfills the alignment requirement, while the segment boundary between segment 1 and 2 does not.

The L2 error E_j of segment j is given by

$$E_j = \sum_{i=1}^{n} x_{ij} w_i (h_i - \mu_j)^2,$$

where x_{ij} is the fraction of item i covered by segment j with $0 \leq x_{ij} \leq 1$ and $\sum_{j=1}^{k} x_{ij} = 1$, and μ_j denotes the weighted mean $\mu_j = \sum_{i=1}^{n} x_{ij} w_i h_i / \sum_{i=1}^{n} x_{ij} w_i$ of the heights in segment j.

Similarly, the L1 error E_j of a segment j is defined as

$$E_j = \sum_{i=1}^{n} x_{ij} w_i |h_i - m_j|.$$

Here, m_j represents the weighted median, which can be found as follows: let σ represent the sequence of all items at least partially covered by j, ordered with regards to height, and let the coverage c_i of an item i be given by

$$c_i = x_{ij} w_i.$$

For each item i in σ, the height h_i^σ and the coverage c_i^σ is determined. The weighted median m_j is then determined as the height h_l^σ of the first item l in σ for which it holds that $\sum_{i=1}^{l} c_i > \frac{1}{2} \sum_{i \in \sigma} c_i$, or as $\frac{h_l + h_{l+1}}{2}$ if $\sum_{i=1}^{l} c_i = \frac{1}{2} \sum_{i \in \sigma} c_i$ (see Figure 2.3 for an example).

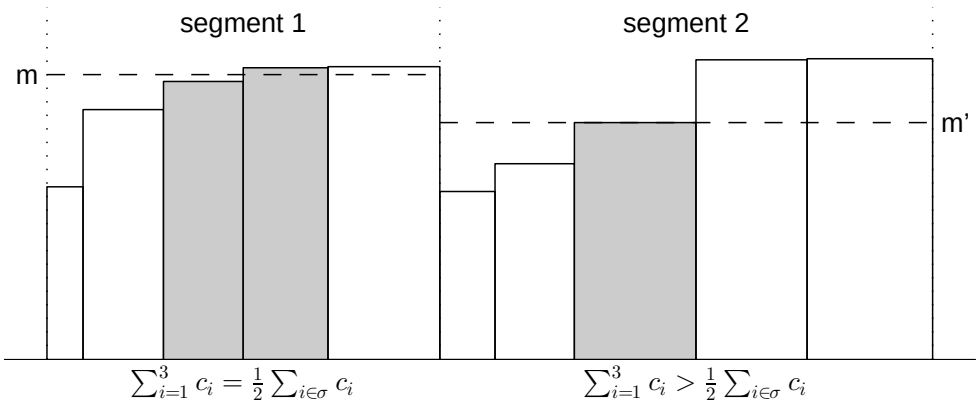


Figure 2.3: Visualization of the weighted segment medians of a sequence of 10 items, divided into two segments 1 and 2. The dashed, horizontal lines represent values of the weighted segment medians, determined from the items marked as gray, for the cases $\sum_{i=1}^{l} c_i = \frac{1}{2}\sum_{i\in\sigma} c_i$ for segment 1, and $\sum_{i=1}^{l} c_i > \frac{1}{2}\sum_{i\in\sigma} c_i$ for segment 2.

More general, we can define the Lp error E_j of a segment j as

$$E_j = \sum_{i=1}^{n} x_{ij} w_i |h_i - \mu_j|^p. \tag{2.1}$$

Here, μ_j does not necessarily represent the mean, but is often more generally called the representative of segment j. Just as the mean of a segment j minimizes the L2 error E_j of that segment, a representative minimizes the Lp error E_j of segment j.

A k-segmentation is considered optimal if the total error $E^{\text{tot}} = \sum_{j=1}^{k} E_j$ is minimal. The solution to the k-segmentation problem for a given sequence and error criterion therefore lies in finding the optimal k-segmentation. Let $\mathcal{S}^{\text{opt}}(k)$ be the optimal k-segmentation out of all possible k-segmentations $\mathcal{S}(k)$ for a given sequence S. The k-segmentation problem can then be formally described as follows.

Problem 4 (The k-segmentation problem). Given a sequence S consisting of items $i = 1, 2, \ldots, n$, with each item having of a duration w_i and height h_i, and given an integer k and an error criterion E, find the optimal k-segmentation $\mathcal{S}^{\text{opt}}(k) = \arg\min_{\mathcal{S}(k)} \sum_{j=1}^{k} E_j$. □

It is well-known from the literature [Bellman, 1961] that an optimal k-segmentation can be found in $O(n^2 k)$ time if the following conditions are met: first, the error criterion E^{tot} must be decomposable; that is, it must be possible to determine E^{tot} from the errors E_j of the individual segments. This is the case for the L1 and L2 error criteria, where the total error can be found as the sum of the individual segment errors. Second, it must be possible to determine the error criteria in poly-

nomial time. Third, the alignment requirement must hold, meaning that all items are fully covered by a single segment; that is, $x_{ij} = 1$ or $x_{ij} = 0$ for each pair i, j.

To illustrate how an optimal k-segmentation can be found through dynamic programming, we can describe the dynamic programming step as follows.

Theorem 2.1 (Optimal k-segmentation through dynamic programming). *Let $\mathcal{S}^{opt}([i,j),k)$ be the optimal k-segmentation on the interval $[i,j)$, let $E^{tot}(\mathcal{S}^{opt}([i,j),k))$ be the total error of the optimal k-segmentation on the interval $[i,j)$, and let $E([i,j))$ be the error of a segment defined on the interval $[i,j)$, with $i < j$. For a sequence with total duration W, the optimal k-segmentation up to duration $j \leq W$ can be found through the following recursive steps.*

$$E^{tot}(\mathcal{S}^{opt}([0,j),k)) = min\, E^{tot}(\mathcal{S}^{opt}([0,i),k-1)) + E([i,j)) \tag{2.2}$$
$$E^{tot}(\mathcal{S}^{opt}([0,j),1)) = E([0,j)).$$

□

In other words, provided that the alignment requirement is adhered to, the optimal k-segmentation for S can be found as the optimal $k-1$-segmentation on the items $1,\ldots,i$, and a single segment containing the items $i+1,\ldots,n$. To execute the recursion in (2.2), a dynamic programming table of size $n \times k$ must be constructed. The optimal k-segmentation can be found by filling this table for each entry i, j as $\min_{l<i} E^{\text{tot}}(\mathcal{S}^{\text{opt}}([0,l),j-1)) + E([l,i))$. The first part of the equation, $E^{\text{tot}}(\mathcal{S}^{\text{opt}}([0,l),j-1))$, can be found from examining the previous column. The second part, $E([l,i))$, needs to be calculated using the error criterion.

However, as also discussed by Terzi [2006], the approach in Theorem 2.1 leads to a time complexity of $O(n^2Tk)$, where T is the time needed to evaluate the error criterion; first, a total of nk table entries must be evaluated. Then, for each entry, the previous n rows must be examined, and the error criterion must be evaluated once. For the L1 and L2 error criteria, which can be computed in $O(n)$, this results in a time complexity of $O(n^3k)$, substantially more than the promised time complexity of $O(n^2k)$. However, for the L2 error criterion, it is possible to make use of the observation that for a segment j defined on $[l,l')$, $E([l,l'))$ can be found as

$$\begin{aligned}
E([l,l')) = E_j &= \sum_{i=1}^{n} x_{ij} w_i (h_i - \mu_j)^2 \\
&= \sum_{i=1}^{n} x_{ij} w_i h_i^2 - 2\mu_j \sum_{i=1}^{n} x_{ij} w_i h_i + \mu_j^2 \sum_{i=1}^{n} x_{ij} w_i \\
&= \sum_{i=1}^{n} x_{ij} w_i h_i^2 - 2\frac{1}{W_j}\left(\sum_{i=1}^{n} x_{ij} w_i h_i\right)^2 + W_j \left(\frac{1}{W_j}\sum_{i=1}^{n} x_{ij} w_i h_i\right)^2 \\
&= \sum_{i=1}^{n} x_{ij} w_i h_i^2 - \frac{1}{W_j}\left(\sum_{i=1}^{n} x_{ij} w_i h_i\right)^2
\end{aligned}$$

where $W_j = \sum_{i=1}^{n} x_{ij} w_i$, and making use of the definitions of E_j and μ_j. This means that, if the alignment requirement is adhered to, only the cumulative sum (cs) and cumulative sum of squares (css) for all items need to be maintained by the dynamic programming algorithm, along with the cumulative sum of item durations. That is, for each item i, $E_{\text{cs}}(i) = \sum_{j=1}^{i} w_j h_j$, $E_{\text{css}}(i) = \sum_{j=1}^{i} w_j h_j^2$, and $w_{\text{cs}}(i) = \sum_{j=1}^{i} w_j$. For any segment j defined on $[l,i)$, $E([l,i))$ can be computed as

$$E([l,i)) = (E_{\text{css}}(i) - E_{\text{css}}(l)) - \frac{1}{w_{\text{cs}}(i) - w_{\text{cs}}(l)} \left(E_{\text{cs}}(i) - E_{\text{cs}}(l)\right)^2 .$$

As this can be evaluated in linear time, this results in the desired time complexity of $O(n^2 k)$.

For the L1 error criterion, it is possible to precompute the medians for each possible segment in $O(n^2 \log n)$ time. Assuming a sequence of items is already sorted, the median of that segment plus an additional item can be found in $O(\log n)$ time - this is the time required for a binary search needed to insert the new item in the sorted sequence. The median can then be found in constant time. In total, there are n possible segment starts, each of a potential length of $O(n)$, yielding the total $O(n^2 \log n)$ processing time. This does result in a total time complexity of $O(n^2 \log n + n^2 k)$, which may be larger than $O(n^2 k)$ if $\log n > k$. However, computing the medians is often considered as a preprocessing step, and as such not factored into the $O(n^2 k)$ time complexity [Terzi, 2006].

The alignment requirement and optimal k-segmentation. In the remainder of this chapter, it will first be shown in Section 2.3 that for sequences not adherent to the alignment requirement, an optimal k-segmentation will still be found in $O(n^2 k)$ time, using the same algorithmic approach described above. Specifically, it will be shown that for two adjacent segments partially covering the same item, the Lp error will be minimal only under conditions where the alignment requirement holds.

We then further explore the behavior of the segmentation error function for segmentations that do not adhere to the alignment requirement, for the L1 and L2 error criteria specifically. For both error criteria, we can show that for two adjacent segments partially covering the same item, the error as a function of the coverage encounters at most a single maximum, or may be monotonically decreasing or monotonically increasing. Apart from a single possible maximum, there are no other stationary points such as saddlepoints or minima (the absence of minima also directly follows from the proof in Section 2.3).

We use the second derivative test to show this for the L2 norm in Section 2.4. In particular, we consider the error as a function of the coverage of a shared item, and show that the first order derivative of the error function never equals zero unless the second order derivative is negative for the domain of the error function. For the L1 error, discussed in Section 2.5, the error function is based instead on the number of subitems of a shared item, that are moved from one segment to another. Here, we

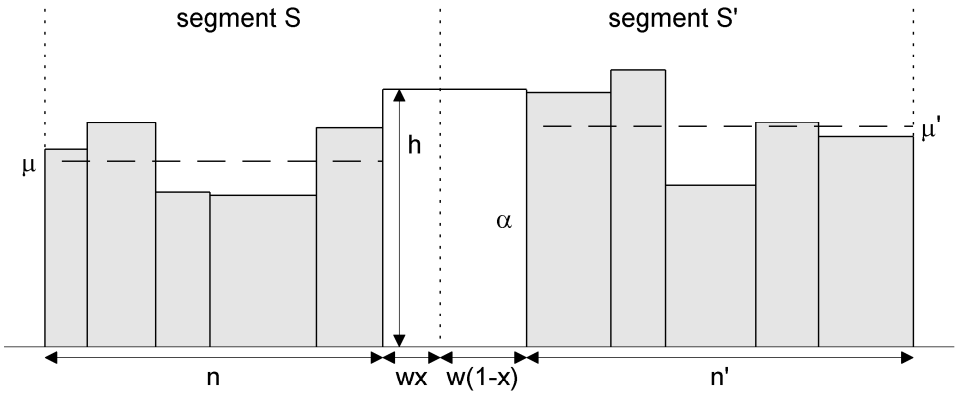


Figure 2.4: Visualization of the notation used in Section 2.3.

show that the error function is concave, akin to a descending first order derivative. In Section 2.6, a few examples of the error function behavior for the L1 and L2 error criteria are discussed. Finally, some generalizations to the results in Sections 2.4 and 2.5 are briefly discussed in Section 2.7.

2.3 Dynamic programming optimality without alignment

In this section, we will show that for sequences with integer or real-valued durations, that do not adhere to the alignment requirement, an optimal k-segmentation will still be found in $O(n^2k)$ time, using the same dynamic programming approach described above. Specifically, it will be shown that for any partial coverage of an item by two segments, the overall segmentation error for those segments can be improved by having the item fully covered by one of the segments.

For ease of notation, let S and S' denote two adjacent segments. Segment S fully covers n items with heights $h_1, \ldots, h_n$ and integer durations $w_1, \ldots, w_n$. In addition, S partially covers an item α with height h and duration w. The fraction of α that is covered by S is denoted as x, with $x \in [0,1]$. The remaining $(1-x)$ fraction of item α is covered by S', which additionally covers n' items with heights $h'_1, \ldots, h'_{n'}$ and durations $w'_1, \ldots, w'_{n'}$. An example of such a sequence and segmentation is shown in Figure 2.4.

Let $\mathcal{D}_p(y, \mu)$ be the distance function for the Lp error norm. The distance function is defined as follows.

Definition 2.1 (Distance function). For the Lp norm, the distance function $\mathcal{D}_p(y,\mu)$ is defined as

$$\mathcal{D}_p(y) = |y-\mu|^p$$

where μ is the representative of a given segment. □

From here on, we will refer to the representatives of segments S and S' as μ and μ', respectively. The total error E^{tot} over the segments S and S' can be determined as the sum of the errors over four separate subsegments, $E^{\text{tot}} = E_S + E_1 + E_2 + E_{S'}$. Here, let E_S is the error over the fully covered items $1,\ldots,n$ in segment S, let E_1 denote the error over α for the initial xw duration, let E_2 denote the error over the remaining $(x-1)w$ duration of α, and finally, let $E_{S'}$ be the error over the items $1,\ldots,n'$ in segment S'. As representatives of their respective segments, μ minimizes the error $E_S + E_1$, and μ' minimizes the error $S_{S'} + E_2$.

Lemma 2.1 (Subsegment errors). *Let $E^{tot} = E_S + E_1 + E_2 + E_{S'}$ denote the total error over the segments S and S′. The errors over the individual subsegments E_S, E_1, E_2, and $E_{S'}$ can be found as*

$$\begin{aligned}
E_S &= \sum_{i=1}^{n} w_i \mathcal{D}_p(h_i,\mu) \\
E_1 &= wx\mathcal{D}_p(h,\mu) \\
E_2 &= w(x-1)\mathcal{D}_p(h,\mu') \\
E_{S'} &= \sum_{i=1}^{n'} w_i' \mathcal{D}_p(h_i',\mu').
\end{aligned}$$

Proof. The subsegment errors E_S, E_1, E_2, and $E_{S'}$ follow directly from Equation 2.1 and Definition 2.1. □

The proof that the total error E^{tot} can be improved by having the item fully covered by one of the segments will follow two steps: first, we show that E^{tot} can be improved or at least maintained by shifting the coverage x, without adjusting the representatives μ and μ'. Second, we show that then adjusting the representatives for the new coverage will further decrease or maintain the total error.

For the remainder of the proof, we distinguish between three possible cases: $\mathcal{D}_p(h,\mu) < \mathcal{D}_p(h,\mu')$, $\mathcal{D}_p(h,\mu) > \mathcal{D}_p(h,\mu')$, or $\mathcal{D}_p(h,\mu) = \mathcal{D}_p(h,\mu')$. The latter case is trivial; regardless of the value of x, the segment representatives μ and μ' will not change, and as such, E^{tot} is identical for all values of x. That is, α will contribute the same amount to the overall error, regardless of the distribution of coverage over either segment. We will show this as follows.

Lemma 2.2 (Coverage independence under equal representatives). *In the case that $\mathcal{D}_p(h,\mu) = \mathcal{D}_p(h,\mu')$, the total error E^{tot} does not depend on the coverage x.*

Proof. From Lemma 2.1, we have $E^{\text{tot}} = E_S + E_1 + E_2 + E_{S'}$. Further, we can see from Lemma 2.1 that E_S and $E_{S'}$ do not depend on x. We then have

$$E_1 + E_2 = wx\mathcal{D}_p(h,\mu) + w(x-1)\mathcal{D}_p(h,\mu').$$

Given that $\mathcal{D}_p(h,\mu) = \mathcal{D}_p(h,\mu')$, it follows that

$$\begin{aligned} E_1 + E_2 &= wx\mathcal{D}_p(h,\mu) + w(x-1)\mathcal{D}_p(h,\mu) \\ &= w\mathcal{D}_p(h,\mu). \end{aligned}$$

As a result, we can see that none of the terms in $E^{\text{tot}} = E_S + E_1 + E_2 + E_{S'}$ depend on x, and therefore, that the total error E^{tot} does not depend on x. □

The other two cases, $\mathcal{D}_p(h,\mu) < \mathcal{D}_p(h,\mu')$ and $\mathcal{D}_p(h,\mu) > \mathcal{D}_p(h,\mu')$, can be proven analogously. Below, we will show the proof for the first of the two cases, where $\mathcal{D}_p(h,\mu) < \mathcal{D}_p(h,\mu')$.

Lemma 2.3 (Coverage adjustment). *If $\mathcal{D}_p(h,\mu) < \mathcal{D}_p(h,\mu)$, then even without adjusting the representatives μ and μ', the total error E^{tot} can be decreased by having α be fully covered by segment S, that is, by setting $x = 1$.*

Proof. To see this, we first note that E_S and $E_{S'}$ are not affected by this change, as they do not depend on the value of x. Further, we have from Lemma 2.1

$$\begin{aligned} wx\mathcal{D}_p(h,\mu) + w(x-1)\mathcal{D}_p(h,\mu) &< wx\mathcal{D}_p(h,\mu) + w(x-1)\mathcal{D}_p(h,\mu') \\ E_1 + \tilde{E}_2 &< E_1 + E_2, \end{aligned}$$

where $\tilde{E}_2 = w(x-1)\mathcal{D}_p(h,\mu)$. The total error $\tilde{E}^{\text{tot}}$ after shifting the coverage of α is then given as

$$\tilde{E}^{\text{tot}} = E_S + E_1 + \tilde{E}_1 + E_{S'}.$$

As $\tilde{E}_2 < E_2$, it follows that $\tilde{E}^{\text{tot}} < E^{\text{tot}}$. □

Similarly, if $\mathcal{D}_p(h,\mu) > \mathcal{D}_p(h,\mu)$, the total error can be decreased by shifting α to be fully covered by segment S'. As shown in Lemma 2.2, in the case of $\mathcal{D}_p(h,\mu) = \mathcal{D}_p(h,\mu')$, α can be fully covered by either segment without a change in error.

Changing the coverage of α will in most cases result in a change of the representatives μ and μ'. However, changing the representatives can only further decrease the error $\tilde{E}^{\text{tot}}$. Let $\hat{E}$ be the error for the updated representatives. Since the updated representatives by definition minimize the error within their respective segments, we can show that $\hat{E}^{\text{tot}}$ will not increase the error of $\tilde{E}^{\text{tot}}$.

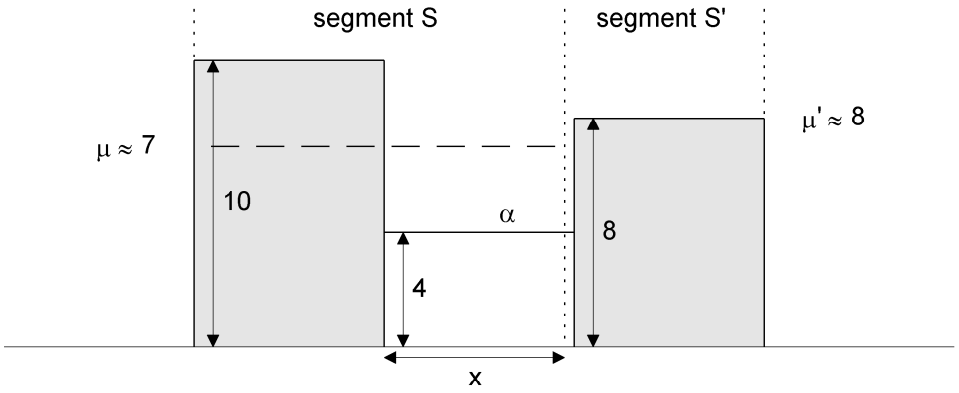


Figure 2.5: Example of a sequence and segmentation for which the method described in Section 2.3 chooses a non-optimal segmentation.

Theorem 2.2 (Optimality without the alignment requirement). *Given two segments S and S', and an item α which is partially covered by both segments with a fraction x of its duration covered by segment S, the segmentation error $\hat{E}^{tot}$ after selecting $x = 1$ if $\mathcal{D}_p(h,\mu) \leq \mathcal{D}_p(h,\mu')$ or $x = 0$ if $\mathcal{D}_p(h,\mu) > \mathcal{D}_p(h,\mu')$ will be less than the error E^{tot} of the initial segmentation.*

Proof. If we assume the case $\mathcal{D}_p(h,\mu) < \mathcal{D}_p(h,\mu')$, then from Lemmas 2.3 and 2.1, as well as the observation that the adjustment of a segment representative minimizes the error over that segment, we have

$$\hat{E}_{S'} \leq E_{S'}$$
$$\hat{E}_S + \hat{E}_1 + \hat{E}_2 \leq E_S + E_1 + \tilde{E}_2 < E_S + E_1 + E_2.$$

As a result, it follows from the definition of E^{tot} that $\hat{E}^{\text{tot}} \leq \tilde{E}^{\text{tot}} < E^{\text{tot}}$. The proof for the case $\mathcal{D}_p(h,\mu) > \mathcal{D}_p(h,\mu')$ follows analogously, and the proof for the case $\mathcal{D}_p(h,\mu) = \mathcal{D}_p(h,\mu')$ is trivial, as $\hat{E}^{\text{tot}} = \tilde{E}^{\text{tot}} = E^{\text{tot}}$. □

Finding the optimal segmentation. Following the method described in Section 2.3 will not always result in arriving at the optimal segmentation, however. To see this, consider the following example for the L2 error criterion: let S consist of a single item of height $h_1 = 10$, let S' consist of a single item of height $h'_1 = 8$, let α have height $h = 4$, and let all items have a uniform duration of one. We then choose the value of x as $x \to 1$; that is, x approaches one. We can then determine the segment means as $\mu \approx \frac{10+4}{2} = 7$ and $\mu' \approx 8$. This sequence is also shown in Figure 2.5.

Given that $\mathcal{D}_2(4,\mu) = |4-\mu|^2 = 9 < \mathcal{D}_2(4,\mu') = |4-\mu'|^2 = 16$, we would choose to have α be fully covered by segment S (i.e., $x = 1$). The total error of this segmentation is then given as $E^{\text{tot}} = (10-\mu)^2 + (4-\mu)^2 + (8-\mu')^2 = 10$. How-

ever, had we opted to have α fully covered by S' (i.e., $x = 0$), we would have found the segment means as $\mu = 10$ and $\mu' = \frac{4+8}{2} = 6$, and the resulting total error of the segmentation would have been $E^{\text{tot}} = (10-\mu)^2 + (4-\mu')^2 + (8-\mu')^2 = 8$. Clearly, having α covered by S' would have resulted in a better segmentation.

In practice, this does not pose much of a concern. In the worst case, we would need to examine the error for the two boundary points to find the optimal segmentation, which is preferable to having to explore the potentially infinite amount of points within the item boundaries. In addition, both points of the item boundaries are examined anyway in the dynamic programming algorithm, so no adaptations are necessary. However, the above example does indicate that the segmentation error does not always follow a monotonically increasing or decreasing trend within the item boundaries. In the remaining sections, we will explore the behavior of the segmentation error in more detail for the L1 and L2 error criteria specifically. In particular, we will show that the segmentation error within the boundaries of an item has at most a single maximum, and no other stationary points.

2.4 Error within item boundaries for the L2 error case

In this section, we further explore the behavior of the segmentation error for the L2 norm specifically when the segment boundary does not align with an item boundary. Here, the segmentation error $E^{\text{tot}}(x)$ is modeled as a function of the fraction of coverage x of the item split by the segment boundary. We will show that the segmentation error as a function of x has at most a single maximum, and no other stationary points, within the range of valid values for x. In other words, $E^{\text{tot}}(x)$ is either monotonically increasing, monotonically decreasing, or unimodal[5].

As before, let S and S' denote two adjacent segments. Segment S fully covers n items with heights $h_1, \ldots, h_n$ and integer durations $w_1, \ldots, w_n$. In addition, S partially covers an item α with height h and duration w. The fraction of α that is covered by S is denoted as x, with $x \in [0,1]$. The remaining $(1-x)$ fraction of item α is covered by S', which additionally covers n' items with heights $h'_1, \ldots, h'_{n'}$ and durations $w'_1, \ldots, w'_{n'}$.

For the sake of notational convenience, it will be assumed that, apart from w, all other durations are of unit length, that is, $w_i = 1$ for all $i = 1, \ldots, n$, and $w'_i = 1$ for all $i = 1, \ldots, n'$. For integer-valued durations, these assumptions can be made without loss of generality; for every item with duration w_i, it would be possible to replace this item with w_i new items of unit duration, with the same height h_i, and adjust n and n' accordingly. A further generalization to items with real-valued durations is discussed in Section 2.7. It is also assumed that, apart from the boundary between S and S', they both fulfill the alignment requirement. These requirements are not necessary (see Section 2.7), but do simplify notation considerably.

[5] That is, monotonically increasing for $x \leq m$ for some value m, and monotonically decreasing for $x > m$.

A visualization of two segments S and S', and their boundary on the item α of height h and duration w, is shown in Figure 2.4.

Lemma 2.4 (Mean). *For any value of x, the fraction of item α covered by segment S, the mean $\mu(x)$ of S can be expressed as $\mu(x) = \mu + \delta(x)$, where μ is the mean for items $1,\ldots,n$, and $\delta(x)$ is given by*

$$\delta(x) = \frac{wx}{n+wx}(h-\mu), \tag{2.3}$$

and the mean $\mu'(x)$ of segment S' can be expressed as $\mu'(x) = \mu' + \delta'(x)$, with $\mu' = \frac{1}{n'}\sum_{i=1}^{n'} h'_i$, and

$$\delta'(x) = \frac{w(1-x)}{n'+w(1-x)}(h-\mu').$$

Proof. Let $\nu = \sum_{i=1}^{n} h_i$. Then, assuming the items $1,\ldots,n$ are of unit duration, it can be seen from the definition of the mean that $\mu = \frac{\nu}{n}$, and $\mu(x) = \frac{\nu+wxh}{n+wx}$. As we have $\mu(x) = \mu + (\mu(x)-\mu)$, it follows that

$$\delta(x) = \mu(x) - \mu = \frac{wxh}{n+wx} - \frac{wx\nu}{(n+wx)n} = \frac{wx}{n+wx}(h-\mu).$$

The proof for $\mu'(x)$ is analogous.

□

The L2 error for S and S' can then be expressed as follows:

Lemma 2.5 (Error). *The L2 error $E(x)$ for segment S is given by*

$$E(x) = E + \frac{nwx}{n+wx}(h-\mu)^2, \tag{2.4}$$

and, analogously, the error $E'(x)$ for segment S' is given by

$$E'(x) = E' + \frac{n'w(1-x)}{n'+w(1-x)}(h-\mu')^2 \tag{2.5}$$

with $E = \sum_{i=1}^{n}(h_i-\mu)^2$, and $E' = \sum_{i=1}^{n'}(h'_i-\mu')^2$.

Proof. Following the L2 error definition for segment S,

$$E(x) = \sum_{i=1}^{n}(h_i-\mu-\delta(x))^2 + wx(h-\mu-\delta(x))^2. \tag{2.6}$$

Making use of the equality $\sum_{i=1}^{n}(h_i-\mu) = 0$, expanding the first term of (2.6) gives

$$\sum_{i=1}^{n}(h_i-\mu-\delta(x))^2 = \sum_{i=1}^{n}((h_i-\mu)^2+\delta(x)^2) = E + n\delta(x)^2. \tag{2.7}$$

The second term in (2.6) can be rewritten using the definition of $\delta(x)$, which yields

$$wx(h-\mu-\delta(x))^2 = wx(h-\mu-\frac{wx}{n+wx}(h-\mu))^2 = \frac{n^2wx}{(n+wx)^2}(h-\mu)^2.$$

Combining this result with (2.7), (2.6) can be rewritten as

$$E(x) = E + \frac{nw^2x^2}{(n+wx)^2}(h-\mu)^2 + \frac{n^2wx}{(n+wx)^2}(h-\mu)^2.$$

With some further simplification, this results in (2.4). The proof of (2.5) can be obtained analogously. □

The total error of both segments $E^{\text{tot}}(x) = E(x) + E'(x)$ is then given as

$$E^{\text{tot}}(x) = E + E' + \frac{nwx}{n+wx}(h-\mu)^2 + \frac{n'w(1-x)}{n'+w(1-x)}(h-\mu')^2.$$

The total error $E^{\text{tot}}(x)$ can be considered as a twice differentiable function of x on the interval $[0,1]$, in which E and E' are constant terms. Any stationary points on $(0,1)$ can then be found by setting the first order derivative of $E^{\text{tot}}(x)$ equal to zero (as the first derivative is fully defined over the interval $[0,1]$). The second derivative test can then be used to determine if the stationary point in question is a maximum (if less than zero), minimum (if greater than zero), or an inflection point (if equal to zero, indicating a possible saddlepoint). We will show that on the domain of $E^{\text{tot}}(x)$, the only stationary points are maxima, by proving that minima and inflection points can not exist on the interval $(0,1)$.

The first-order derivative of $E^{\text{tot}}(x)$ can, after some math, be found as

$$\frac{d}{dx}E^{\text{tot}}(x) = \frac{w(h-\mu)^2n^2}{(n+wx)^2} - \frac{w(h-\mu')^2n'^2}{(n'+w(1-x))^2}. \tag{2.8}$$

Equating this to zero yields, after some more math, two solutions, as the following lemma states.

Lemma 2.6 (Zero derivative). *The two values of x for which the first order derivative of the total error E^{tot} given in (2.8) equals zero are given by*

$$x_1 = \frac{n\Big(w(h-\mu)+n'((h-\mu)+(h-\mu'))\Big)}{w\Big(n(h-\mu)-n'(h-\mu')\Big)} \tag{2.9}$$

$$x_2 = \frac{n\Big(w(h-\mu)+n'(\mu'-\mu)\Big)}{w\Big(n(h-\mu)+n'(h-\mu')\Big)}. \tag{2.10}$$

Proof. To find the values of x for which $\frac{d}{dx}E^{\text{tot}}(x) = 0$, (2.8) gives

$$\frac{w(h-\mu)^2n^2}{(n+wx)^2} = \frac{w(h-\mu')^2n'^2}{(n'+w(1-x))^2}.$$

Combining both fractions and leaving out the common denominator results in

$$w(h-\mu)^2n^2(n'+w(1-x))^2 = w(h-\mu')^2n'^2(n+wx)^2$$
$$\Big((h-\mu)n(n'+w(1-x))\Big)^2 = \Big((h-\mu')n'(n+wx)\Big)^2.$$

Since the result is a statement of the form $A^2 = B^2$, this yields two possible solutions, $A = B$ or $A = -B$. As a result, we have

$$(h-\mu)n(n'+w(1-x)) + (h-\mu')n'(n+wx) = 0 \tag{2.11}$$

or

$$(h-\mu)n(n'+w(1-x)) - (h-\mu')n'(n+wx) = 0. \tag{2.12}$$

By moving x to one side of the equation in (2.11) and (2.12), the expressions for x_1 and x_2 given in (2.9) and (2.10), respectively, can be obtained.

□

Note that x_1 is undefined when $n(h-\mu) = n'(h-\mu')$, and x_2 is undefined when $n(h-\mu) = -n'(h-\mu')$.

The second derivative of $E^{\text{tot}}(x)$ can be found as

$$\frac{d^2}{dx^2}E^{\text{tot}}(x) = -2w^2\left(\frac{(h-\mu')^2n'^2}{(n'+w(1-x))^3} + \frac{(h-\mu)^2n^2}{(n+wx)^3}\right). \tag{2.13}$$

Inserting the two points for which $\frac{d}{dx}E^{\text{tot}}(x) = 0$ into (2.13) results, after some math, in the following two second order derivatives. For x_1 in (2.9), the second order derivative can be obtained as

$$\frac{2w^2(n(h-\mu) - n'(h-\mu'))^4}{n(h-\mu)n'(h-\mu')(w+n+n')^3}. \tag{2.14}$$

And for x_2 in (2.10), the derivative can be obtained as

$$-\frac{2w^2(n(h-\mu) + n'(h-\mu'))^4}{n(h-\mu)n'(h-\mu')(w+n+n')^3}. \tag{2.15}$$

Note that the second derivatives are undefined if $h = \mu$ or $h = \mu'$. From Lemma 2.5, it can be seen that under these conditions, $E(x) = E$ or $E'(x) = E'$, respectively, and that there is no lower error than for $x = 1$ or $x = 0$. In this case, the error is monotonically increasing as x is decreased or increased, respectively.

First, we will examine the existence of any inflection points in the interval $(0,1)$. In other words, x_1 or x_2 must be on the interval $(0,1)$, with a zero second order derivative. It is fairly straightforward to show that such points do not exist, as detailed below.

Lemma 2.7 (Inflection point). *For points x_1 defined by (2.9) and x_2 defined by (2.10) where the first derivative of $E^{tot}(x)$ is equal to zero, it holds that x_1 is not on the interval $(0,1)$, or is not an inflection point. Similarly, it holds that x_2 is not on the interval $(0,1)$, or is not an inflection point.*

Proof. By definition, x_1 or x_2 are inflection points if the second order derivative is equal to zero. From (2.14) and (2.15), we can see that the second order derivative is zero if and only if $n(h-\mu) = n'(h-\mu')$ for x_1, and $n(h-\mu) = -n'(h-\mu')$ for x_2. From (2.9), however, we can see that x_1 is undefined if $n(h-\mu) = n'(h-\mu')$, and from (2.10), we can see that x_2 is undefined if $n(h-\mu) = -n'(h-\mu')$. Therefore, x_1 and x_2 are not in the interval $(0,1)$. □

Second, we will show that no minimum exists on the interval $(0,1)$ through proof by contradiction. Assuming that a minimum does exist on the interval $(0,1)$, and therefore the second derivative must be positive, it can be seen that the expression $((h > \mu) \wedge (h > \mu')) \vee ((h < \mu) \wedge (h < \mu'))$ must hold for (2.14). Similarly, for (2.15), it holds that $(\mu < h < \mu') \vee (\mu' < h < \mu)$. It will now be shown that under these conditions, there are no points in the interval $(0,1)$ for which the first derivative equals zero.

Lemma 2.8 (Local minimum, case 1). *For the first solution for which the derivative of $E^{tot}(x)$ equals zero, x_1, given by* (2.9)*, it holds that either x_1 is not in the interval $(0,1)$, or the second order derivative is not positive at x_1.*

Proof. First assume that $x_1 \in (0,1)$. It can be shown that this leads to a contradiction. For the first condition, $(h > \mu) \wedge (h > \mu')$, observe that the numerator in (2.9) is positive. As the assumption is that $x_1 > 0$, it follows that the denominator is positive as well. Using this observation and the assumption that $x_1 < 1$, it follows from (2.9) that

$$\begin{aligned} wn(h-\mu) - wn'(h-\mu') &> wn(h-\mu) + nn'((h-\mu) + (h-\mu')) \\ -wn'(h-\mu') &> nn'(h-\mu) + nn'(h-\mu') \\ (w+2n)h &< (w+n)\mu' + n\mu. \end{aligned} \tag{2.16}$$

From (2.16), it can be seen that if $\mu \geq \mu'$, it is possible to substitute μ' in (2.16) for μ, yielding $h < \mu$, contradicting the condition $(h > \mu) \wedge (h > \mu')$. Alternatively, if $\mu' > \mu$, it follows that $h < \mu'$, also contradicting our initial condition. Hence, under the first condition, there are no points in the interval $(0,1)$ for which the first derivative equals zero and the second derivative is positive.

For the second condition, $(h < \mu) \wedge (h < \mu')$, a contradiction can be found in similar fashion. As under this condition the denominator in (2.9) is negative, it follows that

$$wn(h-\mu) - wn'(h-\mu') < nw(h-\mu) + nn'((h-\mu)+(h-\mu')),$$

which results in

$$(w+2n)h > (w+n)\mu' + n\mu.$$

Using similar reasoning as for the first condition, it can be seen that for both $\mu \geq \mu'$ and $\mu' > \mu$, the condition $(h < \mu) \wedge (h < \mu')$ is contradicted. □

Lemma 2.9 (Local minimum, case 2). *For the second point for which the derivative of $E^{tot}(x)$ equals zero, x_2 given in* (2.10)*, it holds that either x_2 is not on the interval* $(0,1)$*, or the second order derivative is not positive at x_2.*

Proof. From the requirement that x_2 is in $(0,1)$, assume that $0 < x_2 < 1$. For the first condition, $(\mu < h < \mu')$, $x_2 < 1$ can be rewritten as

$$\begin{aligned} wn(h-\mu) + wn'(h-\mu') &> wn(h-\mu) + nn'(\mu'-\mu) \\ w(h-\mu') &> n(\mu'-\mu) \\ h &> \mu' - \frac{n}{w}(\mu-\mu'). \end{aligned} \tag{2.17}$$

As $\mu' - \frac{n}{w}(\mu-\mu') > \mu'$ in (2.17) under the first condition $(\mu < h < \mu')$, it holds that $h > \mu'$, which contradicts this condition.

For the second condition, $(\mu' < h < \mu)$, it follows that

$$wn(h-\mu) + wn'(h-\mu') < wn(h-\mu) + nn'(\mu'-\mu)$$

which results in

$$h < \mu' - \frac{n}{w}(\mu'-\mu).$$

Since $(\mu' < h < \mu)$, this yields $h < \mu'$, contradicting the second condition. □

Using these results, it can then be shown that $E^{\text{tot}}(x)$ has at most one maximum in the interval $(0,1)$.

Theorem 2.3 (Stationary points for the L2 error criterion). *For two adjacent segments S and S′, with n the number of items completely covered by S (excluding α) and μ the mean of S, and n' the number of items completely covered by S′ (excluding α) and μ' the mean of S′, and an item α partially covered by S and S′ with duration w and height h, and the error function $E^{tot}(x)$ given by*

$$E^{tot}(x) = E + E' + \frac{nwx}{n+wx}(h-\mu)^2 + \frac{n'w(1-x)}{n'+w(1-x)}(h-\mu')^2,$$

where $x \in [0,1]$, the error function $E^{tot}(x)$ has at most one maximum in the interval $[0,1]$*, and no other stationary points.*

Proof. The proof directly follows from Lemmas 2.7, 2.8 and 2.9. As there cannot exist two maxima on the interval $[0,1]$ without also a minimum existing, it follows that there can be at most a single maximum. □

It can also be noted that (for the L2 norm) the above can serve as an alternative proof to Theorem 2.2; as there exist no minima on the interval $[0,1]$, it follows that $E^{\text{tot}}(x)$ can not be less than its value at one of the item boundaries; that is, either at $x = 0$ or $x = 1$.

2.5 Error within item boundaries for the L1 error case

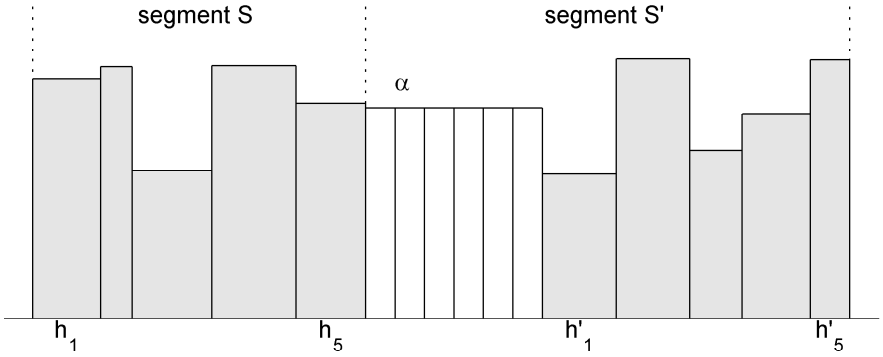
In this section we will show that, like for the L2 error norm, the behavior of the segmentation error $E^{\text{tot}}(x)$ will show at most a single maximum inside of the item boundaries, and no other stationary points. Here, $E^{\text{tot}}(x)$ is the segmentation error as a function of the coverage x of an item split by a segment boundary. To show this, once again two adjacent segments S and S', and an item α placed in between both segments, are considered. Initially, let α be fully covered by S'. By incrementally moving part of this item from one segment to the other through subitems, it will be demonstrated that at most a single maximum can be encountered.

Once again, let S and S' denote two adjacent segments. Segment S fully covers n items with heights $h_1,\ldots,h_n$ and integer durations $w_1,\ldots,w_n$. S' covers n' items with heights $h'_1,\ldots,h'_{n'}$ and durations $w'_1,\ldots,w'_{n'}$. In addition, S' fully covers an item α with height h and duration w, where α is also adjacent to segment S. For ease of notation, assume that each item covered by S and S' is split into w_i subitems of unit duration and height h_i. Furthermore, the assumption is made that, apart from their common segment boundary, S and S' adhere to the alignment requirement. The heights of the items fully covered by S and S', ordered by height and excluding item α, are denoted as $a = a_1,\ldots,a_n$ for items covered by S and $b = b_1,\ldots,b_{n'}$ for items covered by S', where $n = \sum_{i\in S} w_i$ and $n' = \sum_{i\in S'} w_i$.

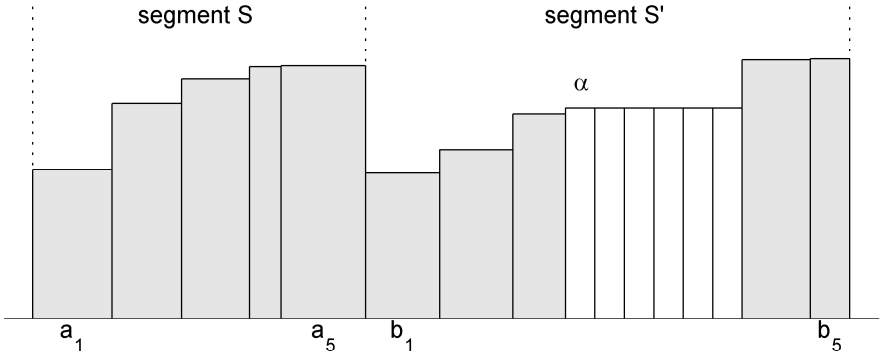
Apart from the above assumption that the items in S and S' can be split into subitems of unit duration, a number of further assumptions are made. First, assume that α can be split into an even number of subitems of unit duration (w is even), and that α is initially fully covered by S. Second, for notational convenience of the median, also assume n and n' are even. Further, the initial heights for each of the segments S and S' ($a_1,\ldots,a_n$ and $b_1,\ldots,b_{n'}$, respectively) are in ascending order - note that the subitems of α are not included in a or b. These assumptions can be made without loss of generality, and are further discussed in Section 2.7.2.

Given the above assumptions, the initial median of segment S can be found as

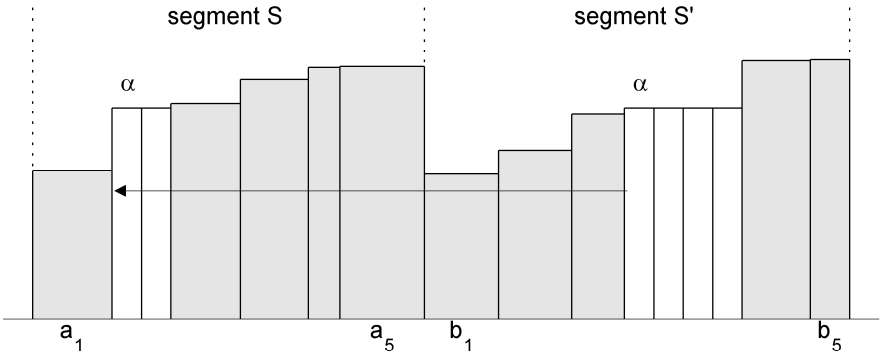
$$m_0 = \frac{1}{2}a_{\frac{n}{2}} + \frac{1}{2}a_{\frac{n}{2}+1}.$$



(a) Initial, unordered sequence of two segments S and S', and item α, where α is initially assumed to be completely assigned to S'.



(b) The original items in the segments are ordered in a and b. Item α is initially fully covered by S'.



(c) Incrementally, subitems of α are moved from segment S to segment S'.

Figure 2.6: Visualization of the notation used in Section 2.5 with $n = n' = 5$.

Part of the coverage of α is then moved from S' to S incrementally. At each increment, two of the subitems are moved from segment S' to S. The number of times that a subitem of α was moved from S' to S is denoted with x. In general, the median of segment S when x subitems of α have been moved to S is denoted as m_x, and the median of segment S' when x subitems of α have been removed from S' as m'_x. We will prove that for all even values of x, the L1 error as a function of x is concave[6], and therefore has at most a single mode (that is, x has at most a single maximum, and no other stationary points). For an example of the notation used, see Figure 2.6.

Depending on the value of the initial median, m_0, there are a number of cases to be considered with regard to the L1 error for segment S. If $m_0 = h$, then regardless of the number of subitems of α added to S (given by x), there will be no difference in error as the median will remain equal to h. This leaves two non-trivial cases, $m_0 < h$ and $m_0 > h$, which will be discussed below. A similar set of cases can be defined for segment S', with the non-trivial cases $m'_0 < h$ and $m'_0 > h$.

The error of both segments S and S' depends on x, the number of subitems of α moved from S' to S. Let $E^{\text{tot}}_{2x} = E_{2x} + E'_{2x}$ be the error of segments S and S' together when $2x$ subitems of α have been moved from segment S' to segment S. To show that E^{tot}_{2x} is a concave function, we must show that

$$E^{\text{tot}}_{2x} \geq \frac{E^{\text{tot}}_{2(x+1)} + E^{\text{tot}}_{2(x-1)}}{2}, \tag{2.18}$$

per the definition of a concave function. To see this, image that $E^{\text{tot}}_{2x} < \frac{E^{\text{tot}}_{2(x+1)} + E^{\text{tot}}_{2(x-1)}}{2}$; in this case, a line can be drawn between $E^{\text{tot}}_{2(x+1)}$ and $E^{\text{tot}}_{2(x-1)}$ that contains a point at position $2x$ greater than E^{tot}_{2x}, invalidating the notion of concavity. We can rewrite (2.18) as $E^{\text{tot}}_{2(x+1)} - E^{\text{tot}}_{2x} \leq E^{\text{tot}}_{2x} - E^{\text{tot}}_{2(x-1)}$. Therefore, E^{tot}_{2x} is concave if the following holds.

Theorem 2.4 (Stationary points for the L1 error criterion). *For all values of $x \in [1, \ldots, \frac{w}{2} - 1]$, E^{tot}_{2x} is concave. That is, $E^{tot}_{2(x+1)} - E^{tot}_{2x} \leq E^{tot}_{2x} - E^{tot}_{2(x-1)}$. As a result, E^{tot}_{2x} is at most unimodal; as such, E^{tot}_{2x} has at most a single maximum, and no other stationary points.* □

Theorem 2.4 can be proven by showing that if the error functions for either segment are concave, then the overall error function E^{tot}_{2x} is also concave.

Lemma 2.10 (Concavity by sum of errors). *If $E_{2(x+1)} - E_{2x} \leq E_{2x} - E_{2(x-1)}$ and $E'_{2(x+1)} - E'_{2x} \leq E'_{2x} - E'_{2(x-1)}$, then $E^{tot}_{2(x+1)} - E^{tot}_{2x} \leq E^{tot}_{2x} - E^{tot}_{2(x-1)}$.*

Proof. This follows from the general observation that if $A \leq C$ and $B \leq D$, then $A + B \leq C + D$. Alternatively, the observation that the sum of two concave functions is also concave can be used. □

[6]For a concave function, it holds that for a line between any two points underneath or on the curve, all points on the line are also underneath or on the curve.

From here, the case $m_0 < h$ will be considered, out of the two cases $m_0 < h$ and $m_0 > h$. The proof will be shown to be analogous for $m_0 > h$ in Section 2.7.2. Since it is assumed that $m_0 < h$, and all items adding to S are of height h, $m_x \leq h$ for any value of x.

2.5.1 Concavity of the error function for the first segment

To show $E_{2(x+1)} - E_{2x} \leq E_{2x} - E_{2(x-1)}$ for segment S, the error E_{2x} must first be determined at each step. To do so, two cases must be distinguished: $m_{2x} < h$ and $m_{2x} = h$. Note that since $m_0 < h$, it holds that $m_{2x} \leq h$, so there is no need to consider the case $m_{2x} > h$.

For $m_{2x} < h$, it holds that

$$E_{2x} = \sum_{i=1}^{\frac{n}{2}+x} (m_{2x} - a_i) + \sum_{i=\frac{n}{2}+x+1}^{n} (a_i - m_{2x}) + \sum_{i=1}^{2x} (h - m_{2x})$$

$$E_{2x} = -\sum_{i=1}^{\frac{n}{2}+x} a_i + \sum_{i=\frac{n}{2}+x+1}^{n} a_i + 2xh. \tag{2.19}$$

Note that the number of items covered by S is $n+2x$, and that the absolute-value operators in the error equation can be eliminated by observing $a_1, \ldots, a_{\frac{n}{2}+x} \leq m_{2x}$, and $a_{\frac{n}{2}+x+1}, \ldots, a_n \geq m_{2x}$. To see this, consider the following: since $m_0 < h$ and $m_0 = \frac{1}{2}a_{\frac{n}{2}} + \frac{1}{2}a_{\frac{n}{2}+1}$, any item of height h that has been moved from S' must have been inserted after $a_{\frac{n}{2}}$ in the list of items a ordered by height, since $a_{\frac{n}{2}} \leq a_{\frac{n}{2}+1}$ and their weighted sum is less than h. Similarly, if $m_{2x} < h$, then any items of height h are inserted after $a_{\frac{n}{2}+x}$, which as a result is the first term of the median equation, yielding $a_1, \ldots, a_{\frac{n}{2}+x} \leq m_{2x}$. $a_{\frac{n}{2}+x+1}$ is then either the second term of the median equation, or is at least equal to the second term of the median equation, resulting in $a_{\frac{n}{2}+x+1}, \ldots, a_n \geq m_{2x}$.

For the case that $m_{2x} = h$, however, a different definition is required, as the assumption that any item of height h that has been moved from S' to a must have been inserted after $a_{\frac{n}{2}}$ may no longer be valid. Therefore, a value q is chosen such that $a_1, \ldots, a_q < h$, and $a_{q+1}, \ldots, a_n \geq h$. The error E_{2x} can then be expressed as:

$$E_{2x} = \sum_{i=1}^{q} (m_{2x} - a_i) + \sum_{i=q+1}^{n} (a_i - m_{2x}) + \sum_{i=1}^{2x} (h - m_{2x})$$

$$E_{2x} = -\sum_{i=1}^{q} a_i + \sum_{i=q+1}^{n} a_i + (2q - n)h. \tag{2.20}$$

Note that in this case, the error E_{2x} no longer depends on x; since the subitems that are moved from S' are of the height $m_{2x} = h$, the error for these items is equal to 0, and as a result the overall error for S does not depend on the number of items of height h.

Since the definition of E_{2x} depends on whether $m_{2x} < h$ holds or not, there are three possible scenarios when considering the difference between the errors given

as $E_{2(x+1)} - E_{2x}$; for each of these scenarios, we require a separate computation of the error differences. The scenarios are: (1) $m_{2x} \leq m_{2(x+1)} < h$, (2) $m_{2x} < m_{2(x+1)} = h$, and (3) $m_{2x} = m_{2(x+1)} = h$. Below, it will be shown for each of these scenarios what the difference in error will be.

Lemma 2.11 (Error difference, case 1). *Given $m_{2x} \leq m_{2(x+1)} < h$, the difference in error $E_{2(x+1)} - E_{2x}$ is given as $2(h - a_{\frac{n}{2}+x+1})$.*

Proof. From (2.19), we have

$$\begin{aligned} E_{2(x+1)} - E_{2x} &= -\sum_{i=1}^{\frac{n}{2}+x+1} a_i + \sum_{i=\frac{n}{2}+x+2}^{n} a_i + 2(x+1)h \\ &\quad - \left(-\sum_{i=1}^{\frac{n}{2}+x} a_i + \sum_{i=\frac{n}{2}+x+1}^{n} a_i + 2xh \right) \\ &= -a_{\frac{n}{2}+x+1} - a_{\frac{n}{2}+x+1} + 2h \\ &= 2(h - a_{\frac{n}{2}+x+1}). \end{aligned} \tag{2.21}$$

□

Lemma 2.12 (Error difference, case 2). *Given $m_{2x} < m_{2(x+1)} = h$, the difference in error $E_{2(x+1)} - E_{2x}$ is equal to 0.*

Proof. Let q be such that $a_1, \ldots, a_q < h$, and $a_{q+1}, \ldots, a_n \geq h$. Using the fact that $m_{2x} < m_{2(x+1)} = h$, it can be determined that $q = \frac{n}{2} + x$ through the definition of the median. To see this, consider that since $m_{2x} < h$, the first term of m_{2x}, $a_{\frac{n}{2}+x}$ must be smaller than h. Since $m_{2(x+1)} = h$, and at least two items of height h are present, the first term of $m_{2(x+1)}$ must be equal to h. Therefore, it must hold that $a_{\frac{n}{2}+x+1} \geq h$, and $q = \frac{n}{2} + x$.

From (2.19) and (2.20), we have

$$\begin{aligned} E_{2(x+1)} - E_{2x} &= -\sum_{i=1}^{q} a_i + \sum_{i=q+1}^{n} a_i + (2q - n)h \\ &\quad - \left(-\sum_{i=1}^{\frac{n}{2}+x} a_i + \sum_{i=\frac{n}{2}+x+1}^{n} a_i + 2xh \right) \\ &= -\sum_{i=1}^{\frac{n}{2}+x} a_i + \sum_{i=\frac{n}{2}+x+1}^{n} a_i + (2(\frac{n}{2} + x) - n)h \\ &\quad + \sum_{i=1}^{\frac{n}{2}+x} a_i - \sum_{i=\frac{n}{2}+x+1}^{n} a_i - 2xh \\ &= 0. \end{aligned} \tag{2.22}$$

□

Lemma 2.13 (Error difference, case 3). *Given $m_{2x} = m_{2(x+1)} = h$, the difference in error $E_{2(x+1)} - E_{2x}$ is equal to 0.*

Proof. From (2.20), we have

$$\begin{aligned} E_{2(x+1)} - E_{2x} &= -\sum_{i=1}^{q} a_i + \sum_{i=q+1}^{n} a_i + (2q-n)h \\ &= -\left(-\sum_{i=1}^{q} a_i + \sum_{i=q+1}^{n} a_i + (2q-n)h \right) \\ &= 0. \end{aligned} \tag{2.23}$$

□

To next show the first part of Lemma 2.10, $E_{2(x+1)} - E_{2x} - (E_{2x} - E_{2(x-1)}) \leq 0$, once again a number of possibilities must be considered. Here, the four possible scenarios are: (1) $m_{2(x+1)} < h$, (2) $m_{2(x+1)} = h$ and $m_{2x} < h$, (3) $m_{2x} = h$ and $m_{2(x-1)} < h$, and (4) $m_{2(x-1)} = h$. Since the initial assumption is that $m_0 < h$, and only add items of height h to S, for all x it follows that $m_{2x} \leq m_{2(x+1)}$.

Next we show that the first part of Lemma 2.10 holds for each of the scenarios.

Lemma 2.14 (Error concavity for segment 1). *For all values of x, it holds that $E_{2(x+1)} - E_{2x} - (E_{2x} - E_{2(x-1)}) \leq 0$.*

Proof. To prove this, it must be shown that the lemma holds for all four possible cases: (1) $m_{2(x+1)} < h$, (2) $m_{2(x+1)} = h$ and $m_{2x} < h$, (3) $m_{2x} = h$ and $m_{2(x-1)} < h$, and (4) $m_{2(x-1)} = h$.

(1): $m_{2(x+1)} < h$. From (2.21), it follows that

$$\begin{aligned} E_{2(x+1)} - E_{2x} - (E_{2x} - E_{2(x-1)}) &= 2(h - a_{\frac{n}{2}+x+1}) - 2(h - a_{\frac{n}{2}+x}) \\ &= 2(a_{\frac{n}{2}+x} - a_{\frac{n}{2}+x+1}) \leq 0. \end{aligned}$$

(2): $m_{2(x+1)} = h$ and $m_{2x} < h$. From (2.21) and (2.22), and the observation that $a_{\frac{n}{2}+x} < h$ (since $m_{2x} < h$), it follows that

$$E_{2(x+1)} - E_{2x} - (E_{2x} - E_{2(x-1)}) = 0 - 2(h - a_{\frac{n}{2}+x}) \leq 0.$$

(3): $m_{2x} = h$ and $m_{2(x-1)} < h$. From (2.22) and (2.23), and since $m_{2(x+1)} = h$, it can be seen that

$$E_{2(x+1)} - E_{2x} - (E_{2x} - E_{2(x-1)}) = 0 - 0 \leq 0.$$

(4): $m_{2(x-1)} = h$. From (2.23), and since $m_{2(x+1)} = m_{2x} = h$, it can be seen that

$$E_{2(x+1)} - E_{2x} - (E_{2x} - E_{2(x-1)}) = 0 - 0 \leq 0.$$

□

2.5.2 Concavity of the error function for the second segment

In a similar manner, it can be shown that the error function E'_{2x} for segment S' is concave. As was the case for segment S, there are once more two cases for the median m'_{2x} that must be distinguished between: $m'_{2x} < h$ and $m'_{2x} = h$. Here, the implicit assumption is made that $m'_w < h$ - the proof for the opposite case can be done analogously, as discussed in Section 2.7.2. Given that $m'_w < h$, for any x, it follows that $m'_{2x} \leq h$, and $m'_{2x} \leq m'_{2(x-1)}$.

For the case $m'_{2x} < h$, the error E'_{2x} can be defined in similar fashion as described in Section 2.5.1.

$$E'_{2x} = \sum_{i=1}^{\frac{n'}{2}+\frac{w}{2}-x} (m'_{2x} - b_i) + \sum_{i=\frac{n'}{2}+\frac{w}{2}-x+1}^{n'} (b_i - m'_{2x}) + \sum_{i=1}^{w-2x} (h - m'_{2x})$$

$$E'_{2x} = -\sum_{i=1}^{\frac{n'}{2}+\frac{w}{2}-x} b_i + \sum_{i=\frac{n'}{2}+\frac{w}{2}-x+1}^{n'} b_i + (w-2x)h. \tag{2.24}$$

For the case $m'_{2x} = h$, a value q' is once again defined such that $b_1, \ldots, b_{q'} < h$, and $b_{q'+1}, \ldots, b_{n'} \geq h$. Then, the error can be defined as follows:

$$E'_{2x} = \sum_{i=1}^{q'} (m'_{2x} - b_i) + \sum_{i=q'+1}^{n'} (b_i - m'_{2x}) + \sum_{i=1}^{w-2x} (h - m'_{2x})$$

$$E'_{2x} = -\sum_{i=1}^{q'} b_i + \sum_{i=q'+1}^{n'} b_i + (2q' - n')h. \tag{2.25}$$

Like before, there are again three possible scenarios for the difference in errors $E'_{2(x+1)} - E'_{2x}$. The three scenarios are: (1) $m'_{2(x+1)} \leq m'_{2x} < h$, (2) $m'_{2(x+1} < m'_{2x} = h$, and (3) $m'_{2(x+1)} = m'_{2x} = h$. For each of these scenarios, the error can once again be determined as:

Lemma 2.15 (Error difference, case 1). *Given $m'_{2(x+1)} \leq m'_{2x} < h$, the difference in error $E'_{2(x+1)} - E'_{2x}$ can be determined as $2(b_{\frac{n'}{2}+\frac{w}{2}-x} - h)$.*

Proof. From (2.24), it can be seen that

$$\begin{aligned} E'_{2(x+1)} - E'_{2x} &= -\sum_{i=1}^{\frac{n'}{2}+\frac{w}{2}-x-1} b_i + \sum_{i=\frac{n'}{2}+\frac{w}{2}-x}^{n'} b_i + (w-2x+2)h \\ &\quad - \left(-\sum_{i=1}^{\frac{n'}{2}+\frac{w}{2}-x} b_i + \sum_{i=\frac{n'}{2}+\frac{w}{2}-x+1}^{n'} b_i + (w-2x)h \right) \\ &= 2(b_{\frac{n'}{2}+\frac{w}{2}-x} - h). \end{aligned} \tag{2.26}$$

□

Lemma 2.16 (Error difference, case 2). *Given $m'_{2(x+1} < m'_{2x} = h$, the difference in error $E'_{2(x+1)} - E'_{2x}$ is equal to* 0.

Proof. Let q' be such that $b_1, \ldots, b_{q'} < h$, and $b_{q'+1}, \ldots, b_{n'} \geq h$. Using similar reasoning as described in Lemma 2.12, along with the observation that we have $m'_{2(x+1)} < m'_{2x} = h$, it can be determined that $q' = \frac{n'}{2} + \frac{w}{2} - x - 1$. Using (2.24) and (2.25), the difference in error can then be found as

$$\begin{aligned}
E'_{2(x+1)} - E'_{2x} &= -\sum_{i=1}^{\frac{n'}{2}+\frac{w}{2}-x-1} b_i + \sum_{i=\frac{n'}{2}+\frac{w}{2}-x}^{n'} b_i + (w-2x+2)h \\
&\quad - \left(-\sum_{i=1}^{q'} b_i + \sum_{i=q'+1}^{n'} b_i + (2q'-n')h \right) \\
&= -\sum_{i=1}^{\frac{n'}{2}+\frac{w}{2}-x-1} b_i + \sum_{i=\frac{n'}{2}+\frac{w}{2}-x}^{n'} b_i + (w-2x+2)h \\
&\quad + \sum_{i=1}^{\frac{n'}{2}+\frac{w}{2}-x-1} b_i - \sum_{i=\frac{n'}{2}+\frac{w}{2}-x}^{n'} b_i - (2(\frac{n'}{2} + \frac{w}{2} - x - 1) - n')h \\
&= 0. \qquad (2.27)
\end{aligned}$$

□

Lemma 2.17 (Error difference, case 3). *Given $m'_{2(x+1)} = m'_{2x} = h$, the difference in error $E'_{2(x+1)} - E'_{2x}$ is equal to* 0.

Proof. From (2.25), it follows that

$$\begin{aligned}
E'_{2(x+1)} - E'_{2x} &= -\sum_{i=1}^{q'} b_i + \sum_{i=q'+1}^{n'} b_i + (2q'-n')h \\
&\quad - \left(-\sum_{i=1}^{q'} b_i + \sum_{i=q'+1}^{n'} b_i + (2q'-n')h \right) \\
&= 0. \qquad (2.28)
\end{aligned}$$

□

To show that the second part of Lemma 2.10 holds, or in other words, that it holds that $E'_{2(x+1)} - E'_{2x} \leq E'_{2x} - E'_{2(x-1)}$, this must once again be proven for four possible scenarios: (1) $m'_{2(x-1)} < h$, (2) $m'_{2(x-1)} = h$ and $m'_{2x} < h$, (3) $m'_{2x} = h$ and $m'_{2(x+1)} < h$, and (4) $m'_{2(x+1)} = h$. Below, it will be shown that for all four scenarios, it holds that $E'_{2(x+1)} - E'_{2x} - (E'_{2x} - E'_{2(x-1)}) \leq 0$.

Lemma 2.18 (Error concavity for segment 2). *For all values of x, it holds that* $E'_{2(x+1)} - E'_{2x} - (E'_{2x} - E'_{2(x-1)}) \leq 0$.

Proof. To show this, it must be proven that this holds for all four possible cases of median values: (1) $m'_{2(x-1)} < h$, (2) $m'_{2(x-1)} = h$ and $m'_{2x} < h$, (3) $m'_{2x} = h$ and $m'_{2(x+1)} < h$, and (4) $m'_{2(x+1)} = h$.

(1): $m'_{2(x-1)} < h$. From (2.26), it follows that

$$\begin{aligned} E'_{2(x+1)} - E'_{2x} - (E'_{2x} - E'_{2(x-1)}) &= 2(b_{\frac{n'}{2}+\frac{w}{2}-x} - h) - 2(b_{\frac{n'}{2}+\frac{w}{2}-x-1} - h) \\ &= 2(b_{\frac{n'}{2}+\frac{w}{2}-x} - b_{\frac{n'}{2}+\frac{w}{2}-x-1}) \leq 0. \end{aligned}$$

(2): $m'_{2(x-1)} = h$ and $m'_{2x} < h$. From (2.26) and (2.27), it follows that

$$E'_{2(x+1)} - E'_{2x} - (E'_{2x} - E'_{2(x-1)}) = 2(b_{\frac{n'}{2}+\frac{w}{2}-x} - h) - 0 \leq 0.$$

Note that here, $q' = \frac{n'}{2} + \frac{w}{2} - (x-1) + 1$.

(3): $m'_{2x} = h$ and $m'_{2(x+1)} < h$. From (2.26) and (2.27), it follows that

$$E'_{2(x+1)} - E'_{2x} - (E'_{2x} - E'_{2(x-1)}) = 0 - 0 \leq 0.$$

(4): $m'_{2(x+1)} = h$. From (2.27), it follows that

$$E'_{2(x+1)} - E'_{2x} - (E'_{2x} - E'_{2(x-1)}) = 0 - 0 \leq 0.$$

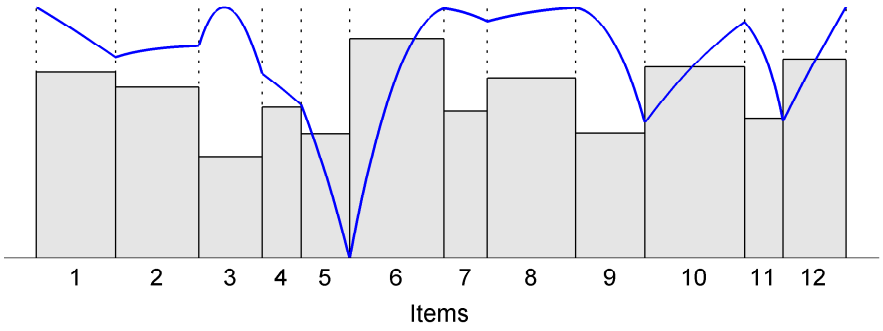
□

By combining the results of Lemmas 2.14 and 2.18, it follows that Lemma 2.10 holds, and as a result, the main Theorem 2.4.

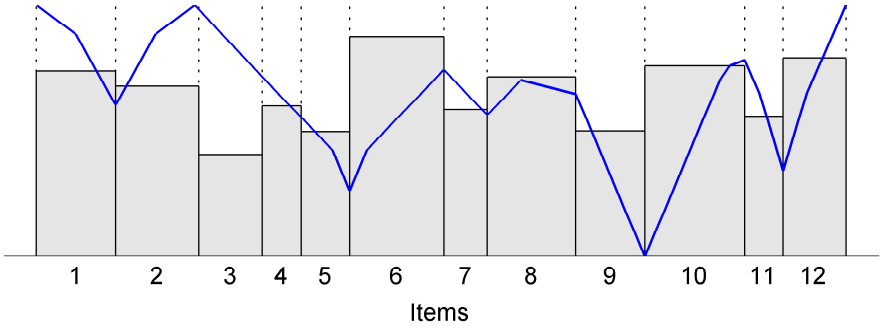
2.6 Examples of the L1 and L2 error behavior

In this section, we briefly discuss some examples of the behavior of the error functions discussed in Sections 2.4 and 2.5. In Figure 2.7, the normalized segmentation errors are shown for a sequence of items for all possible 2-segmentations. This also includes the segmentations that ignore the alignment requirement. In particular, Figure 2.7a shows the normalized error for the L2 norm, while Figure 2.7b shows the normalized error for the L1 norm. Here, we have chosen to use the normalized error rather than the actual error for visibility reasons; the normalization step consists of scaling the error between zero and one.

For both of the simulated examples in Figure 2.7, we can see that the normalized error functions indeed have at most a single maximum between each pair of item boundaries, and that the minimum values indeed correspond to the item boundaries. We can also note that for the L2 error criterion in Figure 2.7a, the error function shows a smooth curve between item boundaries, while the L1 error function in Figure 2.7b appears to be piece-wise linear. This is not surprising considering the nature of the



(a) Example of the normalized segmentation error for the L2 error criterion on a 2-segmentation.



(b) Example of the normalized segmentation error for the L1 error criterion on a 2-segmentation.

Figure 2.7: Examples of the behavior of the L2 and L1 error functions when considering a 2-segmentation of the provided items, also including potential segmentations ignoring the alignment requirement. The segmentation error for a specific segmentation boundary is shown by the blue lines.

median function, where values tend to 'jump' to a new median as the item coverages shift, in contrast to the smoother changes we would observe in the mean function.

2.7 Generalization to real-valued durations

In this section, it is briefly discussed how the proofs given in Sections 2.4 and 2.5 can be extended to sequences of items with real-valued durations, and discuss some of the initial assumptions made in the proofs.

2.7.1 Generalizations for the L2 error case

If the assumption is made that the durations w_i are allowed to have arbitrary real-valued durations, this introduces some additional notation. In particular, the total durations of the segments are given by $\sum_{i=1}^{n} w_i$ and $\sum_{i=1}^{n'} w'_i$, instead of n and n', respectively. For example, instead of $\mu = \frac{\sum_{i=1}^{n} h_i}{n}$, it would be possible to write

$$\mu = \frac{\sum_{i=1}^{n} w_i h_i}{\sum_{i=1}^{n} w_i}.$$

Similarly, (2.3) would become

$$\delta(x) = \frac{wx}{\sum_{i=1}^{n} w_i + wx}(h - \mu).$$

Removing the alignment requirement for other segments similarly introduces additional notation. More specifically, as items may reside in multiple segments, the additional parameters f_i and f'_i are introduced, representing the fraction of w_i or w'_i which belongs to the segments S and S', respectively. For example, in this case μ would be given as

$$\mu = \frac{\sum_{i=1}^{n} f_i w_i h_i}{\sum_{i=1}^{n} f_i w_i}.$$

In both cases, there are no major deviations from the proof given in Section 2.4, and a similar proof can be provided for the above cases analogously to the one given here.

2.7.2 Generalizations for the L1 error case

The proof in Section 2.5 is based on the initial assumptions that $m_0 < h$, and $m'_w < h$. The case of $m_0 = h$ is trivial, as in this case there is no difference in error regardless of the value of x. To show that the proof is still valid for other initial conditions, such as $m_0 > h$, it can be observed from Lemma 2.10 that when changing one of these initial assumptions, one only needs to reconsider the segment for which the assumption was changed. In case of $m_0 > h$, the same reasoning given in Section 2.5.1 can be followed, provided the items in S are sorted in descending, rather than ascending, order. The reasoning is similar for segment S'.

In addition, there is the assumption that n, n' and w are even. To show that the proof holds if one or more of these values are odd, consider that each of the $n+n'+w$ items can be split into two subitems of the same height and half unit duration, and then redefine n, n' and w accordingly. The proof can then be applied to the redefined segments directly, which are identical to the original segments.

A similar approach can be used to show that the proof still holds when a single item is moved to S at each step x, as opposed to two items at a time. In addition, the proof can be extended to durations of fractional numbers by splitting the items of unit duration into subitems of a duration equal to the lowest common denominator of the fractional numbers.

2.8 Conclusions

The results in this chapter show that for any Lp error criterion, an optimal k-segmentation can be found in polynomial time even when items do not have unit duration and segment boundaries need not coincide with item boundaries. In particular, the segmentation error is minimal at the item boundaries, and as such, there is no advantage to be gained (in terms of error) by not adhering to the alignment requirement. In addition, we have shown that for the L1 and L2 error criteria, the segmentation error as a function of the coverage of an item contains at most a single maximum, and no other stationary points.

In this chapter, we addressed the segmentation component of the main research question, showing that the methods for solving the k-segmentation problem can also be applied to sequences of items of non-unit duration, or where the alignment requirement need not hold. While there exist many segmentation methods, algorithms for k-segmentation are often applied in the field of human motion tracking and outside of it, particularly for applications where we have a good idea of the number of segments we are looking for.

In the proofs provided here, the primary focus has been on the L1 and L2 norms as error criteria. These norms have the desirable properties that they respectively minimize the absolute and squared (or Euclidian) error between the items and their representatives, similar to for example the least squares algorithm used in linear regression. As a result, the representatives of the L1 error criterion correspond to the segment median, while the representatives of the L2 error criterion correspond to the segment mean. Note that while we are not always interested in differences in the means or medians of our raw measurements, segmentation can also be applied on the measurement features, such as the measurement intensity or variance.

A limitation of k-segmentation algorithms is that they can naturally only be applied to a finite signal, where we have some indication of the number of segments of interest. If processing time allows however, it is possible to compute the error for multiple values of k and select the optimal value. For some applications, it is fairly straightforward to find the value of k; for example, finding segments of cycling activity for people who cycle to and back from work each day. Another limitation of k-segmentation, and of using the L1 and L2 norms as error criteria, is that it can be difficult to include domain-specific knowledge on the sequence segments (for example, if we know certain item values often occur near segment boundaries). In such cases, a problem-specific segmentation method of often devised.

3

Activity recognition

3.1 Introduction

The work described in this chapter has been previously published and discussed [Pijl, van de Par, and Shan, 2010; Dadlani et al., 2013], and related to a number of other publications [de Ruyter et al., 2011; Pijl et al., 2014].

At a basic level, our ability to remain healthy relies on being able to perform certain activities such as eating, ambulation and maintaining hygiene. These types of activities are often referred to as "activities of daily living" (ADL). Our ability to perform these basic activities is strongly related to our functional independence - that is, losing the ability to perform some or all of these ADLs will result in dangerous or unhealthy situations, or even death, without some form of external assistance. This can be a particular risk for elderly living independently, as ADLs can become difficult to perform due to physical or cognitive decline. Unobtrusive activity monitoring can be helpful in such cases, by preventing the need for pre-emptive institutionalization, by providing peace of mind to family members and caregivers, and by serving as input for context-aware assistive systems.

However, monitoring ADLs is no trivial matter - the field of human activity recognition is fraught with challenges, ranging from gathering accurate and relevant data to dealing with the large variation in human activities. Furthermore, many of the more complex activities cannot be easily distinguished from one another by a single identifying feature, but only by observing a sequence of successive actions. In addition, there exist considerable differences in how a given activity is performed by different

individuals, making activity recognition more challenging across individuals. While a more complex activity will often be made up of the same set of actions, it is entirely feasible that these actions can be performed in a different order to the same effect, or that some actions may be repeated multiple times, or omitted altogether.

In this chapter, we aim to address the first research question related to the main research question, introduced in Chapter 1: Can activities of daily living (ADL) be unobtrusively tracked and recognized? While doing so, we also address the first two components of the model for human motion tracking described in the main research question: segmentation and classification. To classify the ADLs based on human motion data, we simultaneously attempt to find the segment boundaries that mark the transition from one ADL to the next.

To achieve this, we present an examination of human activity data gathered using a single fixed camera and a single microphone in a kitchen environment. This data is employed in the creation of a number of models to classify human activity both after the activity has been completed and while the activity is being performed. Rather than attempting to distinguish between all possible activities, six activities of interest have been selected, with the focus on activities which last for an extended period of time. In other words, the focus is on activities similar to, for example, watching television, rather than recognizing a hand gesture. As the single camera and microphone have limitations with regard to how much can be observed, the activities examined in this chapter are all set within the context of a single room. For the experiments described in this chapter, a kitchen setting was chosen, as this setting allows for a number of extended activities distinct from one another, yet by and large making use of the same facilities the setting provides. Despite selecting a single setting, the methods described here can be applied to any interior setting, or even extended to exterior settings. In total, six activities were chosen to be recognized: storing groceries, preparing dinner, eating, doing dishes, vacuuming, and preparing a drink.

The method proposed in this chapter makes use of two separate classification layers. In the first layer, audio and video scene analysis techniques are used to detect the presence of singular events, such as the location of a person in the room or the presence of a specific sound. This chapter focuses on the second layer; activity recognition using these events as input. For each of the activities to be recognized, a model is created by constructing a Hidden Markov Model (HMM). HMMs are explained in more detail in Section 3.2. Activities are then classified using a history of recently observed events.

There have been numerous studies in the field of human activity recognition, of which ADL detection can be seen as a special case [Bao and Intille, 2004; Chen et al., 2007; van Kasteren et al., 2008; Maurer et al., 2006; Karaman et al., 2014; Konig et al., 2015; Poularakis et al., 2015]. The majority of these studies fall into one of two categories previously discussed in Chapter 1: the use of body-worn sensors to

capture ADLs, or the use of one or more sensors placed in the environment. Examples of the former include the use of accelerometers or heart rate monitors, while examples of environmental sensors used for activity recognition can include RFID sensors, cameras, contact switches and the like. A second distinction can be made between sensors based on the depth of information they provide; for example, a simple pressure switch may only report one of two values at any time (pressed - not pressed). On the other hand, a single camera frame potentially provides thousands of pixels' worth of information. Such sensors are sometimes referred to as low-level and high-level sensors, respectively.

For ADL detection in particular, this distinction is important because different methodological approaches are required for low-level and high-level sensors. High-level sensors require advanced scene analysis algorithms to extract relevant features from the wealth of data they provide, and data fusion algorithms to combine various streams of data. On the other hand, low-level sensors are easy to interpret but often only provide part of the information needed for ADL detection - an issue that may be solved by employing them strategically and in numbers. Here, the challenge is to combine the (often many) different outputs of the individual sensors to arrive at sensible conclusions - sometimes at varying levels of sensor data complexity (a task for which rule engines or ontologies are often employed).

Regardless of the types of sensors used - wearable or environmental, low-level or high - there is often a need to combine multiple sensors, either of the same type, or as a combination of different modalities. In many cases, a single sensor modality in insufficient to obtain the level of context needed to distinguish between complex activities such as ADLs. As a result, the concept of sensor networks often crops up in ADL monitoring. Sensor networks consist of multiple sensors connected to each other, usually through wireless communication protocols. This allows the data collected by multiple sensors to be shared within the network, and activities to be recognized in a more or less real-time fashion. The alternative is to use a central hub that collects the output from the individual sensors. The advantages of a sensor network are that the sensor output is provided in some standardized fashion for all sensors, and that adding new sensors to the network is relatively straightforward. This makes sensor networks particularly advantageous when large numbers of sensors are involved.

Activity recognition using wearable sensors. When it comes to activity recognition using wearable sensors, accelerometers are arguably the most popular choice if current literature is any indication. Bao and Intille [2004] use a network of up to five accelerometers placed on the body to distinguish between 20 activities, including walking, running, vacuuming, eating, brushing teeth, and so on. Bao and Intille [2004] report an overall recognition accuracy of over 80% using decision trees on temporal and spectral features. An interesting aspect of the study is that it in-

cludes both complex activities such as brushing teeth, and more basic activities such as standing up and sitting down.

In general, it can be argued that a certain activity is more complex than another if it can include those others as a subactivity. For example, the process of brushing teeth can include walking to the sink, turning on the tap, and so in. Another (albeit less well-defined) way to view complex and basic activities would be the observation that the complex activities tend to represent the behavioral intentions or aims of an individual. That is, the intension of an individual may be to brush their teeth; walking to the sink is merely a means to that end in this case. Activities like walking or standing up are rarely performed purely for their own merit[1].

From this, we can infer a kind of hierarchical ordering where activities and subactivities exist as part of higher-level activities and goals. Of course, this is somewhat of a simplification of reality, where the distinction between activities and subactivities will not always be so clear. In addition, we often tend to combine multiple activities in our daily lives, such as having a phone conversation while walking to the grocery store. We may therefore encounter subactivities as a result of a secondary task that we would otherwise not expect. Even so, the hierarchical ordering of activities can be beneficial from a modeling perspective.

Activity recognition using body-worn sensors is often focused on the recognition of the relatively more basic activities. Nuynh and Schiele [2005] describe a study where participants perform the activities walking, standing, jogging, hopping, skipping, and riding a bus. Data was recorded using a sensor attached to a backpack strap, which recorded 3D acceleration, along with other modalities such as light, audio and temperature. Different window sizes and features were examined for the optimal recognition of each activity. In particular, features derived from fast Fourier transforms (FFT) were used with k-means clustering to find activity clusters. Recognition for jogging and walking performed particularly well, with accuracies of about 90%.

Maurer et al. [2006] have integrated a set of similar sensors into a wristwatch for the recognition of walking, standing, running, sitting, and walking stairs. Recording devices were placed in locations where one might normally carry a cell phone; that is, in various pockets, attached to the belt, in a bag, and around the neck[2]. The recording devices measured acceleration and light. Using features primarily based on correlation, performances ranging from 17% to 93% were reported based on the sensor modalities used and the wearing position.

Ravi et al. [2005] investigate a similar set of activities as Maurer et al. [2006], but include the activities sit-ups, vacuuming and brushing teeth in favor of sitting.

[1] One could argue that such activities can be performed for their own merit within the context of exercise - however, one could also argue that the exercise itself (e.g., jogging) is the complex activity in this case.

[2] While one might argue that this is an unusual position to carry a cell phone, placing sensors in, for example, a pendant around the neck is not uncommon.

Activity recognition is performed using a single accelerometer attached to the pelvic region. Here, both correlation-based features and features derived from FFT were investigated together with a number of classification algorithms; decision tables, decision trees, *k*-nearest neighbor, support vector machines, and naive Bayesian classifiers. Across participants, a maximum accuracy was obtained of 73% using support vector machines.

As already seen from the work by Bao and Intille [2004], it is also possible to combine the measurements from multiple accelerometers at different wearing positions. Gao, Bourke, and Nelson [2014] describe a comparison between systems using a single or multiple accelerometers for detecting activities such as walking, stair climbing, and sitting down or standing up (for various surfaces). They report improved accuracy for the multiple sensors setup, with 96% accuracy compared to the 93% accuracy of the single sensor system. A downside of such a multi-sensor system is that wearing a larger number of sensors is a larger burden for the user, as well as a potential increase in computational requirements and power consumption. Gao, Bourke, and Nelson [2014] however counter that sensors can be integrated into clothing, and more lightweight algorithms can be used due to the benefits of multiple sensor data sources. Whether this makes up for the additional sensor's power consumption, and how feasible it is to integrate sensors in all items of clothing, remains up for debate.

Lee and Mase [2002] discuss the use of multiple sensors containing an accelerometer and gyroscope to detect both a set of activities (walking, standing, sitting, moving up or down stairs), as well as their location within an environment (such as at their desk, printer, and so on). They use acceleration and rotation features in a dead reckoning approach to estimate activity and location. Exact accuracy numbers are not provided, but Lee and Mase [2002] note that the activity recognition performance was satisfactory in the number of behaviors missed by the algorithm.

A more expansive overview of the application of accelerometers and inertial sensors in this field, and the accompanying methodologies can for example be found in the work by Bulling, Blanke, and Schiele [2014]. Accelerometers are not the only type of body-worn sensor that can be used for activity recognition however; Philipose et al. [2004] and Patterson et al. [2005] make use of an RFID sensor attached to a glove to distinguish between a number of complex activities focused on the kitchen environment, which include making tea, and setting the table (in total, they look at 14 and 11 different activities, respectively). Many of the objects in the kitchen are tagged with RFID tags, generating a signal when they are in close proximity to the glove. A number of activity recognition approaches are described, based on HHMs and dynamic Bayesian networks. Philipose et al. [2004] describe an average precision of 88% and recall of 73%, while Patterson et al. [2005] describe an average accuracy of 81%.

While cameras would traditionally be considered an environmental sensor, miniaturization has allowed cameras to be used as wearable sensors for the purpose of activity recognition, as described by Karaman et al. [2014][3]. Here, a hierarchical hidden Markov model[4] is used to distinguish between 23 activities of daily living, including washing clothes, knitting, and brushing teeth. To do so, video segments where the camera retains the same view are automatically extracted, and from these a number of motion-related and positional features are derived, that are further combined with a number of audio features. The resulting model can distinguish between the 23 activities with a median accuracy of 42%.

Activity recognition using environmental sensors. In terms of environmental sensors, cameras are an often used tool in activity recognition and human movement tracking through the use of computer vision techniques; an overview of the methods used in this field can for example be found in the work of Ke et al. [2013], Gavrilla [1999], and Aggarwal and Cai [1997]. In many studies, camera images are used to determine the possible behavior of or interaction between individuals. More recently, the use of 3D camera setups has also been considered [Aggarwal and Xia, 2014]. Typical activities to be recognized in this context are, for example, a person passing by, two people meeting, and so on. The applications of this type of activity recognition include security [Hongeng, Bremond, and Nevatia, 2000] and human-machine interaction in for example artificial intelligence applications [Kelley et al., 2008]. There have also been investigations into the derivation of more basic activities using camera images [Madabhushi and Aggarwal, 1999], where activities such as sitting down or standing up are examined (in total, 9 activities were examined). Activities were determined through tracking the movement patterns of a person's head, yielding an accuracy of 80% through classification using a Bayesian framework.

An example of the use of cameras to track activities of daily living is described by Konig et al. [2015]. Here, a study is presented where two cameras placed in the environment are used to determine whether a number of ADLs are performed, and additionally kinetic-based features are derived from the camera images to assess how well the ADLs were performed. The aim is to use such a system to provide early indicators for dementia or cognitive decline in seniors. The reported performance of the system in detecting the various ADLs, which include activities such as having a phone call, preparing a pill box, and making tea, was an overall sensitivity of 85% and a precision of 76%. Another example of the tracking of ADLs using a camera is described by Poularakis et al. [2015]. Here, a method based on support vector

[3] Although the camera used in this study is perhaps not the epitome of unobtrusiveness, relatively small cameras of high quality are available, as evidenced by modern smartphones. In studies, requirements in terms of data storage, reliability and battery life can often be more strict than in an eventual application.

[4] Basically a hidden Markov model of hidden Markov models; also see the notes in Section 3.5.

machines is used on various data sets for various settings, achieving accuracies of around 90%.

Video images have also been coupled to audio as an additional modality by Chen et al. [2007], to detect social interactions between elderly in a nursing home environment. This includes standing conversations, interacting, passing by in the hall, and assisting someone with walking. The study included both 'wizard of Oz' sensors and actual sensor recordings. On the latter, classification was attempted using a dynamic Bayes network, decision trees, support vector machines and logitboost, with the dynamic Bayes network yielding the best classification accuracies ranging from 86% to 94% depending on the activity. Hidden Markov models were used to model a number of subactivities (later used in classification of the main activities), including 'approaching' and 'passing'.

Oliver, Horvitz, and Garg [2002] use audio and video, along with keyboard and mouse interactions, to track interactions in the office space - this includes the activities phone conversation, giving a presentation, face-to-face conversation, user present, nobody present, and distant conversation. Here, a layered hidden Markov model is used, a variation on the standard HMM. The layered HMM includes two separate HMMs for processing audio and video features, respectively. The classification results of this layer are fed to a third HMM for activity classification[5]. Oliver, Horvitz, and Garg [2002] report an accuracy on activity sequences of over 99% using the layered HMM.

Apart from studies focusing on more high-level sensors such as audio and video, activity recognition studies have also been performed using multiple low-level sensors. A good overview of some of the sensors and techniques used in the smart home (i.e., a home fitted with environmental sensors) context can be found in the work by Debes et al. [2016]. In addition to this overview, Debes et al. [2016] also apply a number of the described methods on data sets from several households; concluding that overall the Fisher kernel learning method outperformed more 'traditional' methods such as HMMs for this application.

Viard et al. [2016] do conclude that good results can be obtained in such a setting using HMMs, using a large amount of binary sensors (77) placed inside the home. Viard et al. [2016] comment that an advantage of their HMM approach is that it allows training for an individual user without knowledge of the real actions performed during the learning phase, and provide a confusion matrix of the classification results for activities such as eating, dressing, and doing dishes. A limitation of the study is that it was performed with a single participant.

[5]Layered hidden Markov models are comparable to the chained hidden Markov models described in Section 3.2.4, in the sense that they can be decomposed into Cartesian hidden Markov models, and use a particular architecture (also) to reduce the dimensionality of the model.

The work of van Kasteren et al. [2008] describes the use of multiple binary switch sensors for ADL recognition in the home. A total of 14 sensors were placed, with locations including doors, cupboards, the refrigerator, and toilet. HMMs and conditional random fields (a method somewhat related to the HMM) were explored as a means to fuse the various sensor outputs. A total of 8 activities were considered, including toileting, showering, eating breakfast, and eating dinner. Using HMMs, a class accuracy of 79% was achieved using offline processing, and 73% using online processing.

Rashidi and Cook [2009] use sensors embedded in a home environment to find frequent and periodic patterns of activity by discovering sequences of observed events. The sensor modalities used include motion, temperature and light sensors. Participants were asked to follow a script of activities, and include some random (non-scripted) activities as well. From the discovered patterns, Rashidi and Cook [2009] conclude that the script's patterns were discovered correctly. Interestingly, a discussion is also included on how feedback of the user on the recognized patterns of activity can be used to improve the performance of the system.

Urwyler et al. [2015] make use of a number of sensor boxes that can be placed around the home, each included five types of sensors, to detect ADLs. Here, the naive Bayesian classifier and random forest algorithms are explored, as well as introducing two 'ad-hoc' algorithms based on rule inference and circadian activities rhythm. Urwyler et al. [2015] note that the introduced ad-hoc algorithms outperform the traditional algorithms in most cases.

Another approach to the smart home is the use of ultrasonic sensors placed on the ceiling in a grid-like pattern, as described by Okour, Maender, and Basilakis [2015]. Here, the primary focus is on the positional aspects of activities, such as sitting, sleeping, walking straight or 'walking curvy'. To distinguish between these activities, Okour, Maender, and Basilakis [2015] employ a rule-based classifier combined with finite-state machines to ensure that the detected activities follow each other logically. The achieved accuracies ranged from 88% to 99% depending on the activity.

Literature discussion. When taking into account the extensive literature on the topic of human activity recognition, there are a few conclusions we can draw with regard to the field. The first conclusion is that the accuracies of the activity recognition methods reported often vary considerably between studies. The primary causes of this can be described as the activities investigated, the sensor modalities used, and the definitions of accuracy employed.

It appears that some activities are inherently easier to distinguish than others; in particular, studies focusing on more low-level activities such as sitting or walking generally report better accuracies compared to the recognition of ADLs. Naturally, the overall accuracy is also affected by the number of activities taken into consideration. The wide variety of sensor modalities investigated, as well as differences in the number of sensors (and their location), also complicate the direct comparison of

many study results; in particular, we find few multi-modal studies where the impact of the individual modalities is critically evaluated. Finally, we encounter many different (often implicit) definitions of accuracy that are used; for example, there is a noticeable gap in reported accuracy between studies performed on manually segmented activities and studies on continuous measurements containing multiple activities.

Taking this into account, we overall see higher accuracies reported for the recognition of lower-level activities compared to the recognition of ADLs; we can conclude from this that the recognition of more complex activities is therefore a more complex problem. In general, there seem to be three popular approaches to human activity recognition: approaches based in computer vision using cameras, wearable sensors, and the use of multiple (and often large numbers of) environmental sensors. Here, the wearable sensor approach is mostly used for low-level activities. It is notable that there are few environmental solutions that both include multiple modalities, and that would fit inside a single unit, similar to the sensor boxed used by Urwyler et al. [2015]; such a solution would have an advantage in terms of installation compared to other environmental solutions, and would be less obtrusive compared to many wearable sensor solutions.

It is also interesting to note that the methodologies used seem to have changed little over the years; the use of algorithms such as hidden Markov models, k-nearest neighbor, support vector machines, and neural networks remains popular [Gao, Bourke, and Nelson, 2014; Viard et al., 2016; Poularakis et al., 2015; Karaman et al., 2014]. This is perhaps not surprising given that these algorithms seem to provide acceptable levels of performance for most applications. Recently, random forests have also been applied to human activity recognition [Urwyler et al., 2015]. In the coming years, we may additionally see an increase in methods that combine the properties of generative models (such as HMMs) and discriminative models (such as support vector machines), as described by Debes et al. [2016].

In this chapter, we make use of the well-described HMM to detect ADLs in a kitchen environment from a single audio and video sensor. As such, we are looking at a system that will fit into a single 'box'. The aim is to accurately distinguish between six ADLs (storing groceries, preparing dinner, eating, doing dishes, vacuuming, and preparing a drink), using sensor data sequences recorded during a study where participants were asked to perform each of the listed ADLs.

We will examine the classification results from two perspectives: first, we will look at the classification accuracy when the start and end times are assumed to be known (sequences that are already segmented). Second, we will look at the classification accuracy over the entire recorded sessions, without any knowledge of the start and end times of the individual ADLs (unsegmented sequences). In addition, we will also address the relevance of the individual sensor modalities used, and investigate a number of different modeling and training schemes for the HMM.

The choice of HMMs for this work was based on a number of properties of the HMM that match well with the problem described. First, the different states of the HMM, and the ability to traverse these in a multitude of orders with some probability, match with the nature of the ADLs being performed; in particular, we theorize that at least at a conceptual level, ADLs can be divided into a number of subactivities, that do not necessarily need to be performed in a linear fashion. Many of the smaller actions that make up the ADL that are performed at a given time depend on the current subactivity. As such, we expect a strong dependence on the underlying subactivity and the events observed, similar to the hidden state – visible state paradigm of the HMM.

Second, our observations made on the subactivities performed allow us to mitigate one of the weaknesses of the hidden Markov model: the presence of a large parameter-space to estimate. Based on our observations, we can initialize our models based on statistical observations, reducing the parameter estimation problem. Third, HMMs are effective at integrating data from multiple sources, when they are (for example) recorded at different sampling frequencies. Finally, the use of HMMs allows us to implicitly derive segment boundaries of the ADLs in a sequence of continuous measurements.

The remainder of this chapter is organized as follows: in Section 3.2, the hidden Markov model architecture, algorithms and application are described, as well as a number of alternative model construction techniques. Section 3.3 describes the experimental setup and methods used to obtain human activity data. Section 3.4 discusses the theoretical performance of the method on both fully observed activities and partly observed activities. Section 3.5 describes method performance in a more practical application where activities are recognized as they are being performed, using a short history of past observed events. Finally, some concluding remarks are given in Section 3.6.

3.2 Modeling activities

A hidden Markov model (HMM) is a state-space (or stochastic) model that consists of a set of hidden states (or simply states) and a set of observable states, often referred to as emissions. For a given HMM with n hidden states and m emissions, let $U = \{U_1, U_2, \cdots, U_n\}$ be the set of hidden states, and $V = \{V_1, V_2, \cdots, V_m\}$ the set of emissions. At each discrete time step t, the model resides in one of the hidden states, indicated by q_t, with $q_t \in U$ for all t. The state q_t is often referred to as the model state at time t. The values of q_t cannot be directly observed, and as such represent the 'hidden' part of the HMM.

At every time step, the model changes state by randomly selecting a new state based on a set of probabilities called the transition probabilities (the new state might be the same as the current state). The transition probabilities themselves depend solely on the current state of the model; that is, the transition probabilities for select-

ing the model state q_{t+1} depend solely on the model state q_t. This is referred to as the Markov property, and is what is represented by the 'Markov' term in the name of the HMM. The transition probabilities a_{ij} given a state U_i at time t, and U_j at time $t+1$ are defined as follows.

Definition 3.1 (Transition probability). The transition probability a_{ij} from state U_i to state U_j, for any t, is given by

$$a_{ij} = P(q_{t+1} = U_j | q_t = U_i),$$

where for each pair i, j we have $a_{ij} \geq 0$ and $\sum_{j=1}^{n} a_{ij} = 1$. □

At each time step t, the model also produces one of the emissions in V. Which emission is produced depends on the emission probabilities, that are once again dependent only on q_t. Unlike the model state, emissions can be directly observed. The emission probabilities are denoted by $b_j(k)$; that is, the probability of producing emission V_k at time t given the model state $q_t = U_j$. Formally, the emission probabilities $b_j(k)$ are defined as follows.

Definition 3.2 (Emission probability). The emission probability $b_j(k)$ represent the probability of observing emission V_k while in state U_j, given as

$$b_j(k) = P(V_k | q_t = U_j)$$

where for each pair j, k we have $b_j(k) \geq 0$ and $\sum_{k=1}^{m} b_j(k) = 1$. □

Note that the transition probabilities a_{ij} and emission probabilities $b_j(k)$ are independent of t; that is, they do not change over time.

HMMs can be fully defined[6] by the set of transition probabilities a_{ij} and emission probabilities $b_j(k)$, as these implicitly define the number of states n and number of emissions m. In practice, HMMs are often represented by an $n \times n$ matrix $A = \{a_{ij}\}$ of transition probabilities, and an $n \times m$ matrix $B = \{b_j(k)\}$ of emission probabilities. Note that the matrices A and B are right stochastic; that is, all rows consist of non-negative numbers that sum to one.

Rabiner [1989] describes HMMs using three fundamental problems.

- Problem 1: given a sequence of emissions, determine the probability that these emissions were produced by the model.
- Problem 2: given a sequence of emissions, determine the most likely sequence of hidden states that best matches these emissions.
- Problem 3: given one or more sequences of emissions, and assuming the number of hidden states is fixed, determine the set of model parameters (i.e., transition and emission probabilities) that optimally describe the provided emissions.

[6] Sometimes the definition of HMMs additionally includes a set of initial state probabilities π where $\pi_i = P(q_1 = U_i)$. However, if identical initial probabilities are assumed, as we do here, these can be omitted.

For the purposes of constructing activity recognition models and detecting activities from a previously unobserved sequence of emissions, the first and third questions are of interest. The first problem is solved by a method called the forward-backward procedure. For the third problem, a number of approaches exist: first, the Baum-Welch algorithm, which estimates model parameters using one or more positive example sequences. A second approach is the MA algorithm, which can use both positive and negative examples, but requires target probabilities to be set. In addition, we discuss chained hidden Markov models as an alternative model structure, in which audio and video events are modeled as co-occurring, coupled events. Coupled hidden Markov models, and the methods for solving problems one and three are detailed in the following sections.

3.2.1 The forward-backward procedure

Determining the likelihood that a given sequence is generated by an HMM can, naively[7], be performed by computing every possible 'path' through the model in terms of hidden state transitions. However, the problem can be solved much more efficiently by dynamic programming, as implemented in the forward-backward procedure. The probability of the model being in a certain state at a given time t, having observed the given sequence up to time t, can be based upon the probabilities for all states at time $t-1$. After all, the current state is only dependent on the previous state in HMMs. Specifically, the probability at time t is given by multiplying the probability of producing the current emission while in the current state by the sum of the products of the probabilities at time $t-1$ and the transition probabilities from the states at $t-1$ to the state at t. This results in an induction starting from the first emission. The same calculation can also be done starting from the last emission, as implied in the name of the algorithm.

Definition 3.3 (Forward step). For an emission sequence O, where O is given as $(O_1, O_2, \cdots, O_l)$, with $O_t \in \{1, 2, \cdots, m\}$ for $t = 1, \ldots, l$, the forward variable $\alpha_t(i)$ is given as

$$\alpha_t(i) = P(O_1, O_2, \cdots, O_t, q_t = U_i).$$

The value of $\alpha_{t+1}(j)$ can then be found through induction:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{n} \alpha_t(i) a_{ij}] b_j(O_{t+1})$$

$$\alpha_1(j) = b_j(O_1).$$

□

[7] That is, while the approach arrives at the correct solution, the time complexity is exponential in the number of items in the input sequence.

The probability that the emission sequence O was produced by the provided model is then given by

$$P(O) = \sum_{i=1}^{n} \alpha_l(i). \tag{3.1}$$

While this solves the first problem, as we will later see, we do not always want to start the induction process from the start of the sequence at $t = 1$. Fortunately, using similar reasoning as above, the induction can easily be reversed, starting from $t = l$, and iterating back to $t = 1$. In this case, the backward variable $\beta_t(i)$ is defined as follows.

Definition 3.4 (Backward step). For an emission sequence O where O is given as $(O_1, O_2, \cdots, O_l)$, with $O_t \in \{1, 2, \cdots, m\}$ for $t = 1, \ldots, l$, the backward variable $\beta_t(i)$ is found as

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \cdots, O_l | q_t = U_i).$$

Induction rules are then given by

$$\beta_t(i) = \sum_{j=1}^{n} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$
$$\beta_l(i) = 1.$$

□

3.2.2 The Baum-Welch algorithm

Unfortunately, the problem of finding model parameters to best fit one or a set of emission sequences cannot be solved analytically with current techniques. In other words, there is no technique to establish the optimal model parameters for any given (set of) emission sequence(s) globally. However, it is possible to optimize model parameters locally, for instance by using gradient descent methods or the methods described below. As the model parameters can only be optimized locally, it is important to have an appropriate initial model before optimization.

Arguably, the best known approach to estimate model parameters is the Baum-Welch (BW) algorithm [Rabiner, 1989]. The Baum-Welch algorithm works by re-estimating hidden state and emission probabilities over a number of iterative steps. Given both model parameters and emission sequence(s), the BW algorithm computes the expected number of transitions from one hidden state to another hidden state, for each possible combination of hidden states. The updated probability of transitioning from hidden state U_i to hidden state U_j is then defined as the expected number of transitions from state U_i to state U_j divided by the total expected number of transitions from state U_i (to any state).

Similarly, the emission probabilities are re-estimated as the expected number of times the model is in hidden state U_i and emission V_k is observed divided by the

expected number of times the model is in state U_i (regardless of the emission observed). This re-estimation process is repeated over a number of iterations. The expected probabilities are estimated using the forward and backward variables obtained from the forward-backward procedure discussed in Section 3.2.1, as detailed below. If the resulting overall likelihoods of two successive models are sufficiently similar, the algorithm is considered to have converged and the iterative procedure is terminated.

Let $\varepsilon_t(i,j)$ be the probability of transitioning from state U_i at time t, to state U_j at time $t+1$, given an emission sequence O. Formally:

Definition 3.5 (Transition estimate). The probability $\varepsilon_t(i,j)$ of transitioning from state U_i to U_j for an emission sequence O is given as

$$\begin{aligned}\varepsilon_t(i,j) &= P(q_t = U_i, q_{t+1} = U_j|O) \\ &= \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{P(O)}.\end{aligned} \tag{3.2}$$

Equation 3.2 follows from Definitions 3.1, 3.2, 3.3 and 3.4, and from Equation 3.1. □

Further, let $\gamma_t(i)$ be the probability of the model residing in state U_i at time t given O. $\gamma_t(i)$ can therefore be written as follows.

Definition 3.6 (State estimate). The probability $\gamma_t(i)$ of the model being in state U_i at time t for an emission sequence O is found as

$$\gamma_t(i) = \sum_{j=1}^{n} \varepsilon_t(i,j).$$

□

By summing $\gamma_t(i)$ over all t, the expected number of times the model transitions from state U_i is determined. Similarly, summing $\varepsilon_t(i,j)$ over all t yields the expected number of times the model transitions from state U_i to state U_j. As discussed above, the transition probabilities can be re-estimated by dividing the expected number of transitions from one state U_i to another state U_j by the total expected transitions from state U_i. Similarly, the re-estimated emission probabilities $\bar{b}_j(k)$ are obtained by the expected number of times the model is in state U_j and emission V_k is observed, divided by the expected number of times the model is in state U_j.

Definition 3.7 (Baum-Welch re-estimation). The Baum-Welch re-estimation formulae for the re-estimated transition probabilities $\bar{a}_{ij}$ and re-estimated emission probabilities $\bar{b}_j(k)$ are given by

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{l-1} \varepsilon_t(i,j)}{\sum_{t=1}^{l-1} \gamma_t(i)}$$
$$\bar{b}_j(k) = \frac{\sum_{t=1}^{l} (\gamma_t(j) | O_t = V_k)}{\sum_{t=1}^{l} \gamma_t(j)}.$$

□

3.2.3 The MA algorithm

The MA algorithm[8] is an alternative parameter estimation method with a similar function to the BW algorithm [Mamitsuka, 1997]. However, it differs from the BW algorithm in two important ways. First, rather than completely re-estimating the model parameters at each step, the MA algorithm makes incremental updates to the model parameters. As a result, the parameter changes are smoother in the MA algorithm. This can mean, however, that more iterations are required to converge compared to the BW algorithm, as the MA algorithm typically makes smaller changes to the model parameters each iteration. The second, and perhaps most important, difference is that the MA algorithm uses emission sequences which are not associated with the model for which parameters are estimated as a negative example. In contrast, the BW algorithm only uses positive examples (associated with the model) of emission sequences to determine model parameters.

Whereas the BW algorithm attempts to select parameters maximizing the sum of the likelihoods of the training sequences associated with the model, the MA algorithm utilizes a kind of distance measure for each sequence between the likelihood of the sequence 'belonging' to the current model and a target likelihood set for that sequence. The algorithm tries to minimize the total distance of all training sequences. For activity recognition, targets are set to relatively high likelihood scores for training sequences matching the activity represented by the model, and relatively low (close to zero) likelihood scores for sequences representing different activities.

As discussed above, the MA algorithm makes use of smooth, incremental updates rather than using a re-estimation formula. Each iteration, $\bar{a}_{ij}$ and $\bar{b}_j(k)$ are updated according to the following rules, derived from Baldi and Chauvin [1994].

[8] It is not entirely clear where the acronym 'MA' comes from. While some theories can be formulated, a discussion of those is considered somewhat out of scope.

Definition 3.8 (MA re-estimation). The MA re-estimation formulas for the re-estimated transition probabilities $\bar{a}_{ij}$ and re-estimated emission probabilities $\bar{b}_j(k)$ are given by

$$\bar{a}_{ij} = \frac{e^{\lambda w_{ij}}}{\sum_{h=1}^{n} e^{\lambda w_{ih}}}$$
$$\bar{b}_j(k) = \frac{e^{\lambda v_j(k)}}{\sum_{h=1}^{m} e^{\lambda v_i(h)}}$$

where λ represents a (constant) learning rate value, w_{ij} is a $n \times n$ matrix of values representing the hidden state transition probabilities, and $v_j(k)$ is a $n \times m$ matrix of values representing the emission probabilities. □

In the MA algorithm, w_{ij} and $v_j(k)$ are updated, instead of updating a_{ij} and $b_j(k)$ directly, as shown below. As the MA algorithm makes use of emission sequences belonging to different classes for parameter re-estimation, let O^x be the x-th sequence in a set of emission sequences. Further, P_x is defined as the probability of emission sequence O^x being produced by the currently estimated model, that is, $P_x = P(O^x)$. For each emission sequence, a target probability P_x^* is defined. The MA algorithm then tries to minimize a distance metric with respect to P_x and P_x^* as follows.

Definition 3.9 (MA distance metrics). Let d_x and $d_{\max}$ be distance metrics with regard to P_x and P_x^*, given as

$$d_x = \log(\frac{P_x^*}{P_x})$$
$$d_{\max} = \log(\frac{p_{\max}^*}{P_{\min}^*})$$

where $P_{\min}^*$ is the minimum value of all P_x^*, and $P_{\max}^*$ the maximum value of all P_x^*. □

In practice, $d_{\max}$ can be set to any number provided $\forall x(d_{\max} > |d_x|)$. Each iteration, the parameters w_{ij} and $v_j(k)$ are updated according to $w_{ij} = w_{ij} + \Delta w_{ij}$ and $v_j(k) = v_j(k) + \Delta v_j(k)$. Δw_{ij} and $\Delta v_j(k)$ are given as follows.

Definition 3.10 (MA parameter smoothing). Δw_{ij} and $\Delta v_j(k)$ are given as

$$\Delta w_{ij} = C_a \sum_x \frac{d_x}{d_{\max}^2 - d_x^2} \sum_{t=1}^{l_x} (\varepsilon_t(i,j) - a_{ij}\gamma_t(i))$$
$$\Delta v_j(k) = C_b \sum_x \frac{d_x}{d_{\max}^2 - d_x^2} \sum_{t=1}^{l_x} ((\gamma_t(j)|O_t^x = k) - b_j(k)\gamma_t(j))$$

where l_x represents the number of emissions in O^x, and C_a and C_b are given constants. □

For most practical applications, the constants C_a and C_b are set to 1. In the above equations, the first component ($\frac{d_x}{d_{\max}^2 - d_x^2}$) indicates whether to increase or decrease the likelihood of the model producing the given sequence O^x, depending on the current and target probabilities P_x and P_x^*. The second components in both equations represent the difference between the expected parameter probabilities and the current model parameter probabilities.

3.2.4 Chained hidden Markov models

Chained hidden Markov models (c-HMM), described by Brand, Oliver, and Pentland [1997], consist in essence of a number of HMMs in which the hidden states of the separate models are linked together by introducing additional transition probabilities between the hidden states of each model to the hidden states of the other models. The amount of HMMs making up a c-HMM is called the number of chains. For example, a two-chain c-HMM consists of two HMMs, for which there are defined transition probabilities between the hidden states of both models. As a result, at any given time, the model is in two hidden states, and two emission symbols are observed. Note that the individual models are not required to have the same number of hidden states. A c-HMM can be fully expressed as the Cartesian product of the HMMs making up the chains.

As all c-HMMs used in this chapter are two-chain models, they will be the sole focus of this section. However, the same principles apply to models of any number of chains. As there is both audio and video data available for activity recognition, a two-chain c-HMM allows modeling the audio process and the video process separately, rather than as part of a single model. In other words, audio and video can be modeled as two independent, but interacting, processes. This is a considerable difference compared to the standard HMMs, which are unable to distinguish between the source of the events (that is, there is no distinction depending on whether an event is an audio or video event). Theoretically, a c-HMM offers advantages in capturing causal couplings; for example, visiting the sink might increase the chance of observing the sound of running water. A downside is increased model (and particularly state-space) complexity.

In regular HMMs, the current hidden state is solely dependent on the previous hidden state, as stated by the Markov property. However, c-HMMs break the Markov property in this respect, as the current hidden state in a chain of a c-HMM is dependent on both the previous state in that chain, as well as the previous states of every other chain. Note that the current hidden state of a chain is not affected by the current states of any other chains. This requires a number of adaptations to the standard HMM algorithms for parameter estimation and the forward-backward algorithm.

As there are now two or more concurrent hidden states to keep track of, the standard approach of dynamic programming to solve the forward-backward procedure will no longer work. Adaptations have been devised, generally at the loss of accu-

racy to reduce computational complexity to manageable levels [Brand, 1997]. One possibility is to first transform the c-HMM into a regular, 'joint' HMM, and then use the standard forward-backward algorithm. This transformation is accomplished by creating an HMM with the Cartesian product of the hidden states from each chain, where each hidden state of the joint HMM represents a combination of hidden states from the chains. The state transition probabilities are recomputed from the state transitions in the c-HMM. The result is a computation with a computational complexity exponential in the number of chains. However, as only c-HMMs of two chains are used in this work, the computational costs remain manageable. Let x and y represent individual chains, and let c represent the Cartesian HMM. The hidden state transition probabilities for the Cartesian HMM can then be defined as follows.

Definition 3.11 (c-HMM transition probability). The transition probabilities for a c-HMM consisting of two chains x and y, and a corresponding Cartesian HMM c, can be found as

$$\begin{aligned} a_{c_{ip}c_{jr}} &= P(q_{t+1} = U^c_{j,r} | q_t = U^c_{i,p}) \\ &= P(q^x_{t+1} = U^x_j | q^x_t = U^x_i, q^y_t = U^y_p) * P(q^y_{t+1} = U^y_r | q^x_t = U^x_i, q^y_t = U^y_p). \end{aligned}$$

□

As described above, transition probabilities depend on all current states of all chains. If independence between x and y is assumed, the components of the equation above can be written as:

$$P(q^x_{t+1} = U^x_j | q^x_t = U^x_i, q^y_t = U^y_p) = P(q^x_{t+1} = U^x_j | q^x_t = U^x_i) * P(q^x_{t+1} = U^x_j | q^y_t = U^y_p)$$

$$P(q^y_{t+1} = U^y_r | q^x_t = U^x_i, q^y_t = U^y_p) = P(q^y_{t+1} = U^y_r | q^x_t = U^x_i) * P(q^y_{t+1} = U^y_r | q^y_t = U^y_p).$$

In practice, the assumption of independence may not hold. However, the above equations still provide an approximation if this is the case. Inserting these equations gives for the Cartesian hidden state transition probabilities

$$a_{c_{ip}c_{jr}} = a_{x_i x_j} a_{y_p y_r} a_{x_i y_r} a_{y_p x_j}.$$

Similar reasoning can be applied to the Cartesian emission probabilities, giving

$$b_{c_{ip}}(k) = b_{x_i}(k) b_{y_p}(k).$$

The parameter estimation algorithms, such as BW, can then be used mostly unaltered. The only issue arises when the parameters for the c-HMM need to be re-estimated. Whilst it is straightforward to determine the joint HMM's parameters from the c-HMM's parameters, the opposite is not true, introducing some amount of error in the re-estimation. The algorithm used here is based on the method described by Brand, Oliver, and Pentland [1997], by factoring after re-estimation of the joint HMM. Theoretically, convergence to a local optimum is not guaranteed. However,

in practice this has not presented any problems. As has been noted by Brand, Oliver, and Pentland [1997], convergence can be guaranteed by applying a gradient descent algorithm after re-estimation.

3.2.5 Hidden Markov models in activity classification and segmentation

For activity classification, one option is the model each activity as a separate state in the HMM. However, for more elaborate activities, a single state may not provide sufficient complexity to model the activity in an appropriate fashion. As such, the approach chosen for this chapter is to model the activities as a separate HMM each. As there are six classes of activities to be recognized, six separate models need to be constructed, each representing a single activity class. The inputs for these models are sequences consisting of audio and video events, generated from either manually annotated or automatically annotated data, as will be further detailed in Section 3.3 below. For a given sequence of observed events, the probability of a model producing the given sequence is determined for all six models using the forward-backward procedure. The model which generates the highest probability assigns its activity class to the event sequence.

When we apply this scheme to a continuous sequence of measurements, we will note transitions from one activity to another whenever the probability of the previous top model is surpassed by one of the other models. Essentially, such transitions mark the boundaries of ADL segments within the measurement sequence, removing the need for an explicit segmentation algorithm such as the k-segmentation algorithms described in Chapter 2; the HMMs perform implicit segmentation instead. This does not mean that in general the segmentation problem can simply be avoided by the choice of algorithm however; obtaining an implicit segmentation in this fashion depends to a large degree on the classification performance of the models in question. If the algorithm used is ill-suited to the task, the segmentation provided is unlikely to produce good results as well.

Before any of the models can be used for classification, parameter estimation (or training) is required. Training is performed by using one of the parameter re-estimation methods described above (i.e., BW or MA). The input for training takes the form of a set of training sequences, and an initial set of model parameters (see Section 3.3.5). All six models must be trained independently, using different sets of training sequences (in the case of BW) or different sets of likelihood targets (in the case of MA).

3.3 Activity recognition data collection

To both test and train an activity recognition system, a set of data labeled with ground truth values must be created. To obtain this data, a study was performed in the kitchen area of Philips' ExperienceLab in Eindhoven, a facility specifically designed for per-

forming experiments and creating recordings in a home environment. The kitchen area consists of a fully functional and furbished kitchen and contains, among others, a counter, sink, stove, oven, refrigerator, and a small table with chairs. The environment also contains multiple unobtrusively mounted cameras and microphones intended for recording purposes. As mentioned, we only use a single camera and microphone - in this case, a camera mounted on the ceiling in one of the corners of the room (resolution of 576×352, 25 frames per second), and a microphone mounted centrally on the ceiling.

For the study, eleven participants were recruited to perform six different activities in the kitchen environment. These consisted of: storing a set of groceries from a bag they were given at the start of the experiment, preparing a meal, eating the meal, doing the dishes, vacuuming the kitchen floor and preparing a drink. The participants of the study were researchers and students of a research campus, with ages in the range of 20 to 65 (exact ages were not recorded as part of the study). Participants were asked in advance which meal they would like to prepare and eat, so ingredients could be provided. The participants were free in their choice of meal, but it was suggested to select something they were comfortable with preparing, and was not too time-consuming to prepare, such as a simple pasta dish or soup. In addition, participants were free to either do the dishes by hand or use the dishwasher in the kitchen for the 'doing the dishes' activity.

At the start of the protocol, after explaining the aim of the study, the participant was given a tour of the kitchen to familiarize themselves with it. This consisted of instructions on how to operate the sink, dishwasher, and stove (none of the participants made use of the oven). In addition, they were shown the locations of the refrigerator, vacuum cleaner, dinner table, and cooking and kitchen utensils such as knifes, plates, glasses, and condiments already present in the kitchen. After the tour, participants were given the opportunity to ask any further questions regarding the kitchen environment.

Afterwards, participants were asked to accompany the researcher to an adjacent room, closing the door to the kitchen. Here, they were given a list of the activities to perform for reference during the study which they were allowed to take with them into the kitchen. At this point, they were instructed that they were allowed to perform the activities in any order (i.e., not necessarily in the listed order), in whatever manner they wished. The only exception was the request to eat at the specified table, so the participants would remain within the camera's view. In addition, they were instructed to exit the kitchen upon completing all six activities (into the same room), and to close the door after entering or leaving the kitchen. Next, they were handed a bag of groceries containing the ingredients required for preparing their meal. When the participant indicated that everything was clear and they had no further questions, the recording was started, and participants were asked to enter the kitchen and begin the protocol when they were ready.

Activities	*subactivities*	*Audio events*	*Video events*
Storing groceries	Waiting	Movement	Fridge
Preparing dinner	Boiling water	Groceries bag	Dishwasher
Eating	Rinsing dishes	Kitchen door	Sink
Doing dishes	Washing dishes	Groceries	Cupboard
Vacuuming	Cupboard interaction	Fridge	Stove
Preparing a drink	Fridge interaction	Fridge door	Dining table
Other	Dishwasher interaction	Pans	Transition
	Pouring water	Cutlery	Other
	Plating up food	Tap	
	Disposing of garbage	Stove on / off	
	Washing hands	Stove	
	Vacuuming	Plates	
	Stirring	Chair	
	Other	Voice	
		Glass	
		Hot water	
		Vacuum cleaner	
		Cleaning the counter	
		Dishwasher	
		Pouring a drink	
		Background	

Table 3.1: The list of activities and events possible for each annotation track. The 'activities' and 'subactivities' tracks list participant activities, 'audio events' are related to observed sounds, and 'video events' are related to the position of the participant in the kitchen.

During the recordings, the participants were alone in the kitchen. They were however observed through the camera system by a researcher in an adjacent room. During the activities, participants were allowed to leave the kitchen if needed (for example for clarifications), although this did not occur during any of the recordings. The recording session was stopped once all activities were completed and the participant had left the kitchen.

Counted from the start of the first activity to the end of the last activity (in the order they were performed), the participants completed the six activities within a time span ranging from 16 to 42 minutes (average: 26.3 ± 9.1 SD). A lot of the variation in completion time can be related to the time required to prepare the selected meal. Based on technical considerations and time constraints, a total of eight sessions were selected for annotation, as described below. One session was intended as a pilot and featured a different camera setup, and two sessions were not annotated due to time constraints (these were the two last recorded sessions).

3.3.1 Data annotation

To establish a ground truth for the activities and events, manual annotation of the audio and video data was used. Here, the ground truth refers to information available from direct observation, rather than some entire objective measure. As is well-known, annotation, whether done by humans or machines, is generally imperfect, and as such not a completely accurate representation of reality. Within the context of the study, however, it is arguably the most accurate measure of the performed activities and observed events available, and as such will be used as a comparison to the inferred results from the machine learning algorithms.

The annotations themselves take the form of a number of triplets containing the activity or event name, the starting time, and the end time. Here, the time is measured relative to the start of the recording. The data was annotated using a proprietary video annotation tool similar to, for example, Noldus observer XT, allowing for annotation while simultaneously viewing the recorded material. The annotation tool allows for the definition of time intervals in which a certain activity or event takes place. The annotator is able to move through the video and audio data, and at any point in time select a new activity or event. This time point would serve as the start time of the new entry, and the end time of any entry previously selected.

Within the tool, data was annotated on four separate tracks: an activity track containing the ground truth activity value, a subactivity track containing small tasks performed towards completing the current activity, an audio track listing observed audio events, and a video track listing observed video events. For each track, a set of possible values was determined beforehand. A full list of these values is shown in Table 3.1. In total, 7 activity values, 14 subactivity values, 21 audio values and 8 video values were defined. Note that while most tracks have an ‘other’ value, the audio events track does not. For this track, the value ‘background’ also fulfills the ‘other’ role. An example of the four annotated tracks within the annotation tool is shown in Figure 3.1.

It can be argued that the choices made at selection of the activities and events will at least in part affect the outcomes and results. In an attempt to counteract some of this bias, the selected activities were chosen as activities that would commonly be performed in a kitchen environment as part of everyday life, without taking into account any a priori concern with regard to the expected recognition accuracy. Similarly, audio and video events were chosen as general as possible (although related to kitchen activity), in collaboration with experts in the fields of audio scene analysis and video scene analysis, respectively.

All eight selected sessions were annotated by hand. The activity and subactivity track were annotated by the researcher conducting the experiment, while the audio and video tracks were annotated by experts in audio and video scene analysis, respectively. The resulting annotated recordings were used as training and test input for the

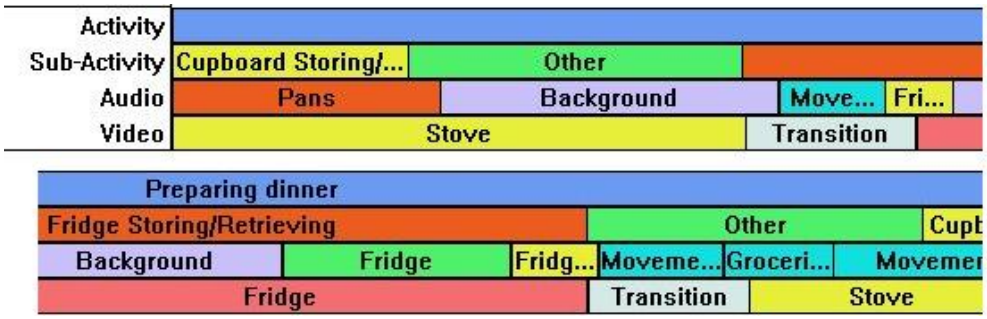


Figure 3.1: Example section of an annotated activity, cut over two lines.

activity recognition algorithms described in later sections. Here, the audio and video event annotations are used as model emissions, and the activity annotations serve as a ground truth. The annotated audio and video data can also be used as input for the audio and video classification algorithms discussed below. Due to the considerable amount of time required for manual annotation, each track was rated by only a single annotator - as such, it is unfortunately not possible to establish inter-rater reliability of the annotations.

3.3.2 Audio and video classification

An alternative to using manually annotated event data to train the activity recognition algorithms is to create audio and video classification algorithms, which classify the raw audio and video input into classes similar to the annotated audio and video classes. This has the advantage that once these algorithms are trained, the collection of new data sets is far less time consuming, as only the ground truth (i.e., activity) needs to be annotated by hand. The downside is that the annotations created by these algorithms are likely to be less accurate than annotations created manually.

As the field of audio and video classification is a topic on its own and as these algorithms are not the focus of this chapter, only a brief description will be provided here. The audio classification algorithm used is based on Gaussian mixture models, and works by extracting a set of features from the raw audio signal. The algorithm uses part of the manually annotated audio data both for training and testing. The video classification algorithm is based on a background subtraction model, using manually defined regions of interest in the kitchen. The algorithm uses part of the manually annotated data for testing purposes. The outputs of both algorithms correspond to categories defined in the audio and video tracks of the annotated data.

Cross validation results on the annotated data have shown that the performance of the audio scene analysis algorithm is within the region of $60-70\%$, depending on the evaluated recorded session. The performance of the video scene analysis algorithm is approximately 71%. Both algorithms perform well above chance, although they are

by no means completely accurate. For the activity recognition algorithms to perform well with the automatically annotated data, it must therefore be tolerant of the errors made in the automatic annotation. The results of the activity recognition algorithms using both the automatically annotated and manually annotated data are discussed in Sections 3.4 and 3.5.

While we will not delve into the details of the scene analysis algorithms, they do suffer from a number of limitations. First, the context in which these algorithms operate can affect their performance; for example, external noises can create false alarms in the audio analysis algorithm, and the video scene analysis algorithm is affected by changes in lighting conditions or by moving around objects such as chairs in the camera's view. Both algorithms have some ability to cope with such changes however: the video scene analysis algorithm uses a background subtraction model to adjust to changing conditions, while the audio scene analysis can be periodically recalibrated.

A further limitation is that the algorithms currently used cannot easily be applied to a different environment (such as a kitchen with a different layout). In many cases, automatic recalibration may suffice for the audio scene analysis; in the worst case however, retraining may be required, which is not a straightforward process. The video scene analysis algorithm requires the user to indicate the location of several places of interest. Fortunately, this can be done in a fairly straightforward fashion, but does mean that some setup is required when moving to a new environment.

3.3.3 Sequence generation

As discussed earlier in this chapter, HMMs generally operate on series of discrete valued emissions ordered in time[9]. The annotated data, however, consists of ordered events with a certain start time and duration, at the end of which a new event starts - similar to the sequences described in Chapter 2. As a result, the continuous data sequences S must be converted into discrete sequences of emissions O.

Figure 3.1 shows an example of part of an annotated data sequence. As can be seen in the figure, there are separate tracks for audio and video events. For c-HMMs, which require both an audio and a video emission for each input, the existence of the two separate tracks matches well with the input requirements of the model. For the non-chained HMM variants however, the audio and video events must be joined into a single sequence of emissions to provide a suitable input sequence. Below, we will describe the methods used to construct a sequence of emissions O from an audio and video track, for both the c-HMM and the regular HMM variants.

To obtain discrete sequences for c-HMM input, the following method was used. Let S^a and S^v be the time-ordered audio and video sequences, respectively, and let O

[9] Although they can be extended to the case of continuously valued emissions by replacing the emission probability matrix with a set of emission probability density functions.

Sequence: pans, stove, background, stove, transition, movement, fridge door, fridge, background, fridge, fridge, ...

Figure 3.2: Example transformation from an annotation sequence to an HMM emission sequence. The colored ellipses indicate when a new event is added to the HMM input sequence. Red ellipses indicate audio events, while purple ellipses indicate video events. The resulting emission sequence is shown below the annotation sequence.

be the (initially empty) sequence of emissions. A sliding window moves simultaneously from the beginning of S^a and S^v to the end of the sequences. Whenever a new event is encountered in either S^a or S^v, a new emission is added to O, consisting of the newly encountered event, and event value of the other stream at that time. For example, if a new audio event is encountered with a timestamp t, an emission is added to O consisting of the new audio event value, and the value of S^v at time t.

Whenever an event remains unchanged for a certain predefined duration d in S^a or S^v, it is added as a new emission to O again (with the event of the other stream) each time after d expires, counted from the time it was last added. This latter rule ensures that events continuing for a longer time are adequately represented in the emission sequence. Equivalently, this can be seen as splitting each event with duration $w_i > d$ in S^a and S^v into $\lfloor \frac{w_i}{d} \rfloor$ subevent of duration d, and a single subevent of duration $\mod(w_i, d)$ (if $\mod(w_i, d) > 0$), and adding an emission to O whenever a new event is encountered in S^a or S^v. For all results discussed below, d was set to one second.

To create input sequences for the regular HMM variants, the method described above was adjusted slightly. Once again, let S^a and S^v be the time-ordered audio and video sequences, let O be the initially empty sequence of emissions, and let a sliding window move simultaneously from the beginning of S^a and S^v to the end of the sequences. Whenever a new event is encountered in either S^a or S^v, a new emission is added to O of the same value - in this case, no events are included from the other stream. In this case, emissions in O consist of a single event value, which might represent either an audio or video event. This is in contrast to the c-HMM method, where each emission in O consists of a pair containing a single audio event value, and a single video event value. As before, whenever an event remains unchanged for a duration d in S^a or S^v, it is added as a new emission to O again each time after d expires, counted from the time it was last added, with d was set to one second.

For the above method, it is impossible to make exact statements regarding the amount of time represented by a given emission sequence. At segments in a sequence S where rapid chances in events are observed, for instance, a relatively large amount of emissions would be generated for the given amount of time. In other words, the number of emissions in the generated sequence represents the information density of the input sequences rather than the linear passage of time. However, the sequence creation method does allow for the definition of an upper bound on the amount of time represented by a given sequence; given an emission sequence O containing L observations, the original event sequences S^a and S^v can have at most a total duration of $\frac{d \cdot L}{2}$. Here, it is assumed that S^a and S^v have the same total duration, as they have been recorded together. Additionally, we can also assume that the combined number of items in S^a and S^v is at most L.

3.3.4 Imbalanced data

A data set is considered imbalanced when one or more classes are much more common or well represented compared to one or more other classes. Although the number of sequences obtained from the data used here is approximately equal for most classes, the average sequence length per activity class can vary greatly depending on the amount of time it requires to perform the activity in real life. In some cases, for instance when dividing sequences into a number of subsequences of a certain length, variations in sequence length between classes can cause the data set to become imbalanced. The occurrence of this is related to the selection of classes for the models (in this case, the selected activities). However, as mentioned, the initial selection of classes was not based on considerations of maintaining a balanced data set.

Working with imbalanced data sets introduces a number of problems. The first occurs when computing accuracy scores. Recognition accuracy on an over-represented class will likely dominate the overall accuracy score. For example, highly accurate recognition of a class that represents 75% of the data will likely result in an overall accuracy of over 75%, even if recognition accuracy on the remaining classes is poor. Similarly, high recognition accuracy on an under-represented class will have relatively little effect on the overall accuracy. Rather than using accuracy as a measure, it can be preferable to report *class accuracy* instead; the average of the recognition accuracies over the individual classes [van Kasteren et al., 2008].

To illustrate, the accuracy of a classification problem, sometimes called timeslice accuracy when applied to a set of windows of time-ordered data, is defined for a set of n classified instances as

$$\frac{1}{n}\sum_{i=1}^{n}(y_i = C_i),$$

where y_i represents the estimated or predicted class, and C_i represents the actual class of instance i. Here, the value of $(y_i = C_i)$ is 1 when the estimated class y_i is equal

to the actual class C_i, and 0 otherwise. Similarly, the class accuracy for l classes $C_i \in \{c_1, \ldots, c_l\}$ is defined as

$$\frac{1}{l} \sum_{j=1}^{l} \frac{\sum_{i=1}^{n_j} (y_i = C_i)_j}{n_j},$$

where n_j represents the number of instances i where $C_i = c_j$, and the value of $(y_i = C_i)_j$ is 1 when y_i is equal to C_i, and 0 otherwise, for instances i where $C_i = c_j$.

A second issue occurs when training the classification models. The presence of an over-represented class can cause bias in the resulting models. As the BW algorithm averages over all input sequences, this method remains largely unaffected. The MA algorithm, by contrast, can be affected, as each sequence is assigned a target score to be optimized. This can cause the models not representing the dominant class to prioritize differentiating from the dominant class over accurately representing its own class, resulting in poor distinction between non-dominant classes. A solution is to forcibly rebalance the data set by undersampling; that is, randomly removing instances of the dominant classes until all classes are balanced. Unfortunately, this can in some cases cause valuable training data to be omitted, but generally this approach results in improved overall performance. Other approaches also exist, such as oversampling, or Monte Carlo simulation.

3.3.5 Model initialization

Regardless of the model's architecture or parameter optimization method used, an appropriate initial model is required for good parameter optimization results. Both the BW and the MA algorithms require a set of initial parameters to create a (locally) optimized model. These initial parameters consist of the number of hidden states, as well as the initial transition and emission probabilities. If the initial parameters are chosen such that they resemble the underlying process that we attempt to model, optimization techniques are more likely to produce a further optimized model that matches the modeled set of emissions well.

One well-known approach is to collaborate with an available domain expert to create the initial model. This approach can sometimes work well, but is rather time consuming, and therefore more suitable when only a few models need to be constructed. The fact that this method is rather subjective can also be seen as a disadvantage. Another method is to set transition probabilities uniformly, or use some random distribution to determine their values. Unfortunately, this yielded poor results for this application, and as such, other methods needed to be devised.

For this work, we developed an approach which uses part of the annotated data to determine the initial parameters. When constructing an initial model for a certain activity, the number of hidden states is determined by a function of the number of annotated subactivities common to that activity. Hidden state transitions are then set based on the sequential patterns between the subactivities. Finally, emission prob-

abilities are determined by calculating the probabilities of the events occurring at a given time during the activity and subactivity.

As an example, consider the activity of preparing a drink[10]. This activity commonly consists of two subactivities, namely retrieving an object from the fridge and retrieving an object (glass) from the cupboard. This would translate into three hidden states, one for each subactivity, and one final hidden state representing other subactivities such as pouring the drink. When one prepares a drink, one might start with retrieving something to drink from the fridge, and then retrieve a glass, or the other way around. However, both subactivities must be completed before a drink can be poured. This would translate into similar values for the transition probabilities for the first two hidden states to all other hidden states, but low transition probabilities from the final hidden state to one of the previous hidden states.

The emission probabilities are computed by recording the total amount of time each event was observed during the activity of preparing a drink and the relevant subactivity. Using this data, the probability of observing a particular event during each set of activity and subactivity is computed. An example of such a distribution for preparing a drink / retrieving an object from the cupboard is shown in Figure 3.3. As each event is linked to an emission, the emission probabilities are set to the computed probabilities.

The above method obtains appropriate initial models for the standard HMM algorithms. However, these initial models did not translate well for use by c-HMM methods. In addition, the above method still requires a certain amount of manual input, such as interpreting the appropriate number of hidden states and adjusting the hidden state transition probabilities. For these reasons, we developed a second approach based on evolutionary algorithms. Evolutionary algorithms are very similar to the genetic algorithms described in Chapter 4 - as such, also see Section 4.3.1.

Initialization through evolutionary algorithms. In an evolutionary algorithm, a collection (or 'population') of candidates is maintained throughout a series of 'rounds'. In this case, the candidates are represented by potential initial models for the HMM training algorithms, each consisting of a set of transition and emission probabilities. Initially, these models are randomly generated matrices, adhering to the requirements discussed in Section 3.2. The aim of the evolutionary algorithm is to evaluate the candidate models, select (with some chance) the best ones, and modify them in some way to obtain new candidates. These steps are called fitness, selection, and reproduction, respectively, and together constitute one round. The process is then repeated for a set number of rounds.

For the candidate initial models, determining the fitness is straightforward, since this problem corresponds to question one in Section 3.2; given a sequence of emis-

[10]Or more specifically, preparing a cold drink. Preparing a hot beverage may involve other subactivities.

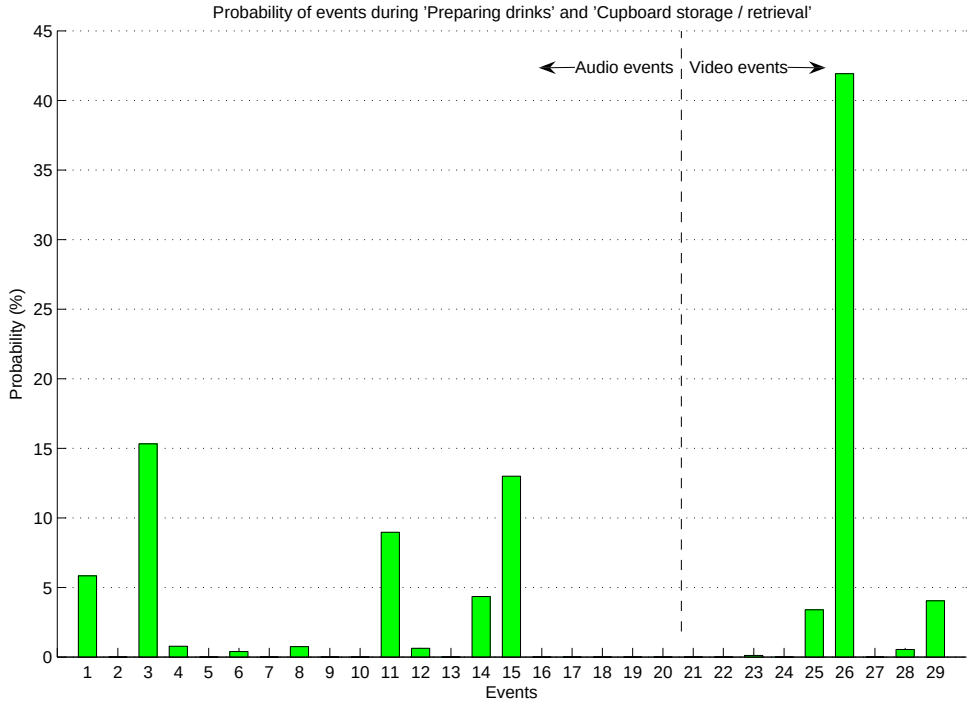


Figure 3.3: Event distribution during the activity 'preparing a drink' and the subactivity 'retrieving / storing an object from / in a cupboard'. Probable audio events include: 'background' (1), 'kitchen door' (3), 'stove' (11), 'movement' (14) and 'glass' (15). Probable video events include: 'sink' (25), 'Cupboard' (26) and 'transition' (29).

sions, determine the probability that these emissions were produced by the model. As a result, fitness can be determined using the forward-backward procedure. Here, a higher likelihood corresponds to a higher fitness score. The fitness score is used as input for the selection procedure.

The selection scheme used here is named tournament selection. In tournament selection, two candidates are selected at random, of which the candidate with the highest fitness is retained, and the other candidate is discarded. This process is repeated by randomly selecting two more of the candidates that have not been selected previously. Once there are no candidates left for selection, the selection process starts anew for those candidates that have been retained, until the desired number of discarded candidates has been reached. Tournament selection is intended to strike a balance between retaining candidates with high fitness, while also retaining some with lower scores, to avoid becoming trapped in a locally optimal solution.

Finally, as part of reproduction, two mechanisms are used to create new candidates, named mutation and cross-over. First, one of the surviving candidates is selected at random as a basis for a new candidate. As part of the mutation mechanism, each transition and emission probability is replaced by a new random probability, with a certain percentage chance. As a result, some of the transition and emission probabilities will be entirely new, while others will be unaltered. For cross-over, a second random surviving candidate is selected. Here, the transition or emission matrices from both candidates are converted to vectors (in a by-column fashion), and a random entry is selected. Then, either all entries after or before are replaced by the same entries from the second candidate. Cross-over for the transition and emission probabilities are random, and so may occur for either, both, or none. After reproduction, the probability matrices rescaled and rebalanced as needed.

Experimental results (see Section 3.5.1) show that this method generates initial models comparable to the previous method described for non-coupled HMMs. In addition, this method also generates appropriate models for use by c-HMM methods. The downside of the evolutionary method is the increased computation time required compared to the statistical method, especially since the method needs to be performed a number of times for different numbers of hidden states to determine the optimal number of hidden states to use. An alternative would be to use the first method to determine the number of hidden states, and the second method to determine the initial parameter values. In Sections 3.4 and 3.5, the evolutionary method for initialization is used unless stated otherwise.

3.4 Sequence-based activity classification

In this section, the classification power of the activity recognition models described in Section 3.2 is examined using sets of emission sequences extracted from the manually annotated and automatically annotated data, respectively. Each emission sequence belongs to a single activity class, and consists of a time-ordered sequence of emissions obtained from a recording for the full duration of the activity performed. In this case, we examine the performance of the activity recognition models under the assumption that we know the actual segmentation of the individual activities. The performance of the models under the condition that the segmentation is unknown a priori (that is, our sequence represents a continuous stream of recorded activities) is discussed in Section 3.5.

As discussed in Section 3.2, an HMM is computed for each of the modeled activities. The observation sequence to be classified is evaluated for each of the models, resulting in a likelihood score for each model. The model with the highest likelihood score matches the observation sequence best, and the emission sequence is classified as belonging to the modeled activity. Comparing this classification against the annotated ground truth for the emission sequence then yields an accurate classification if the classification and ground truth match and an error otherwise.

The accuracy of a method is then defined as the number of accurate classifications divided by the total number of classifications made (both accurate classifications and errors). As some activities occur more frequently than others, there exists the potential of an imbalanced data set, as described in Section 3.3.4. To counteract this, class accuracies are also computed; these are defined as the mean of the classification accuracies of the individual activities. The results listed in this section have been obtained using cross validation to ensure a separate test and training set of data.

In this section (as well as in Section 3.5), we consider both the accuracy on the manually annotated data set and the automatically annotated data set. As discussed in Section 3.3.1, the manually annotated data set consists of annotations for the activities, subactivities, audio events, and video events assigned by experts using the annotation tool. In contrast, the automatically annotated data set uses scene analysis techniques to automatically derive audio and video events. The ground truth activity labels serving as ground truth are in both data sets manually annotated, and identical. In addition to the accuracies for manual and automatic annotation, we also examine the use of a single modality (i.e., audio only or video only) compared to both modalities, and the resulting effects on the recognition accuracy.

Sequence length. An important consideration for the activity classification methods used is the length of the emission sequences which are to be classified. Theoretically, using the entire sequence for classification will yield the best results, as all of the available information upon which classification can be based is taken into account. However, as a result, classification cannot be performed until after the activity has finished. For many real life applications, activities need to be identified while the activity is taking place. Classifying on shorter sequences means less delay between an activity commencing and the detection of that activity, however, it also requires classification with less than complete information, possibly resulting in increased classification error. Therefore, a compromise between classification accuracy and delay between the start of an activity and its recognition must be found. The effects of sequence lengths are discussed in more detail in Section 3.5.

3.4.1 Classification on a set of extracted activities

To examine the results of sequence-based activity classification in the most optimal situation, classification has been applied using the full-length of each of the sequences. The resulting accuracy scores for the manually annotated data are shown in Table 3.2a. When considering the class accuracy scores, which are robust against class imbalance, it can be seen that all methods perform well on classification of full-length sequences. This is especially true for the non-coupled classification methods, and for the MA method in particular, with a classification class accuracy of over 96%. It can also be seen that there is little deviation between overall accuracy and class accuracy, suggesting that the trained classification models are not heavily affected by any class imbalance.

	BW	*MA*	*cBW*	*cMA*
Storing groceries	1	1	1	0.625
Preparing dinner	0.9375	0.9375	0.9375	0.875
Eating	0.8889	1	0.8889	1
Doing dishes	1	1	0.9091	0.9091
Vacuuming	1	1	1	1
Preparing a drink	0.7857	0.8571	0.7143	0.7857
Class accuracy	0.9354	0.9658	0.9083	0.8658
Overall accuracy	0.9242	0.9545	0.8939	0.8636

(a) Manually annotated data

	BW	*MA*	*cBW*	*cMA*
Storing groceries	1	1	0.875	1
Preparing dinner	1	1	1	0.9375
Eating	0.6667	0.8889	0.8889	0.8889
Doing dishes	1	1	1	1
Vacuuming	1	1	1	1
Preparing a drink	0.4286	0.2143	0.2143	0.7857
Class accuracy	0.8492	0.8505	0.8297	0.9354
Overall accuracy	0.8333	0.8182	0.803	0.9242

(b) Automatically annotated data

Table 3.2: Sequence-based activity classification accuracy rates on manually annotated data and automatically annotated data using full-length sequences. The tables list the accuracies using the BW algorithm, the MA algorithm, the coupled BW algorithm (cBW) and the coupled MA algorithm (cMA).

A somewhat different picture can be seen when considering the results of full-length sequence-based activity classification on the automatically annotated data, shown in Table 3.2b. For most classification methods, class accuracy scores are lower compared to the annotated data. This is not surprising, as we expect the automatic classification of observations to be less accurate compared to the manual annotation; the use of scene analysis techniques introduces an additional classification step with an expected error of around 30%, as discussed in Section 3.3.2, likely resulting in a more noisy classification set compared to the manual annotation.

Remarkably, the coupled MA method breaks the trend of poor performance compared to the manually annotated data, and is considerably more accurate on the auto-

matically annotated data than on the manually annotated data. This is an unexpected result, but may be partially explained by both the fact that the coupled MA model considerably outperforms the other models on the automatically annotated data (particularly for the 'preparing a drink' activity), and that the coupled MA model seems to perform relatively poorly on the 'groceries' activity in the manually annotated data.

Also of note is the relatively poor performance of most models on the 'preparing a drink' activity. It is not entirely clear why this is, but as we will see, poor performance on the 'preparing a drink' activity for the automatically annotated data set is somewhat of a theme. A possible explanation for this is that the scene analysis techniques used struggled with accurately capturing the events needed to distinguish this activity from other activities.

In real life applications, classification on full-length sequences is not always an option, as this means classification does not take place until after the activity has finished - and that is assuming we would even have a means to determine that an activity has finished. To simulate a more realistic situation, the full-length sequences are split into subsequences of length 50 each (with no overlap between subsequences). These subsequences represent short, momentary observations of an activity in progress, with no explicit information of how far the activity has progressed in time.

As some activities intrinsically have longer durations than others (e.g., preparing dinner generally takes longer than preparing a drink), splitting sequences into subsequences as described above creates an imbalance in the data set, as discussed in Section 3.3.4. As mentioned there, the MA algorithms are especially susceptible to class imbalances. Empirical evidence suggests that the coupled MA algorithm suffers most from this, while the 'standard' MA algorithm is less affected. As also described in Section 3.3.4, rebalancing the data set can alleviate this problem for the coupled MA algorithm. The results for the coupled MA algorithm both with and without using a rebalanced data set are shown in the following results.

The sequence classification results on subsequences of 50 emissions obtained from the manually annotated data are shown in Table 3.3a. Compared to the full-length sequences, the results on the subsequences show a lower class accuracy overall. This result is expected, as the shorter subsequences contain less information than the full-length sequences. The coupled MA algorithm performs especially poorly, mostly due to class imbalance, caused mainly by the over-representation of the 'preparing dinner' class. When using a rebalanced data set, the results for the coupled MA algorithm improve. These are still relatively low, however, possibly due to a lack of training data caused by rebalancing the data set. The classification results for subsequences obtained from the automatically annotated data, shown in Table 3.3b, show an identical trend.

	BW	*MA*	*cBW*	*cMA*	*cMA - balanced*
Storing groceries	1	1	1	0.9167	0.6364
Preparing dinner	0.7565	0.9336	0.7823	0.7417	0.1667
Eating	0.9259	0.9012	0.9506	0.037	1
Doing dishes	0.9231	0.8462	0.7538	0.1692	0.7273
Vacuuming	1	1	1	0.5313	0.9167
Preparing a drink	0.6667	0.6667	0.6667	0.3333	0.8333
Class accuracy	0.8787	0.8913	0.8589	0.4549	0.7134
Overall accuracy	0.8257	0.9108	0.8216	0.5187	0.7143

(a) Manually annotated data

	BW	*MA*	*cBW*	*cMA*	*cMA - balanced*
Storing groceries	0.7857	0.8571	0.7143	1	1
Preparing dinner	0.8179	0.9072	0.9003	0.7388	0.0714
Eating	0.7778	0.8642	0.9506	0	1
Doing dishes	0.8108	0.6892	0.7568	0.1622	0.3077
Vacuuming	0.9512	0.878	0.8049	0.122	0.7857
Preparing a drink	0.5714	0.1786	0.25	0.3571	0.6
Class accuracy	0.7858	0.7291	0.7295	0.3967	0.6275
Overall accuracy	0.8072	0.828	0.8412	0.4839	0.6265

(b) Automatically annotated data

Table 3.3: Sequence-based activity classification accuracy rates on manually annotated data and automatically annotated data using subsequences of 50 emissions each. The tables list the accuracies using the BW algorithm, the MA algorithm, the coupled BW algorithm (cBW), the coupled MA algorithm (cMA), and the coupled MA algorithm using the rebalanced data set (cMA - balanced).

3.4.2 Audio-based classification

We can also explore to what degree classification is possible if only a single modality is taken into consideration. In this section, we look at classification through audio events only. Table 3.4a shows the results for full-length sequence-based classification on the manually annotated data set, using only observations obtained from audio sources. Note that the coupled classification methods (coupled BW and coupled MA) are not included in the results. As there is only a single modality present, the coupled methods would be reduced to consisting of only a single chain, and as such would be equivalent to their non-coupled alternatives. Comparing these results to those

	BW	*MA*
Storing groceries	1	1
Preparing dinner	0.9375	0.9375
Eating	0.6667	1
Doing dishes	0.9091	0.9091
Vacuuming	1	1
Preparing a drink	0.7143	0.8571
Class accuracy	0.8713	0.9506
Overall accuracy	0.8636	0.9394

(a) Manually annotated data

	BW	*MA*
Storing groceries	0.75	1
Preparing dinner	0.875	1
Eating	0.4444	0
Doing dishes	0.9091	0.6364
Vacuuming	1	1
Preparing a drink	0.5	0
Class accuracy	0.7464	0.6061
Overall accuracy	0.7424	0.5909

(b) Automatically annotated data

Table 3.4: Audio only sequence-based activity classification accuracy rates on manually annotated data and automatically annotated data using full-length sequences. The tables list the accuracies using the BW algorithm and the MA algorithm.

obtained using both modalities shows little difference between accuracy scores, especially for the MA method. This seems to indicate that for the manually annotated data and full sequence lengths, audio information alone is sufficient for purposes of classification.

The results for full-length sequence-based classification on the automatically annotated data, shown in Table 3.4b, show a more substantial difference to those obtained for the automatically annotated data using both modalities. For both methods, classification accuracy is well below that of their equivalent methods, and even further below the optimal classification accuracy obtained on the automatically annotated data using the coupled MA method. The class accuracy scores are also considerably below the class accuracy scores obtained on the manually annotated data

for audio only, especially for the MA algorithm. A possible explanation is that by themselves, the audio observations lack sufficient information for classification in the noisier automatically annotated data set. The poor performance of the MA algorithm specifically might be attributed to the poor performance on 'eating' and 'preparing a drink'; these classes were seemingly never assigned by the algorithm.

When splitting the full-length sequences into subsequences of 50 emissions each, the general trend remains similar. The results for classification on subsequences are shown in Table 3.5a for the manually annotated data, and Table 3.5b for the automatically annotated data. Again, the results on the manually annotated data show class accuracies close to those obtained for classification using both audio and video. Also, the results obtained for the automatically annotated data show class accuracies lower than those obtained for audio and video classification, and again, the MA algorithm performs poorly on the 'eating' and 'preparing a drink' activities. As was the case for the video and audio classification, the classification results on subsequences are overall lower than the classification results on full-length sequences.

3.4.3 Video-based classification

In this section, classification based on video observations only is examined. The results for full-length sequence-based classification on the manually annotated data are shown in Table 3.6a. As for audio-only classification, the coupled BW and coupled MA algorithms are omitted, for reasons discussed in the previous section. The results show the class accuracy scores for both algorithms are considerably lower compared to those obtained using audio and video classification and audio-only classification. This may indicate that video observations alone provide insufficient information to accurately distinguish between the selected activities, even on the relatively low-noise manually annotated data.

The above trend persists in the classification results on full-length sequences obtained from the automatically annotated data, shown in Table 3.6b. The class accuracy scores for both algorithms are considerably lower compared to those obtained using audio and video classification, and they are lower overall compared to the results obtained using audio-only classification. The exception is that the MA algorithm yields a higher class accuracy compared to the audio-only classification due to the 'eating' activity being better recognized.

The above patterns extend to the classification results using subsequences of 50 emissions. The subsequence classification results on the manually annotated data are shown in Table 3.7a, while the classification results on the automatically annotated data are shown in Table 3.7b. Again, the class accuracy scores are consistently lower than those obtained using audio and video classification. The same holds true when compared to the audio-only classification, excepting the MA algorithm on the automatically annotated data. Overall, it can be concluded the video-only classification performs poorest of the three methods discussed.

	BW	*MA*
Storing groceries	1	1
Preparing dinner	0.7778	0.9778
Eating	0.925	0.75
Doing dishes	0.7931	0.6552
Vacuuming	1	1
Preparing a drink	0.75	0.625
Class accuracy	0.8743	0.8347
Overall accuracy	0.8217	0.887

(a) Manually annotated data

	BW	*MA*
Storing groceries	0.8571	0.7143
Preparing dinner	0.4533	0.9267
Eating	0.8049	0.0732
Doing dishes	0.6757	0.6216
Vacuuming	1	0.8889
Preparing a drink	0.5455	0.0909
Class accuracy	0.7228	0.5526
Overall accuracy	0.5909	0.7083

(b) Automatically annotated data

Table 3.5: Audio only sequence-based activity classification accuracy rates on manually annotated data and automatically annotated data using subsequences of 50 emissions each. The tables list the accuracies using the BW algorithm and the MA algorithm.

3.4.4 Discussion

The results outlined in the previous sections show that the combined modalities of audio and video yield higher classification accuracies than methods using just audio or video for classification. Video-only classification results indicate relatively poor classification ability compared to the other methods. The classification accuracies obtained for audio-only classification on the other hand score similarly to those obtained for audio and video classification on the manually annotated data. For the automatically annotated data, however, the audio-only classification performs considerably worse compared to the audio and video classification.

	BW	*MA*
Storing groceries	0.875	0.5
Preparing dinner	0.9375	0.875
Eating	0.3333	0.8889
Doing dishes	1	1
Vacuuming	0.875	0.125
Preparing a drink	0.7143	0.2143
Class accuracy	0.7892	0.6005
Overall accuracy	0.803	0.6212

(a) Manually annotated data

	BW	*MA*
Storing groceries	0.75	0.875
Preparing dinner	1	1
Eating	0.1111	0.6667
Doing dishes	0.8182	0.8182
Vacuuming	1	0.75
Preparing a drink	0.2143	0
Class accuracy	0.6489	0.6850
Overall accuracy	0.6515	0.6667

(b) Automatically annotated data

Table 3.6: Video only sequence-based activity classification accuracy rates on manually annotated data and automatically annotated data using full-length sequences. The tables list the accuracies using the BW algorithm and the MA algorithm.

Intuitively, it might be surprising that the video-only classification performs worse compared to the audio-only classification, as video images are often considered to contain a higher amount of information compared to audio. However, it should be noted that only the location of the participant was extracted from the video information; as a result, the actual information content of the video events may be considerably more restricted compared to the information that would be derived by for example a human observer. It is feasible that if more advanced video scene analysis techniques were used, the performance of video-only classification could be improved.

When comparing the BW and MA methods (and their coupled equivalents), it is difficult to say from these results that one method clearly outperforms the other.

	BW	*MA*
Storing groceries	0.6667	0
Preparing dinner	0.7969	0.8516
Eating	0.9167	0.9444
Doing dishes	0.8276	0.7931
Vacuuming	0.5	0.3571
Preparing a drink	0.6667	0.1667
Class accuracy	0.7291	0.5188
Overall accuracy	0.7963	0.7963

(a) Manually annotated data

	BW	*MA*
Storing groceries	0.8333	0.3333
Preparing dinner	0.9398	0.9549
Eating	0.1111	0.8333
Doing dishes	0.8286	0.8286
Vacuuming	0.8947	0.7895
Preparing a drink	0	0
Class accuracy	0.6013	0.6233
Overall accuracy	0.7595	0.8565

(b) Automatically annotated data

Table 3.7: Video only sequence-based activity classification accuracy rates on manually annotated data and automatically annotated data using subsequences of 50 emissions each. The tables list the accuracies using the BW algorithm and the MA algorithm.

Overall, the MA method seems to do better when there is a high amount of information available in the data; that is, when operating on full-length sequences and manually annotated events. On shorted sequences, the performance of the MA algorithms seems to drop compared to the BW algorithms, although this can partially be explained due to the unbalance introduced in the data set. The performance of the MA algorithms is also lowered compared to the BW algorithms on automatically annotated data, although on full length sequences, the MA algorithms still give the best performance. Interestingly, the trend of lower performance of the MA algorithm on automatically annotated data is reversed for the video-only classification, although performance on this data set is very low in general.

For sequence-based classification, the cHMM algorithms do not seem to offer better performance compared to their non-coupled equivalents - the exception being the coupled MA algorithm on automatically annotated, full-length sequences, which is the only algorithm performing well on the 'preparing a drink' activity. As a result, the theory that exploiting the causal link between audio and video events can yield improved performance has not proven to be the case for this data set. Overall, a recommendation could be that for data sets with a balanced set of observations or little noise, the regular MA algorithm might be the best choice, and to choose the regular BW algorithm otherwise. However, the differences in accuracy between these methods is often small; as both offer high accuracy in many cases, either could work well in a sequence-based activity recognition application.

3.5 Session-based online activity classification

In contrast to the sets of well-defined observation sequences discussed in the previous section, this section focuses on recognizing activities from a continuous stream of audio and video data. As a result, there is no indication when one activity ends and another begins. In addition, the data stream may contain periods of inactivity (e.g., no one is present) or of activities other than those the system is trained to recognize.

To simulate the situation described above, the entire data streams of single recording sessions (see Section 3.3 on gathering data) are used. When activity recognition is being performed on one of the eight recorded sessions, the remaining seven sessions are used to train the activity recognition models (i.e., estimate model parameters), resulting in an analysis method similar to cross validation. The activity recognition algorithm uses a sliding window of a fixed length to store incoming data read from the data stream(s). Whenever the contents of the window are updated, activity recognition is performed on the current contents of the sliding window. In other words, activity recognition is performed over a history of the most recent observations.

In the experiments described in this section, a window size of 50 observations was used, resulting in observation sequences of identical length. As described in Section 3.3.3, a sequence length of 50 observations corresponds to at most 25 seconds of observations. Activity recognition was performed whenever the contents of the sliding windows were shifted by two new observations. Ideally, these two new observations represent both a single new audio event and a single new video event, although in practice this is not necessarily the case (for example, potentially when two new sounds follow each other within the space of a second).

Whenever activity recognition is performed, the result is compared with the ground truth which has been annotated. Accuracy is defined as the fraction of matches between the recognized activity and the ground truth over all windows in the session. Windows for which the ground truth activity is 'other' are discarded for this purpose, as the recognition algorithm cannot recognize this class of activity. The

average accuracy over all sessions is computed to determine the overall accuracy of the activity recognition algorithm.

As the average duration of the activity classes attempted to recognize varies, some activities become over-represented when considering the recognition results of each sliding window individually. As a result, the overall accuracy may not be indicative of the recognition accuracy of individual activities. As such, two accuracy measures will be used: 'standard' accuracy as described above (also referred to as time-slice accuracy in this context) and class accuracy, obtained by computing the accuracies for each activity class individually and computing the mean over these accuracies. The time-slice accuracy and class accuracy measures were previously mentioned in Section 3.3.4, which also includes the definitions of the two accuracies.

As an example, the activity recognition results for one of the sessions have been visualized in Figure 3.4. The horizontal axis represents the sliding window contents on which activity recognition was performed in chronological order. The vertical axis lists the various activity classes. The dashed, red line represents the annotated ground truth, while the blue line represents the recognized activity. When considering the relation between the recognized activity and the ground truth, it can be seen there is a 'delay' between the ground truth and the recognized activity; where a change in the ground truth only results in a change in the recognized activity several windows later. This is a result of using a history of observations: when the ground truth changes, there are at most two observations in the current window which represent the new activity. As such, 48 observations related to the previous activity still remain, causing events produced by the previous activity to dominate events produced by the current activity[11]. As the sliding window is updated, more observations of the current activity enter the window, while the older observations are pushed out, until eventually the current activity becomes dominant.

Theoretically, the point at which both the current and previous activity are equally represented occurs when there is an equal number of observations of both activities in the window. This situation, one might expect, is where the recognition algorithm switches from one activity to another. In practice, the distinguishing value of individual observations also plays a role. Observations which are highly common in one activity while uncommon in another will have a greater influence on the recognition result than observations which are equally common in both activities.

[11] By themselves, hidden Markov models already impose a kind of exponential weighting that can help with this on older observations when determining the probability of being in a certain current state. When comparing to overall probabilities of two different models, however, there is no such weighting in effect. One option would be to build a model architecture where certain states represent activities, and essentially have all activity models interconnected - this is called a hierarchical hidden Markov model. The downside of such a single model for all activities is that the number of parameters to be estimated is much larger.

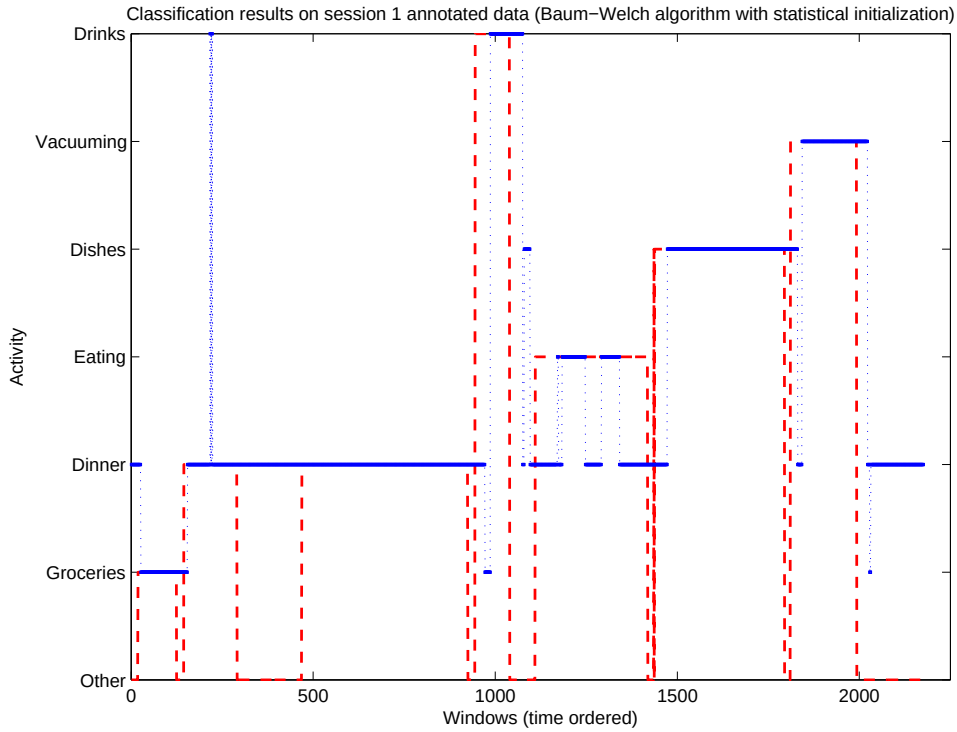


Figure 3.4: Example of activity recognition results on a recorded session over time. The dashed red line in the graph indicates the ground truth activity at each time step during the recording, while the blue line indicates the recognized activity at each of those time steps. An overlap between the two lines indicates an accurate recognition; a difference indicates a recognition error.

For example, 'walking around the kitchen' (the 'transition' video observation) is common both to 'preparing a drink' and 'vacuuming', however the sound of a vacuum cleaner is strongly associated with the latter. Therefore, even a few observations of the sound of the vacuum cleaner are likely to impact the recognition result considerably. Therefore, the presence of distinguishing observations can cause the switch in the recognized activity to occur earlier or later. As nearly all observations are distinguishing to some extent, this switch seldom occurs exactly at the halfway mark.

As the timing of a switch in the recognized activity is influenced by the number of observations considered, it can be reasoned that the delay between ground truth activity and recognized activity increases as the size of the sliding window increases. Figure 3.5 shows an example of the activity recognition results using the same parameters as in Figure 3.4, except for using a sliding window half the original size. Indeed, it can be seen that the delay between changes in the ground truth activities and corresponding changes in the recognized activity is reduced for a smaller win-

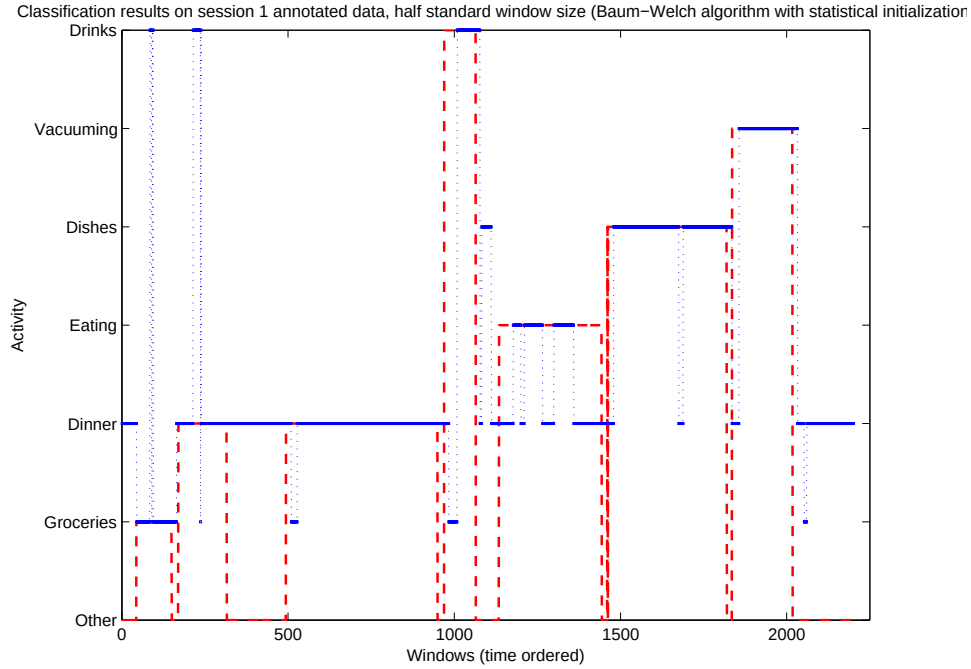


Figure 3.5: Example of activity recognition results on a recorded session over time, using half the normal sliding window size. The dashed red line in the graph indicates the ground truth activity at each time step during the recording, while the blue line indicates the recognized activity at each of those time steps. An overlap between the two lines indicates an accurate recognition; a difference indicates a recognition error.

dow size. However, it can also be seen that errors not related to this delay appear to be more numerous.

To summarize, a larger sliding window size results in more errors due to the aforementioned delay, but fewer errors when the ground truth activity is stable, as there is more information in the sliding window for the activity recognition algorithm to draw on. Therefore, trade-off must be made: if the window size is too small, it contains too little information to base recognition on. If too large, changes in the ground truth will be detected too late or not at all. In either case, the overall error will likely become larger.

A reason to favor larger window sizes slightly over smaller window sizes is the issue of stability[12]. Smaller window sizes result in larger frequencies of changes of

[12]In machine learning, stability (or algorithmic stability) refers to the extend by which the output of an algorithm is affected by small changes in input. If an algorithm is stable, its output will only show small changes in response to a small change in input.

the recognized activity, which in practical applications is often undesirable. Using a larger window size results in a more stable activity recognition result over time, as there are fewer short-lived changes between activities, as can also be seen in Figures 3.4 and 3.5. Even though stability has no direct impact on accuracy scores, it can in practice be beneficial to select a larger than optimal window size[13].

3.5.1 Audio-based and video-based classification

Making use of the methods described above, the result of the accuracies over each session and averages over all eight sessions using the manually annotated data is shown in Tables 3.8a and 3.8b. On the manually annotated data set, the timeslice accuracies, shown in Table 3.8a, of the various methods are very similar, with differences of less than 1%. Indeed, when applying a repeated measures ANOVA test, we can see that there are no significant differences between the four methods overall ($F(3,21) = 0.07$, $p = 0.97$). Similarly, pairwise t-tests do not show any significant differences between individual methods.

When one considers the class accuracies for the four methods, shown in Table 3.8b, the differences between the methods are somewhat more pronounced compared to the timeslice accuracy. For the Baum-Welch method, the class accuracy is similar to the timeslice accuracy, indicating there is little bias towards any imbalanced classes. This is not the case for the MA and coupled Baum-Welch methods, where there is a more substantial difference between timeslice and class accuracy. This bias seems to be exacerbated for the coupled MA method, having the lowest class accuracy of the four methods. Even so, the class accuracy remains at an acceptable level, approximately 6% lower compared to the timeslice accuracy. The repeated measures ANOVA test does show that there is a significant difference in class accuracy between the four methods ($F(3,21) = 4.38$, $p = 0.015$), likely related to the relatively poor performance of the coupled MA algorithm. Pairwise t-tests also show no further significant differences between the individual methods after Bonferroni correction, however.

When considering the accuracy scores of the individual sessions, it can be seen there are considerable differences between sessions, and also between methods for the same session. However, some sessions appear to elicit higher accuracy scores than others, regardless of methods used. A good example of this is Session 3. It therefore appears that some sessions are inherently more difficult to classify than others, likely due to individual differences between participants.

This is supported when we examine the intraclass correlation for the four methods. For the timeslice accuracy, we find an intraclass correlation of 0.96 ($F(7,21) = 94.1$, $p \ll 0.001$) using a two-way model, which indicates close to perfect

[13] It is also possible to use some type of filtering to smoothen the changes in the recognized activities. Depending on the filter used, this will delay the output of the activity recognition algorithm, however.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	0.7406	0.8668	**0.9532**	0.7905	0.7847	**0.8860**	0.8082	**0.4897**	**0.7900**
MA	0.7142	0.8841	0.9439	0.7811	**0.8666**	0.8337	**0.8577**	0.4721	**0.7942**
cBW	**0.8194**	0.8775	0.9422	0.7937	0.8225	0.8249	0.7970	0.4663	**0.7929**
cMA	0.7261	**0.8856**	0.9312	**0.8177**	0.8519	0.8253	0.8457	0.4889	**0.7966**

(a) Timeslice accuracy

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	0.6962	**0.8283**	**0.8591**	**0.7457**	0.8163	**0.8663**	0.7420	**0.7422**	**0.7870**
MA	0.6460	0.8095	0.8334	0.6960	**0.8264**	0.7649	**0.7574**	0.6409	**0.7468**
cBW	**0.7678**	0.7940	0.8199	0.7283	0.8061	0.7570	0.7306	0.7196	**0.7654**
cMA	0.6649	0.8066	0.7848	0.7160	0.7836	0.7022	0.7416	0.6457	**0.7307**

(b) Class accuracy

Table 3.8: Session-based activity classification accuracy rates on manually annotated data using audio-based and video-based classification. The tables show the accuracies per session, numbered 1 through 8, and the average accuracies over all eight sessions. The tables show accuracy results for the Baum-Welch algorithm (BW), the MA algorithm, the coupled Baum-Welch algorithm (cBW) and the coupled MA algorithm (cMA). For each session, the most accurate classification is highlighted.

consistency in accuracy scores between the models. This suggests that much of the variance in the individual accuracies is related to differences between participants. For the class accuracy scores, the intraclass correlation is 0.71 ($F(7,21) = 10.9$, $p \ll 0.001$); while the intraclass correlation is not as substantial as for the timeslice accuracy, this still indicates a strong consistency between the methods.

Using the same methods, the results for the automatically annotated data are shown in Table 3.9a and 3.9b. Using the repeated measures ANOVA test, we can see that there are significant differences between the four methods for the timeslice accuracy ($F(3,21) = 7.78$, $p = 0.0011$). However, there are no significant differences between the class accuracies ($F(3,21) = 1.2$, $p = 0.33$). The significant differences on the timeslice accuracy are likely due to the relatively poor performance of the Baum-Welch algorithm in this aspect. This seems to be confirmed by pairwise t-tests between the methods. On the timeslice accuracy, pairwise tests show significant differences after correction for the Baum-Welch method compared to the MA method ($p = 0.0075$). There are no significant pairwise differences for the class accuracy scores. Intraclass correlation remains high for the timeslice accuracy with a correlation of 0.81 ($F(7,21) = 17.7$, $p \ll 0.001$). However, the same can not be said of the intraclass correlation for the class accuracy, which is quite low at 0.21 ($F(7,21) = 2.06$, $p = 0.094$).

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	0.6155	0.7221	0.9023	0.6621	0.7295	0.7471	0.7613	0.7142	**0.7318**
MA	**0.7825**	0.7664	**0.9299**	0.7106	0.7530	0.8006	**0.8084**	0.7800	**0.7914**
cBW	0.7454	0.8265	0.9112	0.6447	0.7524	**0.8206**	0.7776	0.7773	**0.7820**
cMA	**0.7825**	**0.8437**	0.9205	**0.7467**	**0.7548**	0.8101	0.7668	**0.7826**	**0.8010**

(a) Timeslice accuracy

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	0.6517	0.5955	0.6416	0.6599	0.5491	0.6864	**0.7651**	0.6460	**0.6494**
MA	0.7243	0.6166	**0.7458**	0.6375	0.5201	**0.7488**	0.6635	**0.6509**	**0.6634**
cBW	**0.7432**	0.6476	0.6127	0.6376	**0.5761**	0.6479	0.6990	0.6071	**0.6464**
cMA	0.7427	**0.7382**	0.7191	**0.7128**	0.5305	0.7214	0.6648	0.6232	**0.6816**

(b) Class accuracy

Table 3.9: Session-based activity classification accuracy rates on automatically annotated data using audio-based and video-based classification. The tables show the accuracies per session, numbered 1 through 8, and the average accuracies over all eight sessions. The tables show accuracy results for the Baum-Welch algorithm (BW), the MA algorithm, the coupled Baum-Welch algorithm (cBW) and the coupled MA algorithm (cMA). For each session, the most accurate classification is highlighted.

When considering the timeslice accuracy, it is clear the Baum-Welch method performs poorly compared to the other methods (as also indicated by the pairwise tests). In contrast, the accuracies of the other three methods are much more similar to those obtained for the manually annotated data. However, none of the methods show any significant differences in pairwise tests (once again using the Wilcoxon signed rank test) between the timeslice accuracy scores on manually annotated data compared to automatically annotated data. For all methods, class accuracy is considerably lower compared to those obtained for the manually annotated data. However, this difference is only significant after Bonferroni correction for the coupled Baum-Welch method ($p = 0.0078$), and close to significant for the regular Baum-Welch method ($p = 0.0156$). Closer inspection of individual class accuracy scores indicates this is partly due to poor recognition on the 'preparing a drink' activity class. The high variance in accuracy scores on this class might also offer an explanation for the differences in intraclass correlation between the timeslice accuracy and class accuracy scores on the automatically annotated data, as the 'preparing a drink' class is not a dominant class in the data set.

Remarkably, the coupled MA method shows the highest class accuracy on the automatically annotated data, while showing the lowest on the manually annotated

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
man	0.7515	0.8053	0.8933	0.7571	0.7700	0.7705	0.7498	0.4253	**0.7404**
aut	0.7804	0.5667	0.8826	0.5709	0.7000	0.7672	0.7514	0.7172	**0.7171**

(a) Timeslice accuracy

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
man	0.7426	0.7744	0.7938	0.7012	0.8358	0.7849	0.7949	0.6802	**0.7635**
aut	0.8027	0.6056	0.8248	0.7086	0.6110	0.8157	0.7593	0.7082	**0.7295**

(b) Class accuracy

Table 3.10: Session-based activity classification accuracy rates for the coupled MA algorithm using a rebalanced data set. The tables show the accuracies per session, numbered 1 through 8, and the average accuracies over all eight sessions. In the tables, 'man' represents the results on the manually annotated data, 'aut' represents the results on the automatically annotated data.

data. This is similar to the results obtained in Section 3.4. A possible explanation is a greater robustness against noise introduced by the audio and video classification on certain activities. Also of interest when comparing the automatically annotated data to the manually annotated data are the results on session 8, which shows unusually low timeslice accuracy scores on the manually annotated data. However, this effect is not replicated on the automatically annotated data, possibly indicating an error in the manual annotation of this session, most likely on a dominant activity class such as 'preparing dinner'.

As became apparent from the first runs using the coupled MA algorithm, this algorithm is highly susceptible to the effects of class imbalance. The results of the coupled MA algorithm shown in Tables 3.10a and 3.10b have been obtained using a rebalanced data set, as described in Section 3.3.4. Tests on rebalanced sets have also been performed using the other algorithms described in this section, however these yielded no increases in recognition accuracy, and therefore are not listed. Compared to the respective results previously obtained for the coupled MA algorithm, rebalancing the data set results in lower timeslice accuracies, but improved class accuracy scores. Pairwise tests do not show that these improvements are significant, however.

For the Baum-Welch and MA methods, it is also possible to examine the effect of the different model initialization algorithms, evolutionary and statistical (see Section 3.3.5). The results are shown in Table 3.11a. The differences in accuracy resulting from these initialization methods are very small, less than 1%. The results for class accuracies, shown in Table 3.11b, show a similar trend, with the largest difference between both initialization methods less than 2%. It can be concluded

	Evolutionary method	*Statistical method*
BW timeslice accuracy (manual)	0.7900	0.7997
MA timeslice accuracy (manual)	0.7942	0.7969
BW timeslice accuracy (automatic)	0.7318	0.7370
MA timeslice accuracy (automatic)	0.7914	0.7924

(a) Timeslice accuracy

	Evolutionary method	*Statistical method*
BW class accuracy (manual)	0.7870	0.7867
MA class accuracy (manual)	0.7468	0.7599
BW class accuracy (automatic)	0.6494	0.6616
MA class accuracy (automatic)	0.6634	0.6639

(b) Class accuracy

Table 3.11: Overview of the activity classification accuracy rates using the evolutionary model initialization method and the statistical model initialization method, both on the manually annotated (manual) and the automatically annotated (automatic) data, and examined for the Baum-Welch and MA algorithms.

therefore, that both initialization methods result in comparable models for both the Baum-Welch and the MA methods for audio and video recognition.

3.5.2 Audio-based classification

The previous section describes a method using both audio and video observations to recognize activities. It is also possible to perform a similar experiment using only a single modality, in this case audio. To achieve this, the method described has been altered on some points. Most importantly, only observations originating from the audio stream are considered, similarly as in Section 3.4.2. This reduces the total number of observation classes to 21.

Discarding the video stream causes the minimum number of observations per second to drop from two observations to a single observation (see Section 3.3.3). Accordingly, the size of the sliding window is halved to 25 observations, so as to represent the same time span as before. Empirical evidence seems to suggest this reduction improves performance, as recognition in the audio-only case using a window size of 50 observations would suffer greatly from errors caused by delays between ground truth changes and corresponding recognition result changes (see Section 3.5). Using similar reasoning as above, the recognition result is updated whenever a single new observation enters the sliding window, rather than only for every two new observations.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	**0.8871**	**0.8011**	0.7937	**0.8021**	0.7420	**0.8103**	0.7899	0.4182	**0.7556**
MA	0.8659	0.7907	**0.8442**	0.6917	**0.8129**	0.7936	**0.7961**	**0.7812**	**0.7970**

(a) Timeslice accuracy

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	**0.8416**	**0.8215**	**0.7944**	**0.7597**	**0.8083**	**0.7900**	**0.7506**	0.6586	**0.7781**
MA	0.8201	0.7031	0.6957	0.6734	0.7696	0.6776	0.6814	**0.7052**	**0.7158**

(b) Class accuracy

Table 3.12: Session-based activity classification accuracy rates on manually annotated data using audio-only classification. The tables show the accuracies per session, numbered 1 through 8, and the average accuracies over all eight sessions. The tables show accuracy results for the Baum-Welch algorithm (BW) and the MA algorithm. For each session, the most accurate classification is highlighted.

The results for the audio-only timeslice accuracy and class accuracy on the manually annotated data are shown in Tables 3.12a and 3.12b, respectively. As there is little point in using coupled methods when considering a single modality (as this would imply a single chain, the resulting models would be equivalent to those produced by the appropriate non-coupled methods), they are omitted here. As was the case for audio-based and video-based classification, the MA method achieves high accuracy, but relatively poor class accuracy. Overall, the accuracies achieved are similar to those achieved for audio-based and video-based classification on manually annotated data. This is supported by t-tests comparing the accuracy scores for audio-only to their counterparts using both modalities (using Bonferroni correction), as there are no significant differences between the sets of modalities. For the class accuracy scores however, the difference between the Baum-Welch and MA methods is significant ($p = 0.0158$).

There appears to be more variation regarding the results for timeslice and class accuracy on the automatically annotated data, shown in Tables 3.13a and 3.13b, respectively. The results on both timeslice and class accuracy are considerably lower than those obtained using audio-based and video-based classification. Using t-tests, we can see that for all methods, there are significant differences (after Bonferroni correction) between the accuracy scores using both modalities, and using audio only. This holds for both the timeslice accuracy ($p = 0.0012$ for the Baum-Welch method, $p = 0.0060$ for the MA method) and the class accuracy ($p = 0.0072$ for Baum-Welch, $p < 0.001$ for MA). There were no significant differences between the methods themselves.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	0.4447	0.4474	0.4459	0.5781	**0.6094**	0.5981	0.4067	0.4214	**0.4940**
MA	**0.5837**	**0.6664**	**0.7852**	**0.7273**	0.3333	**0.6877**	**0.5802**	**0.4467**	**0.6013**

(a) Timeslice accuracy

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	0.5524	**0.5385**	0.4809	**0.5836**	**0.5557**	**0.6441**	**0.5370**	**0.5272**	**0.5524**
MA	**0.6031**	0.4553	**0.5763**	0.5226	0.3266	0.5347	0.4142	0.4117	**0.4806**

(b) Class accuracy

Table 3.13: Session-based activity classification accuracy rates on automatically annotated data using audio-only classification. The tables show the accuracies per session, numbered 1 through 8, and the average accuracies over all eight sessions. The tables show accuracy results for the Baum-Welch algorithm (BW) and the MA algorithm. For each session, the most accurate classification is highlighted.

Similar to the results on the manually annotated data, the results of audio-only-classification on automatically annotated data show relatively high timeslice accuracy, but relatively low class accuracy for the MA method. Also, the results on the manually annotated data yield considerably higher accuracy scores compared to the results on the automatically annotated data. This may indicate that while accurate classification with audio only is possible on a clean data set, the additional noise present in the automatically annotated data set means audio-only classification is severely impeded. This result is consistent with the results obtained in Section 3.4.

3.5.3 Video-based classification

Using methods similar to those discussed in the previous section on audio-only classification, it is also possible to perform activity recognition based purely on video-based observations. The method used differs only in that only observations originating from the video stream are considered in this case. Using similar reasoning as in the previous section, the sliding window size was reduced to 25 observations, and activity recognition is performed whenever a single new observation enters the sliding window. By taking only video observations into account, the total number of observations classes is reduced to 8.

The results are shown in Tables 3.14a and 3.14b, listing the timeslice accuracy and the class accuracy respectively for the manually annotated data. As for audio-based classification, the coupled HMM methods have not been included. The results show a considerable difference between timeslice accuracy and class accuracy, the latter being relatively low compared to the class accuracies obtained on manually

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	0.5436	0.7427	0.8930	**0.7534**	0.6545	**0.7737**	**0.8077**	0.3957	**0.6955**
MA	**0.5549**	**0.7978**	**0.8952**	0.7172	**0.7811**	**0.7737**	0.6919	**0.4111**	**0.7029**

(a) Timeslice accuracy

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	**0.4864**	0.5921	**0.6550**	**0.5693**	0.5504	**0.5880**	**0.6244**	0.4280	**0.5617**
MA	0.3751	**0.6507**	0.6147	0.4220	**0.6226**	0.5037	0.4289	**0.4303**	**0.5060**

(b) Class accuracy

Table 3.14: Session-based activity classification accuracy rates on manually annotated data using video-only classification. The tables show the accuracies per session, numbered 1 through 8, and the average accuracies over all eight sessions. The tables show accuracy results for the Baum-Welch algorithm (BW) and the MA algorithm. For each session, the most accurate classification is highlighted.

annotated data using audio and video classification or audio-only classification. The differences between accuracy for video-only classification compared to using both modalities are significant after correction using paired t-tests for timeslice accuracy ($p = 0.0033$ for Baum-Welch, $p < 0.001$ for MA) and class accuracy ($p < 0.001$ for both methods). Again, the MA method shows lower class accuracy compared to the Baum-Welch method on manually annotated data, although there are no significant differences between the methods for both timeslice and class accuracy.

A similar trend can be observed for recognition results on the automatically annotated data, show in Tables 3.15a (timeslice accuracy) and 3.15b (class accuracy). Again, there is a considerable disparity between timeslice accuracy and class accuracy for both methods, as was also the case with the results obtained in Section 3.4. A possible explanation is video-only classification is able to distinguish only a few activities accurately, while being nearly unable to identify the remainder reliably. Looking through individual session / activity accuracies seems to indicate this is the case, with (in general) especially poor recognition rates for 'storing groceries' and 'preparing a drink'. It is possible the 8 observation classes offer insufficient distinguishing information, causing the training methods to learn to start 'guessing' the most commonly occurring activity in this case, which would explain the relatively high timeslice accuracy. Alternatively, it is possible that only some of the activity classes can be accurately recognized using purely video observations, and that these turn out to be the most common ones in the data sets.

Overall, there are no significant differences between the accuracy scores of both methods on the automatically annotated data as indicated by paired t-tests, for both

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	**0.7020**	0.6932	0.8703	0.4707	0.7230	0.5979	0.7535	**0.7621**	**0.6966**
MA	0.5666	**0.7125**	**0.9042**	**0.6049**	**0.7751**	**0.7531**	**0.7608**	0.7593	**0.7296**

(a) Timeslice accuracy

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*acc*
BW	**0.5408**	**0.5635**	0.5722	0.4240	0.4134	0.4874	0.5097	0.5023	**0.5017**
MA	0.3751	0.5331	**0.6643**	**0.4589**	**0.5472**	**0.5787**	**0.5400**	**0.5245**	**0.5277**

(b) Class accuracy

Table 3.15: Session-based activity classification accuracy rates on automatically annotated data using video-only classification. The tables show the accuracies per session, numbered 1 through 8, and the average accuracies over all eight sessions. The tables show accuracy results for the Baum-Welch algorithm (BW) and the MA algorithm. For each session, the most accurate classification is highlighted.

the timeslice accuracy. As mentioned, the timeslice accuracy is relatively high, and there are no significant differences when comparing the video-only timeslice accuracy compared to using both modalities. The reverse is the case for the class accuracy scores however, as for both methods there exist significant differences after correction when comparing the video-only class accuracies with the audio and video class accuracies ($p = 0.0011$ for Baum-Welch, $p = 0.0091$ for MA).

3.5.4 Discussion

From the results discussed in this section, it is clear that session-based activity classification using both audio and video is more accurate on automatically annotated data compared to using only a single modality, based on the class accuracy scores. The highest overall classification on the automatically annotated data was obtained by the coupled MA algorithm using a rebalanced data set, resulting in a class classification accuracy of approximately 73%. In comparison, the highest classification accuracy for audio-only classification is 55%, and for video-only classification 53%. It can therefore be concluded that the combination of both modalities yields a considerable improvement in classification on data obtained from lower level classifiers.

On the manually annotated data, the highest overall classification accuracy using both audio and video is approximately 79%, compared to 78% for audio-only classification, and 56% for video-only classification. For the relatively less noisy manually annotated data (compared to the automatically annotated data), the classification accuracies of the audio-video and audio-only classification are very similar. It therefore seems that given the availability of highly accurate training data, audio information

is sufficient to accurately classify users' activities. In practice, however, such data is unlikely to become available unless additional steps in audio and video scene analysis are made.

When comparing the four different activity classification methods, the accuracy scores themselves do not vary considerably, and there are few cases in which significant differences have been observed. However, it should be taken into account that the number of participants (8) is fairly low for the purposes of statistical analysis (we can note here that a statistical comparison between the four methods was not the primary objective of the study). To determine differences between the methods more accurately, a further study including a larger number of participants would be beneficial. Overall though, it seems that the variance between participants is larger compared to the variance between classification methods, and as such it seems reasonable to conclude that the accuracy differences between the methods are minor overall.

Due to the relatively small differences in classifier performance, it is once again hard to recommend any one classification method over the others. For the manually annotated data, the comparative performances are similar to those for the individual, 50 emission sequences in Table 3.3a in that the Baum-Welch methods match or outperform the MA methods, particularly the coupled MA algorithm which appears to struggle with class imbalance (having both the highest timeslice accuracy and the lowest class accuracy). As in Section 3.4, it appears that in general, the coupled variants add little additional value to the non-coupled variants for this application.

On the automatically annotated data, the MA algorithms perform slightly better compared to the Baum-Welch algorithms, with the coupled MA algorithm even showing the highest class accuracy in this case. However, compared to the 50 emission sequences in Table 3.3b, the overall accuracies are relatively low (in the order of 65% as opposed to 80%), and the individual differences between the methods are relatively small. In all, it was expected that the class accuracies would be less compared to those for the individual sequences, as the latter do not include activity transitions. Overall, the accuracies for the session-based classification are still substantially higher than what would be expected by chance, and all classifier methods seem to be effective in recognizing the different activities across participants.

Finally, while we find few differences between the individual classification methods, there are considerable differences between the different modalities. Here, we observe the same trend as in Section 3.4, with audio and video classification outperforming audio-only classification, which in turn outperforms video-only classification. The performance of the video-only classification is likely related to the fact that only location is derived from the video images - with the use of more advanced computer vision techniques, video as a single modality would likely yield better performance. On automatically annotated data in particular, the use of both modalities

confers significantly better accuracy, strongly suggesting that there is considerable benefit in the fusion of the two modalities in terms of ADL recognition. In some cases, it might be more convenient (or more accepted due to privacy concerns) to use only a single modality - in this case, a trade-off must be made between performance and the other perceived benefits.

A limitation of this study is that only activities performed by a single person are considered, with no other persons present in the room. This is less of a concern when monitoring people who live by themselves, although in this case one must still account for visitors. When multiple people are together, a means to distinguish between different people must be developed. Possible solutions can include face recognition, or the use of tags attached to the body (e.g., in clothing). In addition, the activity recognition algorithm will have to deal with additional noise generated by the presence of multiple persons, for example the sound of conversation, or camera occlusion. Effective ways of dealing with multi-user settings remains an open research questions for many applications.

Another limitation is that with a single camera and microphone, only part of most homes can be fully covered: in this case, we selected the kitchen as a room where a large number of ADLs take place. Naturally, this leaves the system unable to detect ADLs taking place outside of sensor range. This can also be seen as an advantage however, as kitchen-related ADLs are arguably less privacy-sensitive than for example, bedroom or bathroom-related activities. Kitchen-related ALDs have been included in scales to assess functional decline [Graf, 2008] and dementia [Bucks et al., 1996]; as such, it is likely that kitchen-related activities have some predictive value for these applications. Another possibility is to follow the approach of Urwyler et al. [2015], and use multiple sensor boxes placed around the home.

It should also be noted that while there is considerable interest in ADL detection for elderly living independently, this study was performed with a younger population. As such, the results found in this study may not generalize older populations. In addition, the study was performed in a simulated home environment, rather than the participants' actual homes. Potential future work could therefore include a study with elderly participants, in their own kitchen environment.

When comparing our results with the state of the art, it is difficult to make a direct comparison due to differences in the activities recognized and the sensors used. Konig et al. [2015] use a camera to detect six ADLs, in part by determining the position of a user in various zones inside the environment, with a recall of 85% and a precision of 75%. A possible explanation for these accuracy numbers is that there is likely a stronger relation between the set of ADLs investigated and the user's location in the environment (for example, the 'watching tv' ADL is likely related to presence in the 'tv zone').

Poularakis et al. [2015] obtain good results of approximately 90% accuracy using support vector machines on video images of a number of mostly low-level kitchen ADLs. It is possible that the lower level of the activities recognized plays a role in the relatively high accuracy (which they note is similar to other results on the same data sets), however, the results also seem to indicate that improved performance from computer vision methods is possible using more advanced techniques compared to the work described in this chapter.

3.6 Conclusion

In this chapter, an approach has been presented to classify a user's current activity captured by a single camera and microphone, using hidden Markov models. A number of techniques for constructing hidden Markov models have been outlined and tested experimentally. It was shown that for fully observed activities, a recognition (class) accuracy on the six trained activities of 96% can be achieved on data in which events were annotated by hand. On data obtained using scene analysis algorithms, a class accuracy of 93% was achieved. For activities in progress, using an event history of up to 25 seconds, a class accuracy can be achieved of 79% on data annotated by hand, and a class accuracy of 73% on data in which events were classified using video and audio scene analysis algorithms.

On the manually annotated data the 'standard' Baum-Welch algorithm shows the highest accuracy, while on the automatically annotated data the more complex coupled MA algorithm yields the highest class accuracy, provided issues with imbalanced data can be overcome. However, the differences between the different classifier methods are small, and not statistically significant. It was also shown that using a combination of audio and video events as input yields superior results compared to using only audio events, or only video events. This suggests that for ADL recognition, a fusion of the two modalities will lead to better results compared to the use of a single modality only.

4

Prediction of successful participation in a lifestyle activity program

4.1 Introduction

The work presented in this chapter has been previously published and discussed [Pijl et al., 2009; Long et al., 2014; Long et al., 2011], and related to a number of other publications [Long et al., 2009; van Halteren et al., 2014].

As mentioned in Chapter 1, an increasing amount of people lead an unhealthy lifestyle, with a lack of physical activity being one of the major contributors. As such a lifestyle can have serious implications for their future health (including increased risk of cardiovascular disease and diabetes), many people actively seek to increase their levels of physical activity throughout the day. However, long-term motivation can be difficult to maintain, also because the benefits of increasing one's level of physical activity are often not immediately visible. As a result, many fall back into old habits, and the gains made in lifestyle improvement are lost.

In response to this, programs have been developed to assist people in achieving a more sustainable change in their lifestyles. These so-called 'lifestyle physical activity programs' aim to encourage and support its participants in reaching and maintaining a healthy amount of daily physical activity. In this chapter, we consider participants taking part in a twelve-week lifestyle physical activity program, with the aim of determining the current, and particularly the future behavior of the participants. More specifically, we aim to identify those participants at risk of losing motivation and

dropping out of the program; by detecting these participants early, it may be possible to intervene in time and avoid the participant dropping out of the program altogether.

In this chapter, we address the second research question that was introduced in Chapter 1: Can analyzing the behavior of people trying to be more physically active help predict if they will drop out of a lifestyle physical activity program? We will also address the classification component of the model for human motion tracking described in the main research question, by introducing a method to distinguish between dropouts and adherent users of a lifestyle physical activity program.

In particular, the 'DirectLife' lifestyle physical activity program of 2009/2010 is considered here[1]. The program itself is also described by Goris and Holmes [2008], with examples of similar programs also being provided by Ware et al. [2008]. As part of this program, participants receive an activity monitor which measures their energy expenditure. After an assessment week in which their normal (pre-intervention) activity level is determined as a baseline, participants begin a twelve-week program during which they can monitor their activity level on a minute-by-minute basis through an online web interface after syncing their activity monitor to a computer, and can find advice and tips on how to improve their physical activity. Based on their baseline activity level, participants also receive weekly physical activity targets, which increase as the program advances, leading to a healthy (or at least healthier) level of physical activity towards the end of the twelve-week period. In addition, the program offers coaching support to assist and encourage participants where needed. At the end of the twelve-week program, participants have the option of continuing to use the service to maintain their level of physical activity, or to start a new twelve-week program.

Despite the tools and assistance offered by lifestyle physical activity programs, maintaining motivation to persist in the lifestyle changes suggested in such a program can often be a challenge. Loss of motivation is one of the main reasons why participants 'drop out' of the program prematurely, ultimately causing the efforts of the program to fail in effecting a lifestyle change. In the DirectLife program, this issue is partially mitigated by the coaches, who can intervene if a participant is likely to lose motivation. However, the number of participants (950 at the time of analysis) in the program at any given time is sufficiently large to make identifying who is at risk of dropping out a considerable challenge. Therefore, dropout prediction, the automatic identification of participants who are at risk of dropping out of the program, can be extremely helpful for targeted interventions, and will be the main focus of this chapter.

There are many possible reasons why participants might drop out of a program. In the literature, few predictors for adherence have been found, and fewer are uni-

[1] It should be noted that there are considerable differences with the current (2016) DirectLife program, including changes in the program structure, technology used, and participant population.

versally acknowledged. One of the main reasons for adherence is believed to be a participant's level of self-motivation, a concept that has long been associated with adherence to exercise programs [Dishman and Ickes, 1981]. Even so, this relationship is not universally acknowledged; Garcia and King [1991] find little relation between self-motivation and adherence to exercise regimen, instead noting that self-efficacy (the belief in one's ability to reach one's goals) seems to be a stronger predictor. Other predictors associated with adherence to physical exercise include intention, personal capabilities, behavioral skill, commitment and reinforcement [Dishman, Sallis, and Orenstein, 1985].

In this chapter, our main focus will be on dropout prediction; the aim of dropout prediction is to identify (potential) dropouts before they actually drop out of the program. By knowing which participants will drop out in advance, additional attention or interventions can be directed to those participants at risk. What interventions are appropriate depends on the program, but in some cases a simple phone call or email can suffice. To achieve dropout prediction, we make use of the we have data gathered about the participant so far, for instance regarding their participation or progress in the program, self-reported information such as their motivation levels, or demographic information such as age or BMI.

The prediction of participants at risk of dropout has been attempted in a number of studies, although not necessarily in the context of a lifestyle physical activity program. Keijsers, Kampman, and Hoogduin [2001] attempt to predict dropout in a cognitive behavior therapy program for people with panic disorder, using a logistic regression approach. Two predictors (motivation and level of education) were statistically significant, but the authors mention that effect sizes are small, leading to only moderately successful prediction. They also observe that there may be many individual reasons for dropout, rather than finding the existence of some uniform 'dropout group'.

Another popular topic for dropout prediction is the dropout of students in (online) courses, with the aim to pre-emptively intervene for at-risk students. An example is the work of Lykourentzou et al. [2009], where three different machine learning techniques are used to predict students at risk of dropout (neural networks, support vector machines, and probabilistic simplified fuzzy ARTMAP). The authors also investigate ensemble learning techniques for the three algorithms, achieving higher predictive accuracy compared to the individual algorithms. Similarly, Kotsiantis, Pierrakeas, and Pintelas [2003] discuss the prediction of student dropout in distance learning courses by exploring a set of six different algorithms, of which the authors report the Naive Bayes algorithm as being the most appropriate. Results also indicate that dropout prediction becomes more accurate towards the middle of the academic period, compared to the initial predictions using only demographic data.

In terms of dropout prediction in exercise programs, Seelig and Fuchs [2011] discuss dropout of new members of a health-oriented fitness center. The authors propose a rule-based prediction scheme, based on the observation that participation behavior might be better modeled as a categorical variable, meaning that participants are first identified as early dropouts, late dropouts, maintenance, or fluctuation. It should be noted that the authors are looking at a longer time scale compared to a twelve-week program (in the twelve-week program, all dropouts would be considered 'early dropouts').

It should be noted that the majority of work on dropout prediction in exercise programs focuses on scheduled exercise, that is, exercises planned at certain times and days of the week. In contrast, lifestyle physical activity programs aim at increasing overall levels of physical activity by being more active throughout the day. It can therefore be questioned how well the findings from exercise programs generalize to lifestyle physical activity programs. Still, we can conclude that there has been some success in dropout prediction in a number of different settings, and that we have to consider a potentially large number of reasons as to why participant may drop out (as such, it may be required to include a sufficient amount of information and participants in the model to account for this level of variance).

The organization of this chapter is as follows: in Section 4.2, the data set used in this chapter is described, and a number of its properties with regards to dropout prediction are explored. The topic of dropout prediction is further explored in Section 4.3, where a genetic algorithm to aid dropout prediction is introduced, and then evaluated and compared to the method use by the DirectLife coaches in Section 4.4. Finally, conclusions are discussed in Section 4.5.

4.2 Data analysis

The work described in this chapter makes use of a de-identified[2] version of New Wellness Solutions' DirectLife database, containing the physical activity data and characteristics of a large number of participants in the DirectLife program. The physical activity data is recorded through a wearable activity monitor, which uses an accelerometer to estimate the amount of physical activity of its wearer. The activity monitor can be worn at a variety of positions, including the belt, pockets or around the neck. To read out the data, the activity monitor must be connected to the USB port of the participant's computer, called 'docking', at which point the physical activity data is automatically uploaded to the DirectLife database. Docking also allows the battery of the activity monitor to be recharged. Participants can then view their newly recorded data (and previously recorded data) through a web interface. In addition to the physical activity data, the DirectLife database also contains web interface usage

[2]That is, any personal or possibly identifying data has been removed, such as names, addresses, dates of birth, and so on.

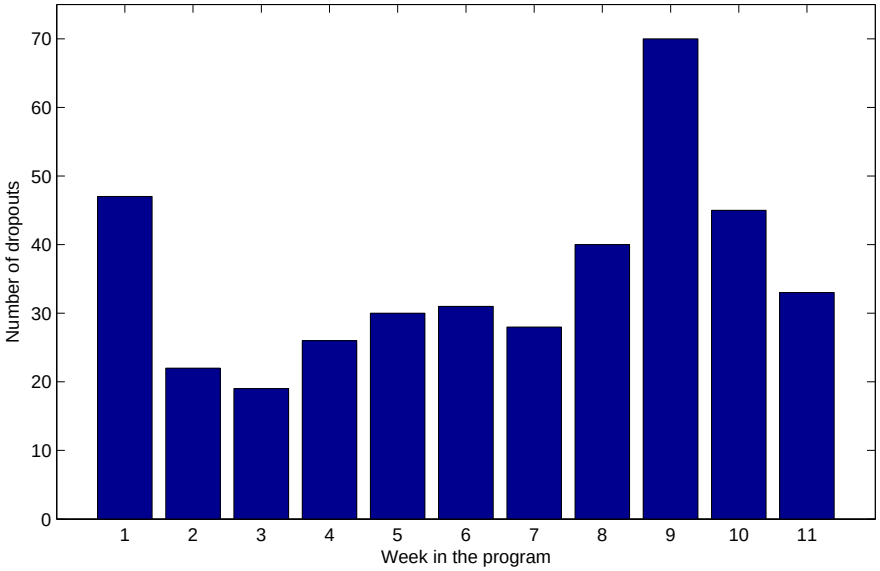


Figure 4.1: The number of dropouts by week in the DirectLife program, based on the dropout day ($d_l + 1$).

statistics, and personal characteristics (such as age, gender, BMI) and activity goals determined by the participant as part of the initial assessment week.

Not all participants present in the database are suitable for analysis, for various reasons. For example, some participated in trials with different kinds of programs, and others signed up, but never actually used the service. To filter out such cases, only participants which match the following criteria are selected for analysis:

- Participants must have participated in a twelve-week program.
- Participants must have completed an assessment week prior to the twelve-week program.
- Participants from DirectLife trials (for purposes of testing, etc.) are not considered.
- Participants must have a profile (containing e.g., age, weight, etc.) available.
- Participants must have at least some participant data available.
- Participants must have logged in to the web interface and docked the device at least once.

Once all the above criteria have been taken into account, a total of 950 selected participants remain in the database. Of the 950 participants, 391 (41%) did not complete their twelve-week program, and are thus labeled as dropouts. To determine

which participants were dropouts, the definition for dropout by New Wellness Solutions was used: a participant is considered a dropout if no activity is recorded within the last two weeks (14 days) of the program.

Definition 4.1 (Dropout). A participant j of a program is considered a dropout if $d_e - d_l(j) \geq 14$, where d_e is the final day of the program, and $d_l(j)$ is the last day on which activity was recorded by participant j. Days are numbered consecutively from the start of the program, with the first day of the program $d_s = 1$. Note that $d_l(j)$ may be after the program ends (i.e., $d_l(j) > d_e$). □

Figure 4.1 shows the number of dropouts per week in the DirectLife program using this definition. To determine the week in which a participant drops out, the dropout day is used; this is the day after the last time a participant recorded activity data (i.e., $d_l + 1$). Note that to still fit the definition of a dropout, the dropout day cannot be later than the first day of week eleven in the twelve-week program. This means that the dropouts under week eleven are all from a single day, rather than a full week.

Given this fact, it is noteworthy that the number of dropouts in week eleven is still comparable to the other weeks. Some insight as to why this is can be obtained from Figure 4.2. Here, it can be seen that there is a strong weekly pattern regarding a spike in the number of dropouts. While the participants in the program start at different dates, the first day of a program is nearly always a Monday. From the figure, it appears that most dropouts occur after the weekend. One possible explanation of this is that participants become occupied with the busier weekday routine, and lose focus on improving their physical activity. Another explanation is that most participants dock their activity monitors during the weekend; as such, they may still be recording activity on Monday, but the data never gets entered into the system as the device is not docked again.

As mentioned, the main focus of this section will be dropout prediction. Basically, the aim of dropout prediction in the context of this Chapter is to predict whether or not a participant will become a dropout in the next week of the program. Here, the weeks refer to the twelve weeks in the program, rather than an exactly seven-day period between the moment of prediction and the dropout day. Specifically, dropout prediction refers to the following problem.

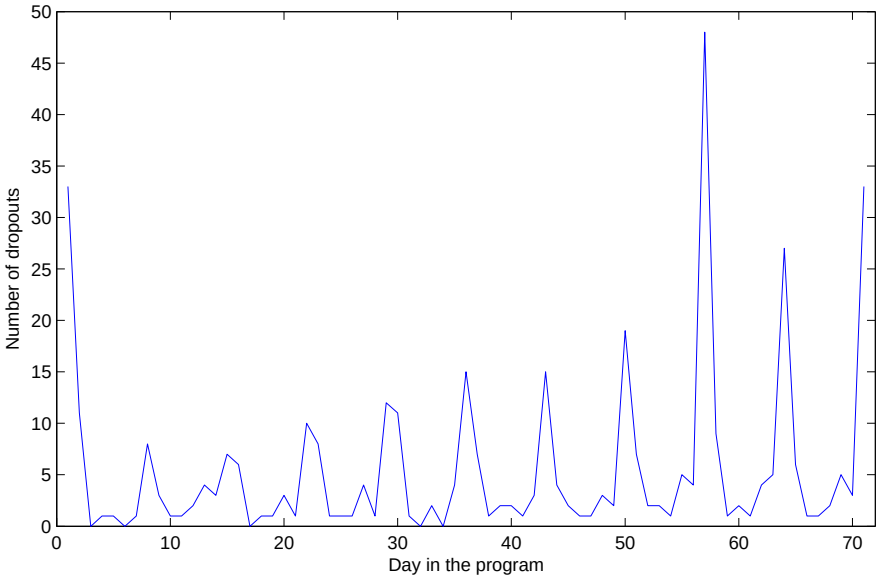


Figure 4.2: The number of dropouts per day in the program in the DirectLife program, based on the dropout day ($d_l + 1$). Note that week twelve is not included in the figure, as a participant's dropout day cannot be in week twelve, per definition.

Definition 4.2 (Dropout prediction). Let $d_l(j)$ be the last day on which activity was recorded, counted from the first day of the program, by a participant j. Then, if participant j is defined as a dropout as per Definition 4.1, let the dropout week $w_l(j)$ be the program week containing the first day after $d_l(j)$, that is, $d_l(j)+1$ (with the first week of the program being days $1-7$, and so on). For each week y, the aim is to classify for each participant j whether or not $w_l(j) = y$, using data up to and including week $y-1$. □

By parsing the data present in the database, a number of features for each participant are retrieved, which are referred to as markers. Markers generally fall into one of three categories. First, static markers represent participant properties which remain more or less static over time, such as a participant's height or gender. Second, assessment markers relate to participant behavior recorded during the assessment period of the program, and include markers such as the average activity recorded during this time. Third, dynamic markers represent (aggregates of) participant behavior during a specific time frame within the twelve-week program. Dynamic markers typically change over time and therefore depend on the time frame under consideration, for example the average activity in the first week versus the average activity in the current week of the program.

In the remainder of this section, two approaches are considered for data exploration. First, the metric of information gain is used to investigate the expected predictive value of the markers calculated from the database. The principles of information gain are briefly described below. Second, the dropout and non-dropout populations are analyzed statistically for markers which indicate significant differences between both populations. The use of markers for the purposes of dropout prediction is then discussed in Section 4.3.

Information gain. Information gain, a notion common in the field of decision trees, is an indication of the increase in purity obtained by splitting a set given some separator or decision boundary. Given a single marker or a set of markers, a hypersurface decision boundary can be defined, splitting the set of participants into two separate subsets. Ideally, for some subset of markers a decision boundary exists which intersects the set of participants such that one of the resulting subsets contains all dropouts, and the other subset contains all non-dropouts. The resulting subsets are then considered 'pure'. In practice, such a decision boundary may unfortunately not exist. However, the purity of a decision boundary's resulting subsets serves as an indication of the predictive power of the corresponding subset of markers over which the decision boundary is defined.

Given a subset of participants, the impurity of the subset S is given by the entropy impurity measure i, which is defined as follows.

Definition 4.3 (Impurity). The entropy impurity i of a subset S is given as

$$i(S) = -\sum_j P(c_j) \log_2 P(c_j)$$

where $P(c_j)$ is the fraction of participants in S belonging to class c_j. □

The entropy impurity is maximal when each class is equally represented (that is, $P(c_i) = P(c_j)$ for all i, j), and minimal (zero) when all participants belong to the same class. From decision trees, it is known that the reduction of impurity for a given decision boundary over a subset S, often called information gain when entropy impurity is used, is given as follows.

Definition 4.4 (Information gain). When splitting a set S into two subsets S_1 and S_2, the information gain $\Delta i(S)$ is given by

$$\Delta i(S) = i(S) - P(S_1) i(S_1) - (1 - P(S_1)) i(S_2)$$

where $P(S_1)$ the fraction of participants in S present in S_1. □

A more detailed description of impurity and information gain can be found, for example, in the book by Duda, Hart, and Stork [2000], within the context of decision trees.

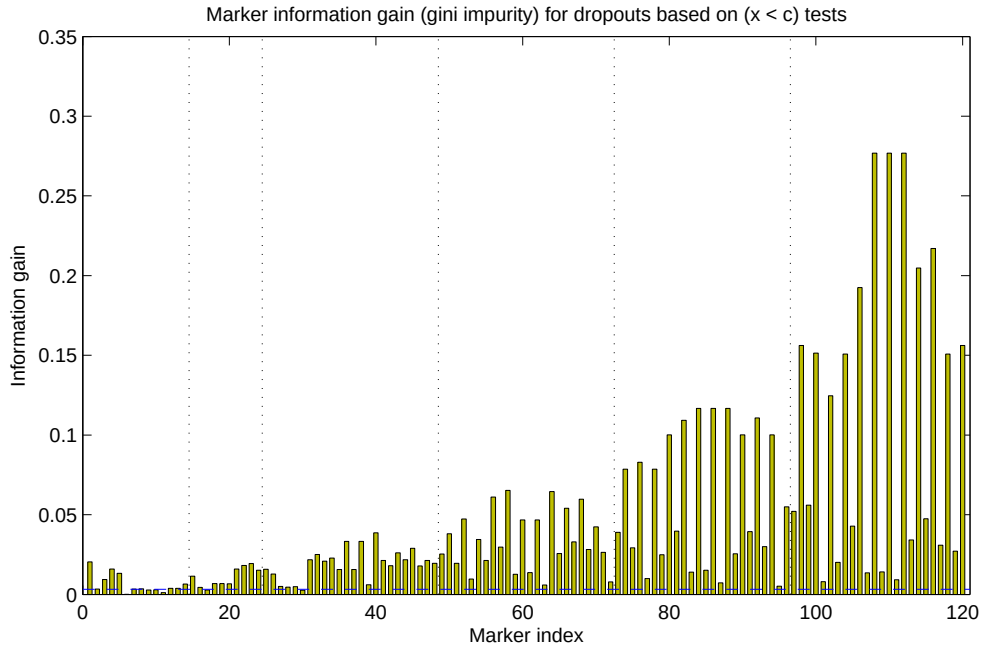


Figure 4.3: The information gain scores of the individual markers, divided into six segments indicated by the vertical, dotted lines. The first segment represents the static markers, the second segment represents the assessment markers, and the final four segments represent the same set of dynamic markers for different time slots. The third segment includes data up till and including the first quarter of the program, the fourth segment includes data up to and including the first half of the program, and so on. The horizontal, dashed, blue line indicates the expected information gain for completely random marker values.

4.2.1 Information gain analysis results

As a first step, the information gain for each of the individual markers was calculated. The aim was to assess the predictive power of the available markers, which might serve as a basis for selecting which markers are relevant with regard to dropout prediction. Let $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ be the set of all available markers, containing n markers in total, and let x_{ij} denote the value of marker $\mathcal{X}_i$ for participant j. For a marker $\mathcal{X}_i$, the optimal decision boundary C_i was computed, where C_i is a constant that maximizes $\Delta i(S)$ by subsetting $\mathcal{X}_i$ as $S_1 = \{x_{ij} | x_{ij} < C_i\}$, and $S_2 = \{x_{ij} | x_{ij} \geq C_i\}$. Computing C_i is fairly straightforward; for a total of h participants, there are at most $h-1$ possible values for each C_i to consider. Given the number of participants (950) and markers (120), these can easily be computed exhaustively. The results are shown in Figure 4.3.

Marker	*Dropout mean*	*Non-dropout mean*	*Test statistic*
High minutes (pa)	**6.21** per day	**9.30** per day	5.72
High minutes (p)	**6.72** per day	**9.30** per day	5.67
Docking (p)	**0.23** times per day	**0.56** times per day	12.15
Docking (a)	**0.65** times per day	**0.9** times per day	4.40
Reported motivation	n/a	n/a	167.1 (df=3)
Docking (pa)	**0.26** times per day	**0.58** times per day	11.58
Moderate minutes (p)	**26.38** per day	**35.89** per day	6.59
Moderate minutes (pa)	**25.73** per day	**35.5** per day	6.49
Coaching method	n/a	n/a	406.7 (df=2)
Moderate minutes (a)	**27.65** per day	**34.24** per day	4.18
Reported difficulty	n/a	n/a	211.6 (df=4)
LEDs observed (p)	**4.15** times per day	**8.46** times per day	15.28
LEDs observed (pa)	**4.14** times per day	**8.13** times per day	14.51

Table 4.1: Markers with statistically significant differences ($p < 0.0001$) between dropouts and non-dropouts. Between parentheses is indicated whether the marker is computed over data in the assessment period (a), the program period (p), or both (pa). Averages for both groups are provided for markers with non-categorical values. The test statistic refers either the χ^2 goodness of fit test statistic (for ordinal values), or the t-test test statistic (for continuous values), depending on the test used.

From these results, as well as from visual inspection of marker histograms, it can be concluded that markers individually possess little predictive power with regard to dropout prediction. All of the highest scoring markers are found in the dynamic markers segments, as can be seen in Figure 4.3. In particular, it can be seen that every second marker in these segments become increasingly more informative with time. This is partly due to the ordering of the markers in the figure: for each quarter, most dynamic markers can be paired with another, similar marker which measures the same feature. One of these markers measures this feature for only the given quarter, while the other measures the same feature for all quarters up to and including that quarter. As more data becomes available, the fact that participants have dropped out becomes more visible in the data (for instance, they no longer record activity), and as such it can be expected that discrimination between dropouts and non-dropouts increases. It is therefore questionable whether these markers would be as successful when all dropouts to be predicted occur only in the future.

4.2.2 Statistical analysis

An alternative approach to finding potentially predictive markers is the use of statistical analysis to find significant differences between dropouts and non-dropouts as groups. The test used to determine this depends on the type of marker. For nominal

or ordinal markers (that is, markers with discrete values such as gender) Pearson's χ^2 test is used to determine whether samples drawn from the dropout population match the distribution of samples drawn from the non-dropout population. For markers with continuous values (for example, the average active minutes per day) the Student's t-test is used to determine if samples from the dropout population share the same mean as samples from the non-dropout population. The t-test was selected as visual inspection of the data suggests that the assumption of normally distributed samples seems reasonable. In addition, for all tests Bonferroni correction was applied to adjust for the large number of statistical tests used.

Table 4.1 shows a list of the most relevant (in terms of the relative difference in means) statistically significant markers (with $p < 0.0001$). The reported means do not include days in which there is no recorded activity (i.e. the activity monitor is not used). The complete list of significant markers contains 34 items. An interesting conclusion is that values measured during the assessment period are almost as indicative as values measured during the remainder of the program. Unfortunately, although these markers show statistical significance with regard to the differences between dropouts and non-dropouts, they do not provide sufficient predictive power by themselves to accurately distinguish individuals from both groups, as is also suggested by the results in Section 4.2.1. Here, the high number of significant markers is in part a consequence of the large number of program participants; while the means of both populations differ significantly, the variance generally seems sufficiently high that defining a clear decision boundary becomes difficult.

The results show that non-dropouts overall have a considerably higher number of moderate and high intensity activity minutes per day than eventual dropouts. Days where no activity was recorded were filtered out for this analysis, which might otherwise explain this effect, as dropouts can be expected to have a greater number of inactive days for obvious reasons. This seems to indicate that non-dropouts tend to be more active than non-dropouts. Indeed, a significant difference in PAL score (the activity levels measured by the activity monitor) was found, although this is not yet visible during the assessment week.

The results also show that on average, non-dropouts dock their activity monitor to upload data more often, and more often trigger the activity monitor for feedback[3]. This seems to indicate that non-dropouts are more engaged with the program's website and the activity monitor throughout the program. There are also significant differences on the self-reported motivation and perceived difficulty (established through questionnaires at the start of the program). Interestingly, dropouts report a somewhat higher motivation than non-dropouts, but also report a higher perceived difficulty of completing the program. Although not significant, non-dropouts also report to be slightly less active than non-dropouts.

[3]The activity monitor will give a visual indication of the amount of daily activity through a series of LEDs if briefly shaken. In Table 4.1 this is indicated as 'LEDs observed'.

4.3 Dropout prediction

The aim of dropout prediction is to identify dropouts before they actually drop out of the program. As was mentioned in Definition 4.1, dropouts are defined as participants who have not recorded any activity data in the last two weeks of the program. Identifying a dropout at the appropriate moment in time is essential: detecting a dropout after the initial dropout point may mean it is too late to take action, and while detecting a dropout weeks ahead of time might be informative, there may be little need to take action at that time[4]. It is therefore assumed optimal to detect dropouts one week prior to the actual dropout week.

To simulate this need for timing, we consider each week of the program individually for making dropout predictions. As participants head towards the end of their program, an increasing amount of data becomes available for analysis (in the form of dynamic markers). For example, in the second week of a participant's program we are agnostic regarding their activity levels in week four. To take this increase in data into account, dropout predictions are made using only the data available at the time. In other words, when making dropout predictions regarding some week y, we only use data available in week $y-1$. This results in the dropout prediction problem definition as stated in Definition 4.2. It is assumed that for any week y, constant markers and assessment markers will be available.

Initial attempts at dropout classification made use of the C4.5 or CART algorithms (which are based on the information gain metric discussed in Section 4.2) to form decision trees by combining markers. Unfortunately, these yielded poor performance and as such, were not very successful. The trees created were often only a single node deep, indicating that once a decision boundary was formed, the addition of further decision boundaries rarely added any meaningful separation to the created subsets. The best performance in terms of class accuracy of the created decision trees on dropout prediction reached a class accuracy of 56.2%, and many of the combinations of markers examined barely performed above random guessing (which is equivalent to 50% class accuracy). The results of the decision tree approach are shown in more detail in Section 4.4.

As an alternative, a classification scheme based on genetic programming to combine markers was developed. The classification process consists of a number of steps. First, a new set of combined markers is created from the available set of markers through genetic programming, as discussed in Section 4.3.1. The aim is to find combinations of markers which together have more predictive power than the individual markers. Then, the dimensionality of the participant data is reduced to remove

[4]In some cases, it may arguably be beneficial to address potential dropouts further in advance. However, a longer time period between the moment of prediction and the actual dropout point increases the odds that an intervention might be perceived as burdensome rather than helpful, and as a result may lead to adverse effects instead.

any highly correlating data using principal component analysis (described in Section 4.3.2). Finally, participants are classified using the nearest neighbor algorithm, which is briefly discussed in Section 4.3.3. The classification results are compared to the method currently used by the program's coaches, detailed in Section 4.3.4, and the results of the decision tree classification described above.

The choice to use a genetic programming approach for this problem is mainly motivated by the fact that both analysis of information gain and statistical analysis have shown few promising markers in terms of predictive power in our set of markers. As we do not have a clear solution on what markers to include in a prediction algorithm, a method is needed to address the feature selection problem, and to enhance the current markers in terms of predictive power, both of which the genetic programming provides [Yang and Honavar, 1998; Guo, Jack, and Nandi, 2005].

4.3.1 Genetic programming

The aim of the genetic programming algorithm described here is the find combinations of markers that are more predictive compared to the markers individually. To estimate the predictive power of a set of markers, the notion of information gain is used once more. Rather than determining a decision boundary over a single marker, a decision boundary is created over a linear combination of a subset of markers.

Let $\mathcal{X}$ be the set of all n markers, and let X be a sequence of n markers drawn from $\mathcal{X}$ (that is, X is ordered, and may contain the same marker more than once). Finding an optimal decision boundary over X directly is problematic as n increases; the number of potential decision boundaries quickly becomes unmanageable. In order to ensure the optimization problem remains tractable, only decision boundaries of the form $f_X < C$ are considered, where f_X is a linear function over X of n terms, with each marker in X as a single term. Like before, C is the optimal decision boundary constant, and can once again be found as a one-dimensional optimization problem, which can be solved in $O(h)$ for h participants. The challenge is then to find the optimal sequence of markers X, and the corresponding expression f_X.

Finding the globally optimal values for X and f_X would require an exhaustive search over all sequences X and functions f_X, which quickly becomes intractable as n increases. However, the optimal decision boundary can be approximated by genetic programming techniques. In genetic programming, a set of candidates E, each consisting of a set of markers X and function f_X, are maintained.

Definition 4.5 (Candidate). A candidate e is defined as a sequence of markers X of length n, and a function $f_X : X \to \Re$ of n terms, where each marker in X corresponds to a single term in f_X. □

In the context of the dropout prediction genetic algorithm, the marker functions f_X are represented as a set of operators $o_1, \ldots, o_{n-1}$, with $o_i \in \{+, -, *, /\}$. These are combined with the markers in X, yielding expressions of the form $X_1 o_1 X_2 \ldots o_{n-1} X_n$.

These expressions are assumed to be left-associative. Although the expressions themselves are very basic, they already allow for considerable variety in marker combinations.

The set of candidates $E = \{e_1, \ldots, e_m\}$ is updated over a number of rounds in an iterative process, consisting of the following, recurring steps:

- fitness: each of the candidates is assigned a score, or 'fitness' of the candidate.
- selection: based on the fitness of each candidate, a number of candidates are eliminated from the candidate pool.
- reproduction: the candidates are altered and re-arranged to form a new set of candidates through mutation, cross-over, insertion and deletion.

The purpose behind these three steps is loosely based on the concept of 'survival of the fittest' from evolution theory[5]. Fitness represents how well the candidates are adapted to the environment; in our case, the expected ability to distinguish between dropouts and non-dropouts. The candidates essentially represent the genetic code to create a separator between these two classes. Selection determines which candidates 'survive' to create offspring and pass on their genetic code. As expected, the more 'fit' candidates have a better chance of making it through the selection. Reproduction, as the name suggests, involves the creation of new candidates from the genetic code of the surviving parents, which is altered through cross-over, mutation, insertion and deletion.

The four reproduction mechanisms all seek to create new genetic codes based on the parent codes, but achieve this in different ways. The aim is to use all the mechanisms together, to create sufficient variation from the original parent code. Mutation, like its real-life counterpart, creates small changes by randomly altering some values. Cross-over creates a new code by combining random parts of two parent codes. Insertion and deletion do not really fit to a real-life analogy, as our genetic code generally does not vary in length, but essentially add new sections or remove sections of the genetic code. The four mechanisms will be discussed in more detail below.

Essentially, repeating the steps of fitness, selection and reproduction, and going through the various 'generations' amounts to a semi-guided search of the marker space to find an effective combination of markers to distinguish between dropouts and non-dropouts. In other words, the genetic programming procedure aims to incrementally improve the pool of candidates by means of the mutation and cross-over mechanics, and selecting the best results each round. Each of the three steps will be discussed in more detail below. For a more in-depth introduction to genetic programming, see for example the book by Koza [2010].

The genetic algorithm terminates once a certain number of rounds have been completed, or alternatively, once a target overall fitness has been reached. Features

[5] In the same way that neural networks are related to the neurons in the brain; that is, the comparison does not really extend beyond the superficial level.

can then be selected from the pool of remaining candidates. There are several options to select the final features; either the entire surviving candidate pool can be used, or a number of high-fitness candidates can be selected. Other alternatives include selecting high-fitness candidates from previous rounds as well.

For the results described here, the entire candidate pool was used, after which the total dimensionality of the feature set was reduced using principle component analysis (PCA). The motivation for the use of the PCA step is that often, a genetic programming population will contain a number of candidates that are highly similar to each other. Using PCA, we will be able to group these similar candidates together in a fashion, reducing the overall dimensionality of our candidate space, and reducing the risk of overfitting as a result.

Fitness. Given a set of participants, the fitness of a candidate expression can be determined by the information gain of that expression, as defined in Definition 4.4. In practice, the set of participants used for this purpose is a subset of all available participants, as it is advisable to maintain a separate train and test set to avoid contamination of the results due to possible overfitting. As decision boundaries take the form of $f_X < C$, the optimal value for C must be found to determine the fitness for each candidate. As before, computing $f_X(x_j)$ for a total of h participants, where x_j are the values of the markers in X for participant j, results in at most h unique outcomes. As a result, there are at most $h-1$ values for C to consider.

In general, shorter expressions are preferred over equally informative longer expressions[6]. In this case, the longer expression is needlessly complicated, and a simpler expression would be easier to interpret. More importantly, even if they are somewhat more informative, longer expressions are more likely to result in overfitting, and as such may lack generalization ability. For these reasons, a penalty to the fitness score is introduced, which is deducted from an expression's fitness for each operator in the expression beyond a fixed number (three for the results in Section 4.4). As a result, introducing additional operators to an expression will only result in a higher fitness if the increase in information gain due to these operators exceeds the penalties incurred.

Selection. To allow for the creation of new candidate expressions, a number of candidates are removed each round (see Section 4.4). Generally, it is preferable to retain the candidates with the highest fitness scores. However, it is often beneficial to also retain a number of candidates with lower fitness scores, to retain the ability to move away from the current local optimum and transverse the feature space for other optima. This feature, in part, differentiates genetic algorithms from some local search algorithms such as gradient descent. To accomplish this, candidates to be retained are generally selected by chance, where candidates with higher fitness are more likely to be retained than candidates with a lower fitness.

[6]This can be seen as a version of Occam's razor.

Numerous selection algorithms exist based around this principle, of which tournament selection is one of the more popular. In tournament selection, two candidates are selected at random, of which the candidate with the highest fitness is retained, and the other candidate is discarded. This process is repeated by randomly selecting two more of the candidates that have not been selected previously. Once there are no candidates left for selection, the selection process starts anew for those candidates that have been retained, until the desired number of discarded candidates has been reached (this may happen before a tournament round is complete, in which case any remaining candidates are automatically retained). The process of tournament selection is such that candidates with higher fitness are preferred. Due to randomly selecting sets of candidates however, candidates with lower fitness also have a chance to be retained.

Reproduction. After selection, a new set of candidate expressions is created using the selected candidates as a basis. In genetic algorithm, two main types of mechanisms govern this reproduction process. First, mutation of an existing candidate involves small random changes to the characteristics of the original candidate. Second, crossover involves combining attributes of two existing candidates to create a new candidate. The two mechanics are described in more detail below. In addition, the insertion and deletion mechanics are also discussed, which allow for the addition or removal of markers from the candidates.

As mentioned above, let the functions f_X be represented as a set of operators $o_1, \ldots, o_{n-1}$, with $o_i \in \{+, -, *, /\}$. The mutation mechanism can be implemented either through random changes to the sequence of markers X, or to one of the operators in f_X. The probability of such a change occurring for a specific marker variable or operator is given by parameter called the 'mutation rate'. The mutation of a specific marker or operator in $f(X)$ can be defined as follows:

Definition 4.6 (Mutation). For a sequence of markers X of length n, drawn from the set of all markers $\mathcal{X}$, and a function f_X represented by a sequence of consecutive operators $o_1, \ldots, o_{n-1}$, yielding an expression of the form $X_1 o_1 X_2 \ldots o_{n-1} X_n$, the mutation of a marker X_k consists of a transformation of the form

$$X_1 o_1 X_2 \ldots X_k \ldots o_{n-1} X_n \rightarrow X_1 o_1 X_2 \ldots \hat{X}_k \ldots o_{n-1} X_n$$

where $\hat{X}_k \in \mathcal{X} \setminus \{X_k\}$. Similarly, the mutation of an operator in f_X is a transformation of the form

$$X_1 o_1 X_2 \ldots o_k \ldots o_{n-1} X_n \rightarrow X_1 o_1 X_2 \ldots \hat{o}_k \ldots o_{n-1} X_n$$

where $\hat{o}_k \in \{+, -, *, /\} \setminus \{o_k\}$. □

Alternatively, a new candidate expression can be created through cross-over. Here, two candidate expressions are selected at random, and each expression is split at a random point along its length. A new candidate expression is created by either

attaching the front of the first expression to the tail of the second expression, or vice versa. Naturally, the resulting candidate expression must be a valid expression itself. This can be achieved by always splitting expressions after (or before) an operator. The impact of the cross-over mechanism on reproduction is governed by the cross-over rate parameter, which determines the fraction of new candidates created through the cross-over mechanism. The definition of cross-over is given as:

Definition 4.7 (Cross-over). For two sequence of markers X of length n and $\hat{X}$ of length $\hat{n}$, drawn from the set of all markers $\mathcal{X}$, and corresponding functions f_X and $f_{\hat{X}}$ represented by sequences of consecutive operators $o_1,\ldots,o_{n-1}$ and $\hat{o}_1,\ldots,\hat{o}_{\hat{n}-1}$ respectively, yielding expressions of the form $X_1o_1X_2\ldots o_{n-1}X_n$ and $\hat{X}_1\hat{o}_1\hat{X}_2\ldots\hat{o}_{\hat{n}-1}\hat{X}_{\hat{n}}$, cross-over at markers k,l can be defined as

$$X_1o_1X_2\ldots X_ko_k\hat{X}_l\hat{o}_l\ldots\hat{o}_{\hat{n}-1}\hat{X}_{\hat{n}}$$

where $1 \leq k < n$ and $1 < l \leq \hat{n}$. □

Often, genetic algorithms operate on candidates with some fixed length representation. However, candidates can be altered in length by removing or introducing new markers and operators. The insertion and deletion operators can be used for this purpose. The insertion mechanism adds a new marker and operator pair to the expression at a random position. Similarly, the deletion mechanism removes a random operator and marker pair from an expression. The probability of insertion or deletion occurring in a new candidate is governed by the insertion rate and deletion rate, respectively. First, insertion is defined as:

Definition 4.8 (Insertion). For a sequence of markers X of length n, drawn from the set of all markers $\mathcal{X}$, and a function f_X represented by a sequence of consecutive operators $o_1,\ldots,o_{n-1}$, yielding an expression of the form $X_1o_1X_2\ldots o_{n-1}X_n$, insertion is a transformation of the form

$$X_1o_1X_2\ldots X_ko_kX_{k+1}\ldots o_{n-1}X_n \rightarrow X_1o_1X_2\ldots X_ko_k\hat{X}\hat{o}X_{k+1}\ldots o_{n-1}X_n$$

where $\hat{X} \in \mathcal{X}$ and $\hat{o} \in \{+,-,*,/\}$. □

Similarly, deletion is defined as

Definition 4.9 (Deletion). For a sequence of markers X of length n, drawn from the set of all markers $\mathcal{X}$, and a function f_X represented by a sequence of consecutive operators $o_1,\ldots,o_{n-1}$, yielding an expression of the form $X_1o_1X_2\ldots o_{n-1}X_n$, deletion is a transformation of the form

$$X_1\ldots o_{k-1}X_ko_kX_{k+1}\ldots o_{n-1}X_n \rightarrow X_1\ldots o_{k-1}X_{k+1}\ldots o_{n-1}X_n.$$

□

4.3.2 Principal component analysis

Using principal component analysis (PCA), a data set can be transformed from a high dimensional space to a lower dimensional space, in which the principal dimensions explain the largest amount of the variance in the data. This means that the higher dimensions are likely highly correlated, and can therefore be discarded without much loss of information. The advantage of this is that data with fewer dimensions is easier (and faster) to perform calculations on, and that the original correlated elements in the data might act as a source of noise to any classification algorithm.

To use principal component analysis, each participant is considered an m-dimensional data point, where m is the number of combined markers selected (which in turn may consist of a number of the original markers). After applying PCA, these dimensions are reorganized, and those explaining little or no variance are removed. For this work, the number of dimensions retained was selected such that at least 95% of the variance in the data would be explained, under the assumption that maintaining sufficient variance would be beneficial in distinguishing between dropouts and non-dropouts.

After the genetic programming step, PCA is used to reduce the dimensionality of the marker space, as the marker combinations created by the genetic programming algorithm can result in sets of marker combinations which are highly correlated. As such, PCA is used as an intermediate step before classification by the nearest neighbor algorithm described in the next section.

4.3.3 Nearest neighbor algorithm

The nearest neighbor classification algorithm classifies an unknown instance (in this case an unknown participant) according to the spatial distance to the nearest known instance. The well-known variant k-nearest neighbor is used here: instead of looking only at the closest known instance, it examines the k closest known instances, and classifies the unknown instance based on the most common class (dropout or not) in the subset of closest known instances. As outlined in the previous section, participants can be represented as data point in the principal component space, and as such distance measures can be used to determine the distance between two participants.

The optimal value for k was determined experimentally in this study. That is, for each week of the program a number of classifications are made on a separate test and train set with different k, and the optimal value found is finally used to classify the unknown participants. As there are more non-dropouts each week than dropouts, dropouts were assigned greater weight than non-dropouts. In Section 4.4, the nearest neighbor algorithm is used for the classification of dropouts on the marker space created by the genetic programming algorithm, after PCA transformation. Other classification algorithms were also explored, including a naive Bayesian classifier and learning vector quantization (LVQ), but these offered no further improved performance compared to the k-nearest neighbor algorithm.

Program week	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
Number of days	3	3	3	6	6	14	14	14	10	10	5	5

Table 4.2: The number of days without docking a participant must equal or exceed to be labeled as a potential dropout, according to the DirectLife coaches' method. The number of days a participant is allowed to go without docking depends on the week of the program they are in; for example, if a participant is in week six, and has not docked for (at least) the last 14 days, they would be considered as a dropout per this method.

4.3.4 Coaches' method

At the time of the investigation, the DirectLife coaches made use of an inferential classification method derived from their own experiences to signal potential dropouts amongst their assigned participants. This method is reasonably straightforward, by counting the number of days since a participant has last docked their activity monitor and uploaded their data. Once this number equals or exceeds a certain constant, the participant is automatically listed as a potential dropout, and the coach will examine the participant's data in detail. The amount that must be equaled by the number of days without docking depends on which week of the twelve-week program the participant is currently in, as detailed by Table 4.2. Note that the numbers indicate the total amount of days since the participant has last docked their activity monitor, and not only the number of days in that particular week.

4.4 Results

Ideally, future dropouts are identified as such before actually dropping out of the program, as this would allow for timely interventions to prevent this. As discussed in Section 4.3, the aim is to predict dropouts one week prior to actually dropping out, using data available at that time. First, for all known dropouts amongst the participants, the actual dropout day is determined, and similarly, the dropout week. For classification, dropouts are only labeled as dropouts during their dropout week. After their dropout week, participants are no longer considered as instances for any subsequent weeks. As there are 12 weeks in the program, and dropouts per definition have no recorded activity in the last two weeks, a separate set of dropouts is created for 11 of the program weeks (dropouts in week 11 have the first day of week 11 as their actual dropout day).

As mentioned, when classifying dropouts in week y, only data available up to and including week $y-1$ is considered. For example, when classifying dropouts in week 10, data recorded in weeks 10, 11 and 12 is not considered. In effect, this creates

an expanding window of data, growing in size with each week classified. For each week, a separate set of genetic markers, principal component analysis and k-nearest neighbor classifier is constructed. 10-fold cross validation was used each week to obtain separate train and test sets. In 10-fold cross validation, data is separated into ten subsets, and iteratively one is selected as test set while the remaining subsets form the training set. Classifiers are trained and evaluated for each test and train set individually, and results are averaged to obtain a classification score for each week.

For the genetic algorithm, a total population of 200 candidates were used, of which 40 were maintained each round as part of selection. For reproduction, the following probabilities were used: a mutation rate of 0.05, a cross-over rate of 0.9, and an identical insertion and deletion rate of 0.3. In total, 100 rounds were completed each iteration.

As the set of dropouts is divided over 11 weeks (and the non-dropouts remain present throughout the program), the number of non-dropouts vastly exceeds the number of dropouts for any given week, creating a class imbalance. Using the standard measure of accuracy for classifier results (number of correct classifications divided by the total number of classifications) in such a situation leads to misrepresentation of classifier performance, as the accuracy on the non-dropout class dominates the classifier accuracy. Therefore, as discussed in Chapter 3, the class accuracy measure is used. Here, the accuracies over the individual classes (dropouts and non-dropouts) are computed first, and the class accuracy is defined as the mean over the accuracies of the individual classes.

To determine the improvement gained from the use of genetic programming, classification results from markers obtained from genetic programming are compared to classification results from markers obtained directly from the participant database. As described above, dropouts are determined for each week separately, using data available at that time. In both conditions, principal component analysis is used to reduce the dimensionality of the participant data, after which the participants are classified using a k-nearest neighbor classifier. The optimal value for the parameter k is established experimentally for each week separately. Due to the sensitivity of the k-nearest neighbor algorithm to class imbalance, discussed in the previous paragraph, before classification dropout samples in the training set are oversampled (i.e., randomly duplicated) until the training set becomes balanced.

In addition, we compare the results of the nearest neighbor algorithm, both with and without genetic programming, to the results of the decision tree model described in Section 4.3. The decision trees are constructed using Gini impurity (see Section 4.2) and use pruning to avoid overfitting.

The classification results are shown in Table 4.3, where the classification accuracies of the decision tree model are shown in the left. As mentioned in Section 4.3, the classification results are fairly low overall, for some weeks approaching 50%

Week	*Decision tree*	*Without GP*	*With GP*
Week 1	0.5768	0.6873	0.7524
Week 2	0.6229	0.6629	0.7396
Week 3	0.5093	0.6667	0.7310
Week 4	0.5188	0.6043	0.6484
Week 5	0.5142	0.6443	0.7201
Week 6	0.6407	0.6364	0.7528
Week 7	0.5455	0.6389	0.6711
Week 8	0.5839	0.6586	0.7838
Week 9	0.6830	0.6165	0.8107
Week 10	0.4895	0.6247	0.7334
Week 11	0.4973	0.5679	0.5773
Average	**0.5620**	**0.6371**	**0.7201**

Table 4.3: Class accuracy per week of dropout prediction using markers obtained from participant data for the decision tree method, and for the nearest neighbor algorithm both with and without the use of genetic programming to adapt the feature space. Classification was done using the k-nearest neighbor algorithm, after PCA transformation. The class accuracies obtained without the use of genetic programming are listed under 'Without GP', while the class accuracies obtained using genetic programming are listed under 'With GP'. Class accuracies were obtained using 10-fold cross validation.

class accuracy, which is equal to the performance of guessing the class label through random chance. The average class accuracy over all weeks is 0.5620, that is, approximately 56%.

The results of dropout prediction using basic markers obtained from the available participant data are shown in the middle column of the table. The table shows class accuracy results for every week in which classification is performed. Over the eleven weeks, the recorded class accuracy scores remain fairly consistent, with slightly higher scores towards the start of the program. The final week of the program, however, scores considerably lower in class accuracy. As can be seen in the table, the average class accuracy over all eleven weeks for this method is 0.6371 (i.e., approximately 64%).

The results of dropout prediction using markers obtained through genetic programming are shown on the right hand side of Table 4.3. Over the eleven weeks, results again appear fairly consistent, with the exceptions in week four, seven and most noticeably week eleven. When compared to the results obtained using only basic markers, the use of genetic programming results in an increase of class accuracy

Week	*Coaches' method*	*With GP*
Week 1	n/a	0.7524
Week 2	0.54	0.7396
Week 3	0.61	0.7310
Week 4	0.60	0.6484
Week 5	0.62	0.7201
Week 6	0.62	0.7528
Week 7	0.52	0.6711
Week 8	0.56	0.7838
Week 9	0.55	0.8107
Week 10	0.62	0.7334
Week 11	0.51	0.5773
Average	**0.58**	**0.7168**

Table 4.4: A comparison of class accuracy per week of dropout prediction using markers obtained from participant data with the use of genetic programming and using the coach's method. The class accuracies obtained using the coach's method are listed under 'Coaches' method', while the class accuracies obtained using genetic programming are listed under 'With GP'. Class accuracies were obtained using 10-fold cross validation. The average class accuracy for the genetic programming method excludes the accuracy in week 1, for the benefit of comparison to the coaches' method.

for every week of the program. As can be seen in Table 4.3, the average class accuracy over all weeks of the program when using genetic programming is 0.7201, or 72%.

Using pairwise t-tests, we can conclude that even without the use of genetic programming, the nearest neighbor algorithm performs significantly better compared to the decision tree model ($t(10) = 3.81$, $p = 0.0034$). We can also see that adding the genetic programming approach further improves the classification results significantly compared to the basic nearest neighbor algorithm ($t(10) = 5.37$, $p < 0.001$). Given these results, we might expect the difference between the decision tree model and the nearest neighbor algorithm with genetic programming to be significant as well, and this is indeed the case ($t(10) = 9.83$, $p \ll 0.001$). Note that the differences remain significant if Bonferroni correction is taken into account for the multiple comparisons between the three classification methods.

The results are also compared to the method used by the DirectLife coaches at the time of the study in order to examine if any improvements can be made. Table 4.4 shows a comparison between the results of dropout classification using genetic pro-

gramming and the results using the coaches' method described in Section 4.3.4. The results of dropout classification using genetic programming is of course identical to those listed in Table 4.3. The average class accuracy of the coaches' method is approximately 58%, which is below the accuracies of dropout prediction using markers either with or without genetic programming. Note that the coaches' method does not allow for classification in the first week of the program, so no accuracy score is listed for this week. Using a paired t-test, the class accuracy scores between the coaches' method and the genetic programming approach are shown to be significantly different ($t(9) = 6.59$, $p < 0.001$).

4.5 Conclusions

From the analysis in Sections 4.2.1 and 4.2.2, it can be seen that while there are on average significant differences between dropouts and non-dropouts for many markers, individual markers offer little predictive value regarding the dropout potential of a participant. This can be explained by the large amount of overlap between the dropout and non-dropout population. If these populations are modeled as a normal distribution for some marker, the means may be some distance apart, but the variances of one or both distributions are sufficiently high to make classification with any certainty difficult. Amongst the current markers, there appears to be no marker for which both distributions are sufficiently separate to be considered a powerful predictor between these two groups.

In Section 4.4, it was shown that classification using basic markers resulted in an average class accuracy of 0.6371, while classification using genetic programming resulted in an average class accuracy of 0.7201. When compared to a majority class (or random guessing) baseline, which in a two class problem yields a class accuracy of 0.5, both of the classification methods discussed here clearly outperform this baseline. When comparing both methods, classification after genetic programming clearly outperforms classification based on basic markers only, as expected, with a significant difference in accuracy rates between both methods. When examining the error rates of both methods (given as error rate $= 1 -$ accuracy), the use of genetic programming results in a reduction in the number of classification errors of approximately 23% ($\frac{(1-0.6371)-(1-0.7201)}{1-0.6371} \cdot 100\%$). When compared to the currently used coaches' method, it is clear that classification using genetic programming provides a higher classification accuracy. The latter method results in a reduction in the number of classification errors of approximately 33% ($\frac{(1-0.58)-(1-0.7201)}{1-0.58} \cdot 100\%$).

When comparing the classification results on a weekly basis, the low class accuracy in week eleven forms an obvious outlier for all three classification methods. The most likely explanation seems to be found in the definition of a dropout; those who drop out in week eleven are close to the decision boundary which separates dropouts from non-dropouts. In essence, week eleven contains dropouts from a single day

rather than a week - 'dropouts' which happen later that week are not labeled as such. As this decision boundary is somewhat arbitrary, it may algorithmically be difficult to distinguish between participants close to the decision boundary, yielding poor classification accuracy scores as a result. Note that for week four, a similar observation can be made with regard to class accuracy for the marker-based methods, although proximity to the decision boundary seems an unlikely cause in this case. As such, the cause is not entirely clear in this case, and the difference may simply be due to variance in the data.

In contrast, other weeks show considerable deviation from the average classification accuracy for the genetic programming method, but no coinciding deviation in accuracy for the method using the coaches' method or basic markers only. Examples of this include week seven and nine. Possible explanations are that either for these weeks the combination of markers adds an amount of information that is lesser or greater than average (for weeks seven and nine, respectively), or that the cause lies in inconsistent performance often inherent in genetic algorithms due to their stochastic nature.

From a practical perspective, the above results show improved classification accuracy can be obtained through the use of genetic programming. However, there are also a number of disadvantages to this method. First, the large number of parameters can present difficulties for those unfamiliar with the algorithm. Fortunately, experiments have shown classification results to be fairly invariant to small parameter changes, and there was no need to construct a different set of parameters for each week of the program. Second, genetic programming algorithms can become computationally expensive, especially as the number of participants and markers in the database increases. Fortunately, classification only requires a single run of the algorithm per week regardless of the number of classifications made, which should ensure the computational costs remain manageable.

A third disadvantage is that genetic programming methods can often be susceptible to overfitting. To counteract this, a penalty function is often required, as we applied in this work. In addition, we used principle component analysis to reduce the dimensionality of our marker set. As the genetic programming algorithm was included in the cross-validation procedure, the results obtained suggest that fortunately, there is little negative effect of overfitting in our application. Finally, the use of possibly complex combinations of markers, rather than single markers, complicates human interpretation of classifier decisions. Indeed, examination of the combined markers generated by genetic programming rarely yields intuitive combinations. This may have consequences with regard to the acceptance and practical usefulness of these classifiers to prevent dropout through interventions in a lifestyle physical activity program.

When comparing our results to the state of the art, in should be noted that several variants of the decision tree classifier we used initially have gained increased popularity with regard to prediction problems, specifically the random forest [Ellis et al., 2014] and gradient boosting algorithms [Lombardo et al., 2015]. In that respect, it would be interesting to explore if these techniques would improve over the rather poor baseline results obtained using the more classical decision tree approach.

Genetic programming-based methods remain popular tools for the predictive modeling of applications with a large number of potential features [Márquez-Vera et al., 2013]. In terms of predictive modeling for health-related applications, methods based on (generalized) linear regression have also seen use [op den Buijs et al., 2015; Tran et al., 2014]. In the near future, it is also feasible that deep learning approaches would be applied to these problems, as these methods provide a means for automatic feature selection. As such, it would be interesting to investigate these methods in the context of the problem and data set described in this chapter.

In conclusion, the work here shows that the addition of genetic programming to dropout classification yields a considerable increase in classifier accuracy. It also shows that combining markers, even using only simple mathematical operators, can yield measures which are more informative with regard to a participant's future behavior than measures using these markers individually. The use of genetic programming is particularly useful when the number of basic markers is very large, making exhaustive approaches to combining markers infeasible, and when there are no clear intuitions regarding which or how markers should be combined. It was shown that when applied to prediction of participants' continued participation in a lifestyle activity program, the use of genetic programming to create combined markers resulted in a reduction in the number of classifier errors of 23% compared to using basic markers, and 33% compared to the currently used coaches' method.

As a more accurate prediction opens up the possibility of intervention for participants who are at risk of dropping out of the activity program, the number of participants completing the program is likely to increase as a result. The results presented in this document emphasize that through combining markers considerable improvements in the prediction of dropouts can be attained, a task that can be considered at least challenging for humans. Therefore, the use of data mining techniques to predict participants dropping out of an activity program ahead of time seems a promising step toward the effective prevention of dropout through timely interventions.

5

Step detection

5.1 Introduction

The work described in this chapter is based on a publication submitted to Gait & Posture [Pijl and Smits, 2016], and related to a number of publications [Pijl, 2015; Pijl, Fulton, and Baldus, 2015; Pijl and Baldus, 2016].

Gait can be described as a particular way of moving on foot, or alternatively, as the way locomotion is achieved (by humans) through the use of their legs. While gait may, at first glance, appear to be achieved similarly from person to person, there are subtle differences between individuals that are characteristic enough to have applications in security. For example, Iwama et al. [2012] use gait detection from CCTV video for person-verification in forensics. Examples of commonly explored aspects of gait include a person's walking speed, cadence (steps per minute) and step variance. Other examples include hip sway, angles of the toes or knees, and so on.

Gait can, very generally, be decomposed into a number of phases; most importantly, each leg can be in a swing phase or a stance phase. During the stance phase, the foot is connected to the ground, while during the swing phase, it is moving through the air. While walking, gait alternates between a double support phase where both legs are in contact with the ground (that is, both legs are in stance), and a single support phase where one leg is disconnected from the ground (that is, one leg is in swing phase and the other leg remains in stance phase). Here, the swing phases alternate between the left and right leg. An illustration of the gait phases is shown in Figure 5.1.

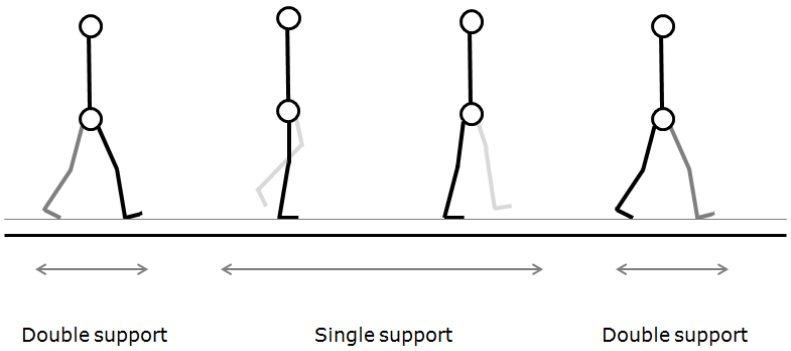


Figure 5.1: Illustration of the gait phases, showing the transition from double support to single support, and back to double support.

A person's gait is closely related to their health. It is well-known that sufficient physical activity is required to maintain a healthy lifestyle [Haskell et al., 2007; Warburton, Nicol, and Bredin, 2006]. For many people, walking is one of the primary contributors to their daily amount of physical activity. As a result, many lifestyle physical activity programs such as the one discussed in Chapter 4 include feedback on the number of steps taken, or the daily distance traversed on foot. Algorithms for step detection and gait analysis are therefore included in numerous commercially available products related to sport, exercise and healthy lifestyle, ranging from pedometers to healthwatches.

Gait analysis also has a role in the medical domain and healthcare; gait is a complex operation, involving amongst others the musculoskeletal system, the processing of sensory information, balance and coordination. Unsurprisingly, if any of the systems involved in gait become compromised, the aspects of a person's gait may change, in many cases sufficiently so that this causes a so-called gait abnormality (some of the better known examples include ataxia, waddling gait or Parkinsonian gait). In many cases, gait abnormalities can be the first warning signs of an underlying condition; Parkinson's disease being arguably the best known example of this [Dijkstra et al., 2008; Ying et al., 2007]. However, it's not just gait abnormalities that can be informative of potential health-related issues; more subtle changes in gait can already be linked to a potential decline in health. For example, a study by Shinkai et al. [2000] with 736 seniors found that walking speed was highly correlated with becoming functionally dependent within a period of six years. Gait is also closely connected to fall risk in elderly.

In addition, decline in gait (i.e., deterioration in one or more aspects of gait) has repeatedly been linked to (the onset of) dementia and cognitive decline [Waite et al., 2005; Deshpande et al., 2009; van Iersel et al., 2004]. It is not entirely clear what exactly is the cause of this link, although there are a number of theories that might

explain the relationship. Several of them are summarized by Scherder et al. [2007]. One explanation can be found in atrophy of the hippocampus, which is known to be functionally involved in memory and complex cognitive tasks. Atrophy of the hippocampus is believed to play a part in memory decline in normal ageing, mild cognitive impairment (MCI), and various types of dementia. It is believed that the hippocampus, among other things, plays a role in spatial orientation and navigation.

Another possible explanation offered by Scherder et al. [2007] is damage to the periventricular white matter, which connects areas of the brain that are a long distance away from each other. It is believed that periventricular white matter lesions are related to a decline in gait and balance. In addition, it was found by Ryberg et al. [2007] that atrophy of the corpus callosum correlated with MMSE score (or mini-mental state examination score, referring to a questionnaire by Folstein, Folstein, and McHugh [1975] widely-used to measure cognitive impairment). In general, the corpus callosum has been reported to play an important role in bilateral motor coordination. It is also possible is that cognitive decline can cause changes in gait through a more direct fashion. For example, a reduced ability by the patient to divide attention effectively can have a negative impact on gait, especially under conditions with a large amount of distractions.

As a result, the automatic assessment of gait parameters such as cadence and walking speed has received considerable attention in research. Accelerometry is an often-used measurement technique for estimating gait parameters as accelerometers are small, cheap and power-efficient, and can be easily worn or attached to objects or clothing. As such, a large variety of techniques and algorithms have been developed for step detection and gait assessment based on input from a tri-axial accelerometer, for various wearing positions, including ankle, wrist and torso positions.

In this chapter, we aim to address the third research question introduced in Chapter 1: Is it possible to determine (psycho)motor skills such as gait accurately using wearable sensors? In doing so, we also address the regression component of the model for human motion tracking described in the main research question, by investigating a number of methods that can be applied to the estimation of cadence, defined as the number of steps a person takes in the span of one minute.

In particular, changes in cadence have often been related to changes in cognitive or physical health [Verghese et al., 2008; Wittwer, Webster, and Menz, 2010; Maquet et al., 2010]. There is a close relation between cadence and step detection; essentially, cadence can be seen as the number of steps detected per minute. Compared to some applications of step detection however, cadence estimation does not necessarily require knowledge of when each step occurred, only that the step happened. Cadence is also strongly related to walking speed and physical activity intensity, and can be indicative of behavioral patterns when studied in a free-living environment [Tudor-Locke and Rowe, 2012].

Specifically, our aim is to investigate the performance of a number of cadence estimation methods using tri-axial acceleration data, for a variety of walking speeds, at both the wrist and torso (pendant) position. As mentioned, preferred and maximum walking speed can often differ considerably between individuals and subpopulations [Bohannon, 1997]. To assess differences and changes in walking characteristics it is important that algorithms perform well across the expected range of walking speeds. In particular, it has often been found that performance can degrade at faster and slower walking speeds [Dijkstra et al., 2008; Zijlstra and Hof, 2003; Storti et al., 2008].

There is a wide body of literature available on methods for cadence and step detection. The literature regarding individual methods is discussed in Section 5.2, whenever appropriate. With regard to more general discussions, a performance comparison for four freely available step detection methods on both healthy participants, as well as mobility impaired geriatric participants, is described by Marschollek et al. [2008]. The authors do not investigate the effect of specific walking speeds on the performance of the selected step detection methods. There, the authors conclude that all four algorithms work poorly overall, and that algorithms customized to a specific sample often perform worse on a sample with different motor characteristics (for example, elderly).

The validity of step detection approaches with a tri-axial accelerometer is investigated by Fortune et al. [2014], using an activity monitoring system and a number of commercially available products, with walking speeds ranging from 0.1 to 4.8 m/s. The authors conclude there is high agreement with manually-recorded step counts for these methods, consistent across most walking speeds except for low speeds (< 0.5 m/s). Ryan et al. [2006] compare the performance of three commercially available pedometers at five different treadmill walking speeds, and three free-walking, self-selected speeds. They conclude that the performance of two of the pedometers decreases at slower walking speeds. The latter two studies mostly focus on commercial devices and do not include any systematic comparisons of multiple algorithms and wearing positions.

Below, the performance of a number of cadence estimation methods using tri-axial acceleration data is investigated, for a variety of walking speeds, at both the wrist and torso (pendant) position. Preferred and maximum walking speed can often differ considerably between individuals and subpopulations. For example, walking speed generally decreases with age [Bohannon, 1997]. It is therefore important to investigate whether the performance of an algorithm varies across different walking speeds. It should be noted that the selected algorithms fit the criteria of being 'easy to use' in practice, in the sense that they do not assume a fixed orientation with regards to the body, or require calibration for individual users to be effective.

In the remainder of this chapter, the cadence estimation algorithms used are first discussed in Section 5.2. The data set used to investigate the performance of these algorithms is then discussed in Section 5.3, followed by the results on algorithm performance on the data set in Section 5.4. Finally, conclusions are provided in Section 5.5.

5.2 Cadence estimation algorithms

Cadence estimation algorithms are generally closely related to step detection algorithms, as cadence is simply defined as the number of steps per minute. As a result, cadence estimation algorithms can often be used for step detection and vice versa, sometimes with some adaptations required. Both cadence estimation and step detection often rely on finding steps from the raw accelerometer signal, which are often visible as peaks in the signal. Therefore, peak detection algorithms are commonly employed.

There is a wide variety of accelerometry-based step detection algorithms described in literature. However, some broad categorizations can be made. First, as walking is a highly periodic signal, a group of algorithms exists which relies on transformations to the frequency domain or other methods to determine periodicity. This includes algorithms based on Fourier transform, autocorrelation or wavelet analysis [Ladetto, 2000]. Second, there is a group of algorithms which relies on filtering or other forms of preprocessing to improve the effectiveness of peak detection algorithms, with examples including threshold-based methods [Mladenov and Mock, 2009; Ryu et al., 2013] and the QRS algorithm [Ying et al., 2007; Marschollek et al., 2008]. Third is a group of algorithms which determines steps through measures of similarity between neighboring segments of the signal, dynamic time warping being arguably the best-known example of this [Li et al., 2012]. Finally, there is a group of algorithms which relies on heuristics for step detection [Huang et al., 2010; Oner et al., 2012]; many rule-based approaches fall into this category.

It should be noted that these categories are not entirely free of overlap, nor are they exhaustive - other examples include the use of finite state machines [Alzantot and Youssef, 2012], or hidden Markov models [Mannini and Sabatini, 2012]. The set of algorithms considered here includes algorithms from each of these groups; these are algorithms based on autocorrelation, dynamic time warping, iterative step detection based on heuristics, and the QRS algorithm. As mentioned in the introduction, an additional requirement was that the algorithms do not assume a fixed accelerometer orientation with regards to the body, or require calibration for individual users. These requirements mean that the algorithms can be more readily used in real-life applications. The algorithms described here can be used for both cadence estimation and step detection, with some modifications in the case of the autocorrelation algorithm.

Walking segmentation. In this chapter, we primarily focus on cadence estimation, and do not discuss the related problem of segmenting measurement sequences recorded during day-to-day activity into segments of walking and segments containing other behaviors. As the data set used in this study does not contain annotated, longitudinal data, we cannot use this data to determine the performance of such a segmentation method. Even so, the use of sequence segmentation to derive walking sequences is an important step in many applications that involve the estimation of cadence or other gait parameters from daily life measurements.

One approach to this is to use a method for k-segmentation as described in Chapter 2, and apply the algorithm to (for example) each day of measurements. As we would generally not have knowledge on the total number of walking segments, the algorithm would need to be repeated for a number of different values for k. For some applications however, it can be sufficient to simply find the k best fitting (according to our error criterion) walking segments, and derive gait estimates from these segments.

In this application of k-segmentation, using the L1 or L2 norm described in Chapter 2 directly on the raw acceleration measurements may not be the optimal approach, as the signal mean or median of a segments may not be the strongest indicator of walking behavior. As the acceleration signals during walking are generally cyclic in nature, and tend to show a high signal intensity, applying the L1 or L2 norm to features such as signal intensity, variability, or regularity may be more effective.

More heuristic based approaches to walking segmentation can also be used; these benefit from being usable in an online fashion, and impose little computational load, often at the cost of being less discriminative. Such methods often take the form of a simple thresholding scheme, for example based on signal intensity as described by Dijkstra et al. [2008], or variance as described by Ladetto [2000].

Outside of threshold-based segmentation, Bradjic and Harle [2013] investigate a number of walking segmentation algorithms, including methods based on short-term Fourier transform, wavelet analysis, hidden Markov models and k-means clustering[1]. A Fourier transform-based method for step segmentation is also used by Zhang et al. [2012] for detecting gait in a middle-aged to elderly population.

5.2.1 Notation

At each time interval, the tri-axial accelerometer simultaneously produces three measurements, one for each axis x, y and z. The accelerometer measurements can be considered a vector $\boldsymbol{a} = (a_x, a_y, a_z)$. The magnitude (or norm) of the signal (see Figure 5.2a) is determined as

$$s = \sqrt{a_x^2 + a_y^2 + a_z^2},$$

[1] As mentioned in Chapter 2, the k-means problem is closely related to the k-segmentation problem, with the exception that the k-means problem does not impose an ordering on the items.

resulting in a sequence $S = (s_1, \ldots, s_n)$ of magnitude measurements. Using the magnitude of the signal rather than the measurements of the individual axes has the advantage that the magnitude is more or less invariant of the orientation of the sensor. In addition, as the gravity component is constant throughout S, it can easily be subtracted.

For a given sequence S, let $X = (x_1, \ldots, x_m)$ be a sequence of indices denoting all local maxima of S, with $x_i \in (1, \ldots, n)$. Here, x_i represents the location of the peak in S; the height of the peak itself is given by s_{x_i}. The peaks in X are assumed to be sorted in increasing order, that is, $x_i < x_j$ for all $i < j$ with $i, j \in (1, \ldots, m)$. A method on how to find such a sequence of peaks X is detailed in Section 5.2.1.

While peak detection can in principle be used to directly determine footsteps from the raw magnitude signal, in practice such an approach often proves to be highly sensitive to noise in the measurements. As such, rather than using peak detection directly as a cadence estimation algorithm, here it is introduced as a component for a number of the cadence estimation algorithms discussed here, where peak detection is preceded and / or followed by a number of pre- or post-processing steps.

Peak detection

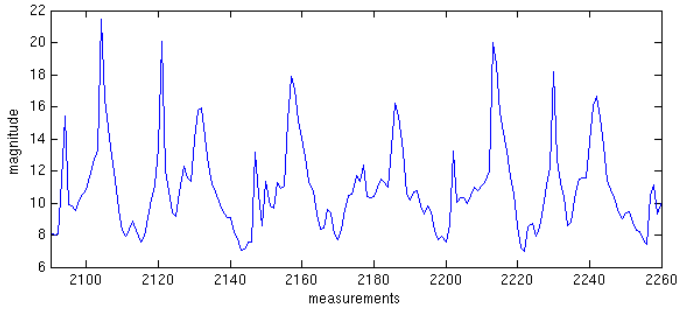
Many step detection algorithms use peak detection as one of their components, as such algorithms rely on the ability to find peaks in a given signal. This section describes the peak detection algorithm used by a number of the cadence estimation methods described in later sections. Here, a fairly straightforward algorithm is used that makes use of a minimum height difference δ between successive peaks and valleys. The algorithm works by iterating over a sequence S, alternatively searching for either a peak or valley, and retroactively assigning a value as a peak or valley once a value is found which exceeds the minimum height difference δ.

At all times, the algorithm maintains the values and location of the current maximum and minimum values, $c_{max} = (S_a, a)$ and $c_{min} = (S_b, b)$, respectively. For each new item i in S, c_{max} and c_{min} are updates as follows:

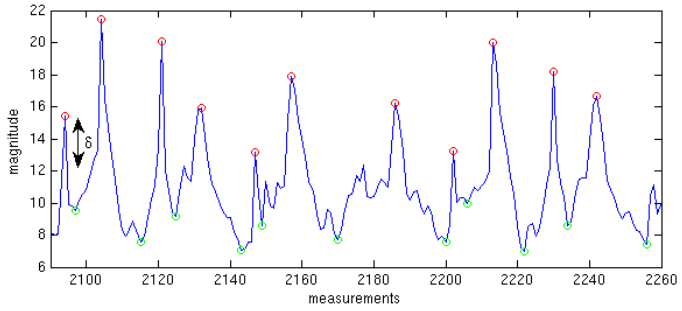
$$c_{max} = \begin{cases} (s_i, i), & \text{if } s_i > s_a \\ (s_a, a), & \text{otherwise} \end{cases}$$

$$c_{min} = \begin{cases} (s_i, i), & \text{if } s_i < s_b \\ (s_b, b), & \text{otherwise} \end{cases}$$

If the algorithm is currently searching for a peak, then a new peak is added to the set of peaks X if $s_i < s_a - \delta$. In this case, a is added to X, $c_{min} = (s_i, i)$, and the algorithm will next look for a valley. Similarly, if the algorithm is searching for a valley and $s_i > s_b + \delta$, the algorithm will set $c_{max} = (s_i, i)$ and will next look for a peak. An example of the peak detection is shown in Figure 5.2b.



(a) Example of the magnitude of a sequence of accelerometer measurements for a participant walking at a speed of 4 km/h.



(b) Example of peak detection results using the peak detection algorithm described in Section 5.2.1, on the walking data shown in Figure 5.2a, using a minimum height difference δ of 3. Detected peaks are shown in red, valleys are shown in green.

Figure 5.2: Example of walking data magnitude and peak detection.

5.2.2 Autocorrelation

Autocorrelation algorithms for step detection make use of the highly periodic nature of walking to determine at which intervals the signal is highly correlated with itself. There are several strategies to implementing a step detection algorithm based on autocorrelation, but most rely on finding the time interval, or lag, at which the autocorrelation reaches a peak. The autocorrelation is defined as

Definition 5.1 (Autocorrelation). The autocorrelation R_{SS} for a sequence S is given by

$$R_{SS}(j) = \sum_{i=0}^{n-j-1} s_{i+j} \cdot s_i,$$

where j is the autocorrelation lag. □

For cadence estimation, it is possible to examine $R_{SS}(j)$ for a number of lags up to a reasonable delay, of for instance 2 seconds. In the implementation used here, the

signal is first smoothed with a moving average filter with a window size equal to one-fifth of the sampling rate, in order to remove some degree of noise from the signal. Then, the autocorrelation is calculated for all successive lags. For the experimental results described in Section 5.4, this is generally equivalent to a delay of 30 seconds. The cadence is then determined based on the average lag between all peaks found by the peak detection algorithm.

As an alternative to autocorrelation, methods based on Fourier transform or wavelet analysis can also be used to find the step frequency. We chose to use the autocorrelation algorithm over these more advanced techniques as in practice, autocorrelation-based methods often seem to be preferred due to their relatively low computational requirements.

5.2.3 QRS

The QRS algorithm described here is based on the well-known Pan-Tompkins algorithm for detecting QRS-complexes in ECG (heart rate) signals described by Pan and Tompkins [1985], and has previously been applied to step detection (for example by Marschollek et al. [2008]). Like the QRS-complexes, the steps in the accelerometer signals tend to be of a periodic nature as well, and as such the principles of QRS detection can also be applied to step detection. The basic premise of the Pan-Tompkins algorithm is a series of four cascading filters which remove noise and locate the periodic peaks. To improve performance on step detection problems, a few adaptations have been made compared to the original algorithm, as will be detailed below.

As mentioned, the QRS algorithm can be seen as a series of four cascading filtering steps. The first step is applying a band-pass filter to the signal. In the original Pan-Tompkins algorithm, this was accomplished by cascading a low-pass and a high-pass filter to obtain a passband of $5-15$ Hz. As the cadence of most (healthy) people tends to be around two steps per second or less, resulting in a step frequency of around 2 Hz, this passband is not optimal for step detection applications. In this implementation, a single band-pass filter was used instead, with a passband of $1-2.5$ Hz, designed using the Parks-McClellan algorithm in Matlab.

In general, the filters used as part of the QRS algorithm are applied through convolution, which is indicated as $S*F$ for a sequence S and a filter F.

Definition 5.2 (Convolution). The convolution $S*F$ is defined as

$$(S*F)(k) = \sum_j s_j f_{k-j}$$

for all indices j for which s_j and f_{k-j} are valid items. □

The second step consists of applying a derivative filter to the signal. The aim of this filter is to derive information regarding the slope of the signal.

Definition 5.3 (Derivative filter). The derivative filtered signal S^{der} is given as $S^{der} = S' * F^{der}$, for the band-pass filtered signal S', and the derivative filter F^{der} given as $F^{der} = \frac{[-1\ -2\ 0\ 2\ 1]}{8}$. □

In the original Pan-Tompkins algorithm, the third step consists of squaring the signal through the filtering step $s_k^{sq} = (s_k^{der})^2$. The aim of this is to identify large changes in the signal, whether in the positive or negative direction. For step detection, this approach turns out to often not be very effective, as it can result in two peaks per step, one during the stance phase and one during the swing phase. Instead, a slightly modified approach to squaring is used in the implementation described here.

Definition 5.4 (Squaring). The squared signal S^{sq} is defined as

$$s_i^{sq} = \begin{cases} q_i^2, & \text{if } q_i > 0 \\ 0, & \text{otherwise} \end{cases}$$

where $q_i = s_i^{der} - \overline{S^{der}}$, and where $\overline{S^{der}}$ represents the (local) mean of S^{der}. □

The fourth and final filtering step consists of applying a simple moving window integration, with the aim of smoothening out the resulting signal.

Definition 5.5 (Moving window integration). The signal after moving window integration S^{mw} is defined as $S^{mw} = S^{sq} * W$, with $W = \frac{[1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]}{10}$. □

Note that the size of the moving window, which is chosen as 10 here, will depend on the sample rate of the accelerometer signal. At a sample rate of 25 Hz, a window of 10 samples will be able to cover most peaks as a result of walking, while being unlikely to merge the peaks of several steps together.

Peak detection is then used to find the steps in the resulting signal. In the original algorithm, a series of adaptive thresholds is used to find QRS-complexes. For step detection however, a single, fixed threshold is used; this is to avoid detecting steps erroneously in the signal during periods where someone is not actually walking. The exact threshold may vary depending on the accelerometer used.

5.2.4 Dynamic time warping

Dynamic time warping (DTW) is a technique used to find the optimal alignment between two time-ordered sequences g and h of length n and m respectively. In a sense, DTW allows for a measure of similarity between two sequences which may have been shifted in time (relative to each other). DTW aims to align the sequences g and h through the construction of a *warping path*.

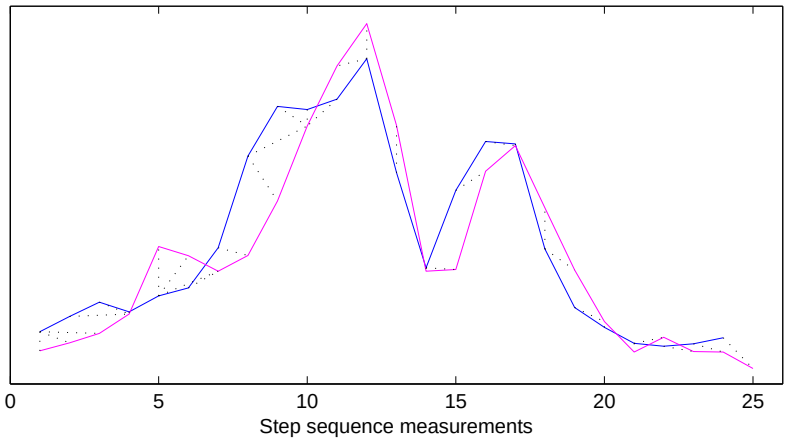


Figure 5.3: Example of the warping path for two step sequences. The dashed lines between the two sequences indicate which points have been matched together by the optimal warping path, which is shown in Figure 5.4.

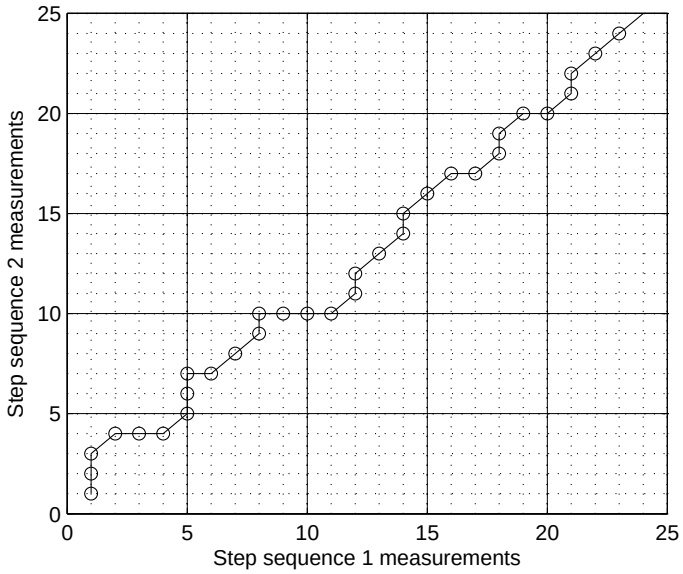


Figure 5.4: Optimal warping path for the sequences shown in Figure 5.3. Here, sequence one corresponds to the sequence marked in blue in Figure 5.3, and sequence two corresponds to the sequence marked magenta. The path on the grid indicates which sequence measurements have been linked together as part of the optimal warping path.

Definition 5.6 (Warping path). A warping path p is a sequence of length l with $p_i(a,b) \in (1,\ldots,n) \times (1,\ldots,m)$. A valid warping path must adhere to the following three conditions:

- *Boundary condition*: $p_1 = (1,1)$ and $p_l = (n,m)$.
- *Monotonicity condition*: $a_1 \leq a_2 \leq \ldots \leq a_l$ and $b_1 \leq b_2 \leq \ldots \leq b_l$.
- *Step size condition*: $p_{i+1} - p_i \in \{(1,0),(0,1),(1,1)\}$ for $i = 1,\ldots,l-1$.

□

Briefly, these conditions state that: (1) the first and last items in g and h must remain the first and last items, respectively, (2) the items in g and h cannot swap their ordering in time, (3) no item can be skipped, and (4) there are no repeated pairs in the warping path p. Here, (1) corresponds to the boundary condition, (2) corresponds to the monotonicity condition, and (3) and (4) correspond to the step size condition.

Definition 5.7 (Optimal warping path). The cost $c_p(g,h)$ of a warping path p is defined as $c_p(g,h) = \sum_{i=1}^{l} c(g_{a_i}, h_{b_i})$, for some cost function c. The warping path that minimizes the cost $c_p(g,h)$ is called the optimal warping path. □

Here, the L1 distance metric is chosen as the cost function of the warping path, giving $c(g_{a_i}, h_{b_i}) = |g_{a_i} - h_{b_i}|$. Finding the optimal warping path can be done efficiently through dynamic programming [Muller, 2007]. The resulting cost of the optimal warping path can be seen as a measure of similarity between g and h. An example of the optimal warping path for two sequences can be seen in Figures 5.3 and 5.4.

To determine the potential step sequences, the accelerometer magnitude signal S is first filtered using a zero-phase 20th order band-pass filter with a passband between 1 and 2.5 Hz. The filter was created using the Parks-McClellan algorithm implemented in Matlab.

Peak detection is then used to find a set of peaks X on the filtered signal S'. To avoid too much overlap, peaks that are too close to each other are removed using the following procedure: starting with the first detected peak x_1, and for each successive peak x_i, determine

$$\arg\max_{j=1,\ldots,m} s'_{x_j} | x_j \in [x_i + d_{\min}, x_i + d_{\max}]$$

for parameters $d_{\min}$ and $d_{\max}$. In other words, for a peak x_i, it is determined if there exist any peaks in the interval $[x_i + d_{min}, x_i + d_{max}]$, and if so, determine the highest peak x_j in the interval. If such a peak exists, x_j is considered the next peak after x_i, and all peaks x_l with $i < l < j$ are removed from X, and the algorithm proceeds from x_j. If no peak is found within the interval, the algorithm continues from x_{i+1}.

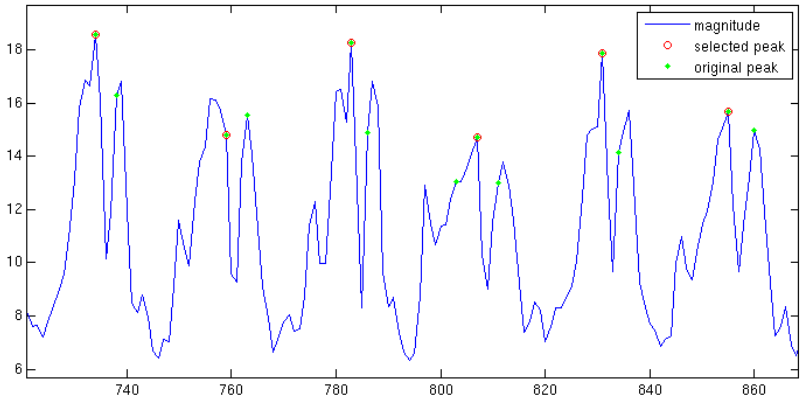


Figure 5.5: Example of the ISD algorithm on walking data, showing the accelerometer magnitude, the originally detected peaks, and the resulting peaks selected by the ISD algorithm. Note that the figure shows the unfiltered accelerometer magnitude, which as a result causes some of the peaks detected on the filtered magnitude to appear misaligned.

For each remaining peak $x_i \in X$, a potential step sequence is generated.

Definition 5.8 (Step sequence). The step sequence $q(x_i)$ for peak $x_i \in X$ is defined as $q(x_i) = s'_{m^-(x_i)}, \ldots, s'_{m^+(x_i)}$, with

$$m^+(x_i) = \underset{j=x_i+1,\ldots,x_i+\delta(X)}{\arg\min} s'_j$$
$$m^-(x_i) = \underset{j=x_i-\delta(X),\ldots,x_i-1}{\arg\min} s'_j,$$

where $\delta(X)$ is the median distance between the successive peaks in X and s'_j refers to the filtered signal S'. □

Each sequence is then compared to its four closest neighbors (two on each side) by calculating the average cost of the optimal warping paths:

$$c(x_i) = \frac{1}{4} \sum_{j \in \{-2,-1,1,2\}} c\left(q(x_i), q(x_j)\right).$$

A peak is then accepted as a step if $c(x_i) < T$ for some threshold T. For the algorithm, the threshold was set as $T = 0.8$, based on empirical evaluation.

5.2.5 Iterative step detection

The iterative step detection (ISD) algorithm works by iteratively adding or removing peaks from a selection in order to optimize an error or fitness function and obtain the most likely set of footsteps. At each iteration, the algorithm will greedily add or

remove one or two peaks from the current selection, depending on what yields the largest improvement to the overall error, continuing to iterate until the improvement of the error falls below a certain threshold, or a maximum number of iterations is reached.

Like the dynamic time warping algorithm, the first step consists of applying a zero-phase band-pass filter, after which the local maxima X are determined on the filtered sequence S'. The algorithm maintains a subset of peaks $Q^t \subseteq X$, where t represents the number of iterations. The initial set of peaks is set to all detected peaks, that is, $Q_1 = X$. Like the peaks in X, the peaks in Q^t are assumed to be sorted in increasing order, or in other words, for all peaks $q_i^t \in Q^t$, it holds that $q_i^t < q_{i+1}^t$ for $i = 1, \ldots, m^t - 1$. Here, m^t represents the number of peaks in Q^t for iteration t.

For each peak $q_i^t \in Q^t$, a fit $f(q_i^t)$ can be calculated. This fit is based on the distance to nearby peaks, and the differences in amplitude between peaks. The fit can either be positive (for a good match) or negative (for a poor match). The fit for a selection of peaks Q^t is then given as

$$f(Q^t) = \sum_{i=1}^{m^t} f(q_i^t).$$

The algorithm then tries to improve on the overall fit $f(Q^t)$ by including and / or removing one or two peaks during each iteration. As a result, a new selection Q^{t+1} which maximizes the fit $f(Q^{t+1})$ is created by exploring all possible peak pairs. The iterative process continues until a maximum number of iterations is reached, or $f(Q^{t+1}) - f(Q^t) < T$ for some threshold T.

To determine $f(q_i^t)$, a fitness metric is used which relies on a set of four parameters, $\theta = (l, d, a, b)$. Here, l represents the number of other peaks that q_i^t will be compared to, given by the sequence $o_i = (q_{i-l}^t, \ldots, q_{i-1}^t)$. Furthermore, d is a fitness weight parameter for metrics related to the distance between peaks, a is a weight for metrics related to peak amplitudes, and b is a coefficient for the difference in time between successive peaks. The fitness metrics themselves are based on the following observations:

- Step durations are generally constant while walking.
- Successive steps must not be too close or too far after each other.
- Step duration rarely changes more than then 15% between successive steps.
- Step amplitudes are of roughly similar height (with some upward spikes).

Definition 5.9 (Iterative step detection fit). The fit $f(q_i^t)$ is determined as the sum $f(q_i^t) = f_1(q_i^t) + f_2(q_i^t) + f_3(q_i^t) + f_4(q_i^t)$ of the following four components:

- $f_1(q_i^t) = -b|\Delta t(o_i) - \Delta t(q_i^t, q_{i-1}^t)|$, where $\Delta t(q_i^t, q_{i-1}^t) = q_i^t - q_{i-1}^t$, and $\Delta t(o_i)$ is the mean difference in time between successive peaks in o_i.
- $f_2(q_i^t) = \begin{cases} d, & \text{if } \Delta t(q_i^t, q_{i-1}^t) > \delta_{\min} \wedge \Delta t(q_i^t, q_{i-1}^t) < \delta_{\max} \\ 0, & \text{otherwise.} \end{cases}$
- $f_3(q_i^t) = \begin{cases} d, & \text{if } |\Delta t(o_i) - \Delta t(q_i^t, q_{i-1}^t)| < 0.15 \cdot \Delta t(o_i) \\ 0, & \text{otherwise.} \end{cases}$
- $f_4(q_i^t) = \begin{cases} -a, & \text{if } s'_{q_i^t} < s'_{o_i} \\ 0, & \text{otherwise} \end{cases}$

  where s'_{o_i} is the mean of the peak height in o_i.

□

The optimal values for $\delta_{\max}$ and $\delta_{\min}$ depend on the target population (e.g., healthy or movement-impaired). The minimum and maximum step lengths used here are 0.4 seconds and 0.8 seconds, respectively.

An example of the ISD algorithm on walking data is shown in Figure 5.5. For the implementation used in the results (Section 5.4), the parameters were set based on empirical evidence as $l = 7$, $d = 45$, $a = 35$, and $b = 3$. It should be noted that when implementing this algorithm, it is not necessary to recalculate the fit of every peak whenever a new peak is added or removed, since such a change affects at most l peaks for which the fit needs to be recomputed. Implementing the algorithm in such a manner can greatly increase the speed of the algorithm, particularly for sequences with a large number of peaks. For very large sequences, an 'online' version of the algorithm can be used, which only evaluates a certain window of peaks at a time, although this can lead to a small loss of performance.

5.2.6 Improving cadence estimation performance through post-processing

Regardless of the algorithm used, cadence estimation can sometimes be improved through post-processing steps based on the final set of detected steps - for instance, steps which are too close to one another can be removed to create a more accurate sequence of steps. Below, two post-processing approaches are discussed, which are used to improve cadence estimation for a number of the algorithms described above.

Improving performance through heuristics

A way to improve cadence estimation on longer sets of walking data is finding 'good' segments of the data to base the cadence estimate on, and disregard results on noisy data where steps are missing or detected incorrectly. To determine the segments that are kept or discarded, a set of heuristics can be used similar to the ISD algorithm.

In this case, the observation that successive steps rarely change more than 15% in duration is used, as well as the fact that there is an expected maximum and minimum duration between successive steps. Steps which do not fit these criteria are marked as invalid.

If one or more segments of at least 15 successive, valid steps are found, then these segments are used to determine the estimated cadence. If no such segment is found, cadence is determined over all footsteps that were detected. Using this method, it is important to retain a decent number of steps to reduce the risk of high variance in the cadence estimates, caused by basing the estimate on only a small sample of the total data. It is therefore advisable to choose parameters such as maximum and minimum step length conservatively.

The heuristic described here is used for the DTW and QRS algorithms. It was not added to the ISD algorithm as this set of heuristics is already included implicitly in its fitness function. Similarly, it was left out of the autocorrelation algorithm as the algorithm does not explicitly detect steps.

Improving performance with autocorrelation

Apart from directly estimating cadence with autocorrelation, it is also possible to use the autocorrelation of the walking signal to try and improve other step detection algorithms. One way to use the autocorrelation is to iterate over all footsteps detected successively, using the autocorrelation lag as an indicator of where the next step is expected to be located. For a detected peak x_i and lag l, let x_a be the peak closest to $x_i + l$. Then, if $x_i + l - \delta \cdot l \leq x_a \leq x_i + l + \delta \cdot l$, x_a is selected as the next peak, and all intermediate steps between x_i and x_a are removed. Otherwise, x_{i+1} remains the next peak and the algorithm continues from there.

It should be noted that using the autocorrelation lag in this manner can also hurt performance when the lag does not accurately match the actual step duration. This can result in removing detected steps which were actually valid.

5.3 Data collection

To investigate the step detection algorithms detailed in Section 5.2, a wireless inertial measurement unit named ‘SMM’ was used. The SMM contains a number of sensors, including a gyroscope, magnetometer, pressure sensor and three tri-axial accelerometers. For the purpose of this study, only the output from one of the tri-axial accelerometers was considered; for many practical applications, only a single accelerometer is available, mainly due to constraints related to power consumption. Accelerometer data was collected at a sampling rate of 50 Hz, and stored in the internal flash memory of the SMM. At the end of the protocol, the data was transferred to a computer through a USB connection.

As part of the study, 20 healthy participants (average age: 35.4 ± 10.3) were recruited for a treadmill-walking task and a free-walking task. During these tasks, the participants wore one SMM data collection device at the wrist using a fitted pouch attached to a Velcro strap, and one SMM as a pendant using a cord around the neck. No particular attention was given to the orientation of the SMM, however, participants were asked to wear the SMM on their non-dominant wrist.

During the treadmill task, participants walked on a treadmill at 12 different speeds, ranging from 1.5 to 6 km/h, with increments of 0.5 km/h. Participants walked each speed for a duration of 90 seconds; for the purposes of data analysis, only the last 60 seconds were considered, in order to allow the participants to adjust to the speed of the treadmill. They were asked to walk with their arms free (as opposed to holding the side rails) if they felt secure in doing so.

During the free-walking task, participants were asked to walk a pre-defined 24 meter trajectory consisting of straight line down a corridor, with marked start and end points. Each participant completed this track three times at three different speeds: their preferred speed, slow, and fast. Participants were allowed to use their own interpretation of what they considered as slow or fast speeds. During both tasks, a camera was used to collect ground-truth gait information through manual annotation of the number of footsteps at each treadmill speed. Due to technical issues with the devices (1) and (partially) missing or unusable video recordings (5), data from 14 participants was included in the analysis.

5.4 Results

To estimate the performance of the algorithms on cadence estimation, the root mean square error (RMSE) between the recorded ground truth and the cadence estimated by the algorithms is used. The RMSE is defined as

$$e = \sqrt{\frac{1}{p}\sum_{i=1}^{p}(y_e - y_{gt})^2}$$

where p is the number of participants, y_e is the estimated cadence, and y_{gt} is the cadence recorded as the ground truth.

Using the accelerometer worn on the pendant position as input, the performance of the algorithms on the treadmill task is shown in Figure 5.6. As can be seen in the figure, all algorithms perform well on the higher walking speeds of 3.5 km/h or more. At lower walking speeds, however, the RMSE increases for many of the algorithms, and for the autocorrelation algorithm in particular. The boosted QRS algorithm is less affected by the decrease in walking speed.

For the wrist position, we would generally expect algorithms to perform worse; this is in part because we measure movement further away from the 'source', and because the wrist position is likely to experience more movements not related to

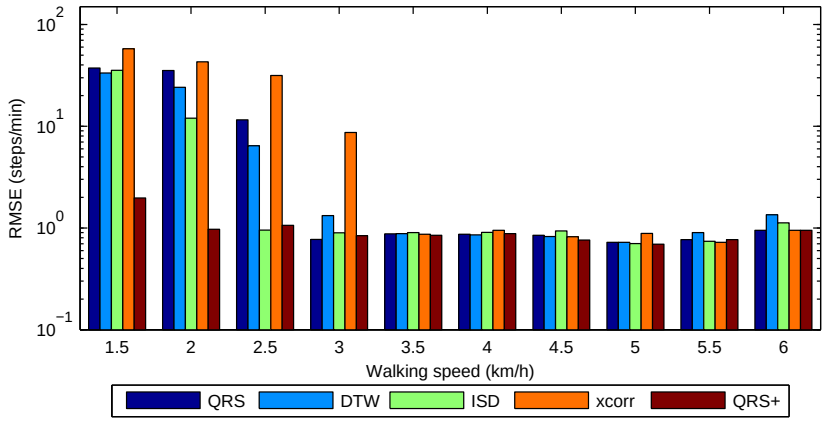


Figure 5.6: Performance of the cadence estimation algorithms on the treadmill task, using data from the pendant position. For each algorithm, the root mean square error is shown on a logarithmic scale for each of the walking speeds.

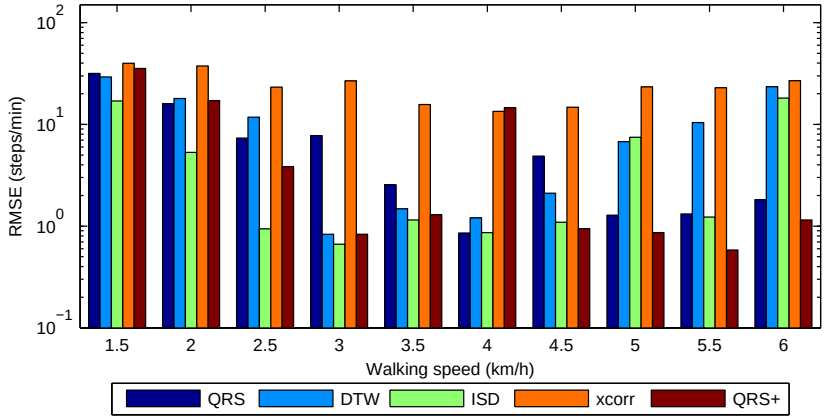


Figure 5.7: Performance of the cadence estimation algorithms on the treadmill task, using data from the wrist position. For each algorithm, the root mean square error is shown on a logarithmic scale for each of the walking speeds.

walking. In Figure 5.7, we can see that this is indeed the case. Here, relatively large errors do not only occur at slow walking speeds, but also at higher walking speeds. This is possibly due to the algorithms having difficulty distinguishing between the step frequency and the stride frequency in the signal.

In general, the algorithms still perform better at higher walking speeds compared to the lower walking speeds. For some algorithms, outliers in RMSE can be observed at particular speeds. This is generally due to misestimation of a single participant's

	pendant					wrist				
speed	QRS	DTW	ISD	xcorr	QRS+	QRS	DTW	ISD	xcorr	QRS+
1.5	37.30	33.41	35.51	57.61	**1.97**	31.72	29.23	**16.96**	39.90	35.37
2	35.33	24.19	12.01	42.90	**0.97**	15.95	17.96	**5.30**	37.52	17.13
2.5	11.57	6.42	**0.95**	31.54	1.06	7.32	11.76	**0.94**	23.19	3.84
3	**0.77**	1.32	0.90	8.69	0.84	7.75	0.83	**0.66**	26.75	0.83
3.5	0.88	0.88	0.90	0.87	**0.85**	2.54	1.48	**1.15**	15.68	1.29
4	0.87	**0.86**	0.90	0.95	0.88	**0.86**	1.20	**0.86**	13.44	14.61
4.5	0.85	0.82	0.93	**0.72**	0.76	4.88	2.10	1.09	14.76	**0.94**
5	0.72	0.72	0.70	0.88	**0.69**	1.28	6.77	7.46	23.41	**0.86**
5.5	0.77	0.90	0.74	**0.72**	0.77	1.32	10.40	1.23	22.95	**0.58**
6	**0.95**	1.35	1.12	**0.95**	**0.95**	1.82	23.51	18.15	26.84	**1.15**

Table 5.1: Root mean squared error scores for the QRS, DTW, ISD, autocorrelation (xcorr) and QRS combined with autocorrelation (QRS+) algorithms at various treadmill speeds for the pendant and wrist positions. The bolded values indicate the best performance for each walking speed and wearing position.

position	speed	QRS	DTW	ISD	xcorr	QRS+
pendant	preferred	2.18	2.04	2.28	2.44	2.22
	slow	31.41	23.09	21.52	38.81	2.38
	fast	4.01	8.97	5.40	3.59	4.11
wrist	preferred	9.30	13.13	7.11	35.26	1.85
	slow	25.53	18.20	9.57	31.73	21.19
	fast	24.78	36.87	39.38	46.32	25.47

Table 5.2: The root mean squared error of each of the algorithms on the free-walking task, for both the pendant and wrist wearing positions, while walking at either the preferred, slow or fast walking speeds.

cadence. Examples include the combined QRS and autocorrelation algorithm, and the ISD algorithm on the wrist position, at the 4 km/h and 5 km/h speeds respectively. The exact RMSE for each of the algorithms for all treadmill speeds and both wearing positions is shown in Table 5.1.

The deterioration in performance can partly be attributed to a decrease in signal-to-noise ratio at slower speeds. For step detection in healthy adults, however, this may not be a major issue, as many people have a preferred walking speed which is sufficiently high to avoid these issues. For populations that tend to walk at slower speeds, the deterioration in performance may prove to be substantial. Table 5.2 shows

the RMSE of the algorithms on the free-walking task for both wearing positions, and the slow, fast and preferred walking speeds.

At the preferred speed (average in km/h: 4.84 ± 0.52 SD), most algorithms provide reasonable estimates for the actual cadence regardless of wearing position. When the participants were asked to walk slowly (average in km/h: 2.97 ± 0.85 SD), however, we can see in the table that there are more cases where the estimated cadence differs substantially from the actual cadence, resulting in increased RMSE. At fast speed (average in km/h: 6.36 ± 0.60 SD) there is a clear difference between the wearing positions. At the pendant position, performance remains fairly high, while at the wrist position, a deterioration similar to slow walking speeds can be observed; likely again due to the difficulty in distinguishing between steps and strides in the accelerometer signal.

5.5 Conclusion

As was expected, walking speed has a considerable impact on the performance of the accelerometry-based algorithms for cadence analysis. For walking speeds above 3 km/h, it can be seen that at least for the pendant position, all algorithms show comparable performance. Below this speed, the error rates rise considerably, in some cases dramatically so. It is notable that at these low speeds, the QRS combined with autocorrelation shows the best performance for the pendant position, while the ISD algorithm shows the best performance for the wrist position. This may be an example of step detection algorithms being specialized and performing well in one scenario, while suffering in others as a result - an observation previously made by Marschollek et al. [2008].

For the wrist position, there are occasionally large increases in error rate for higher speeds as well. This is likely due to underestimation of the cadence, which is most likely caused at least in part by the swing of the arm that often accompanies walking. The duration of the arm swing is typically equal to the stride duration, rather than the step duration, resulting in the actual cadence being underestimated.

The algorithms in this study have not been tailored to the individual characteristics of the study participants. The advantage is that these algorithms are expected to provide the same level of performance to new users 'out of the box'; a similar approach is applied by the majority of commercial applications of step detection. Theoretically, performance may be improved by learning the individual walking characteristics of a user. However, there is currently little research into this topic, also because applications are often constrained in terms of computational resources. Even so, the automatic and unobtrusive learning of a user's walking characteristic may be an interesting topic for future research.

Comparing the performance of the investigated algorithms to the state of the art, Storm, Heller, and Mazzà [2015] describe a study showing that the best perform-

ing commercially available step detectors achieve an accuracy of $1-3\%$ in terms of mean absolute percentage error. While not directly comparable to RMSE, the errors observed during the faster walking speeds are in the region of $0.7-0.9$ RMSE, which seem comparable in terms of magnitude. A number of step detection methods and applications investigated by Fortune et al. [2014] show similar results. Like we observed in our study, agreement between the methods and the ground truth decreases at lower walking speeds. Marschollek et al. [2008] report relatively high errors in the step detection methods investigated; this is potentially related to testing on an older population (likely yielding lower walking speeds), and a relatively short walking distance of 20 meters.

There are a number of limitations to this study. First, participants were selected from a research campus and as such are not representative of the population of healthy adults as a whole. Also, while walking slowly is more similar to the normal gait of elderly or people with mobility impairments, there are likely to be differences in gait between these groups which might impact the results of the algorithms. In addition, it has been noted that people change their gait while walking on a treadmill compared to walking freely, which might also have an effect on the result. Finally, while none of the algorithms have been 'trained' on the participants of this study, in some cases parameters have been (globally) adjusted to obtain accurate performance. Further study is required to determine if these parameter changes are optimal for other participants and populations.

The results described in this chapter suggest that for cadence estimation and step detection, care must be taken when measuring populations that are likely to walk slowly, such as elderly, people with mobility impairments, or Parkinson patients, as accuracy drops off considerably for many algorithms. In addition, measuring at a position other than the wrist might be preferable for these groups. For populations more inclined to walk at regular speeds, most algorithms investigated here show similar, fairly accurate performance. For such populations, wrist-based solutions can be preferable, as accelerometers can be easily integrated into wrist-worn devices such as a smart watch.

6

Conclusion

The research in this thesis describes the measuring and interpretation of human activity and mobile behavior through the use of pervasive and unobtrusive sensors. It is well understood that activity and health are closely related. The need for the promotion of an active lifestyle in an increasingly sedentary society where obesity and cardiovascular disease are becoming more widespread is a well-known example of this. The link between health and activity does not stop at physical exercise, however. There is a lot of information to be gained by observing people while they are active; changes in motor function can herald a decline in health, and the lack or presence of certain behaviors can indicate physical or mental issues as well.

The applications described above all benefit from continuous measurement, in contrast to occasional check-ups. The former can provide instant and more detailed feedback, and changing values can be picked up immediately. To facilitate continuous, or pervasive, measurement, sensors should ideally be minimally obtrusive, so they do not interfere with normal behavior or become a burden to the user. In addition, to allow ease of use such sensors are often restricted in size or power. Regardless of the sensor modalities used, the raw output takes the form of a time series; a series of discrete measurements ordered in time. These form the basis of the analysis and interpretation of the measured activities and behaviors.

In this thesis, a number of topics related to human motion tracking are addressed. In Chapter 2, the k-segmentation problem is discussed; this is a common problem

in the analysis of data over time, and consists of partitioning a set of measurements into k segments, such that an error criterion is minimized. In many practical applications, one is interested in finding particular segments that for example match certain activities, or correspond to a particular time of day. For many error criteria, such as the Lp criteria (with $p = 1, 2, \ldots$), the k-segmentation problem can be solved in polynomial time using dynamic programming, for measurements with a fixed, equidistant interval, and where segment boundaries align with measurement boundaries. In this chapter, we show that also for measurements with variable or real-valued durations, or cases where item boundaries and segment boundaries are not aligned, an optimal k-segmentation can still be found in polynomial time for the Lp error criteria. In addition, we show for the L1 and L2 error criteria that more specifically, the segmentation error as a function of the coverage of an item contains at most a single maximum, and no other stationary points.

In Chapter 3, a method is described to unobtrusively track activities of daily living (ADL) in a kitchen environment. ADLs are daily activities that play a role in self-care; examples include dressing, eating, personal hygiene, and so on. ADLs can be tracked using a camera and microphone; through the camera, a user's location can be tracked within the room, while the microphone can pick up characteristic sounds. Scene analysis techniques can then be used to identify locations and sounds from the raw sensor data. Methods are detailed for deriving discrete events from the observed locations and sounds, and for recognizing ADLs through the use of hidden Markov models (HMMs). A number of variations of the HMM are explored, including the MA algorithm for parameter estimation, and the use of coupled hidden Markov models for combining inputs from audio and video. The chapter also describes a study with eight participants in which six different ADLs are recorded for each participant. Audio and video events were both annotated manually and derived automatically through scene analysis techniques. Using the data from the study, the well-known Baum-Welch parameter estimation is compared to the MA parameter estimation method. The use of chained hidden Markov models for ADL classification is also investigated. When ADLs are classified offline, after observing the entire activity, a maximum accuracy of 97% is achieved on annotated data, and 94% on scene analysis data. Classifying online using a sliding window approach, where only part of an activity is observed and different ADLs may overlap, yields an accuracy of 79% on annotated data, and 73% on scene analysis data. In addition, it is shown that the multi-modal approach of using both video and audio yields superior performance compared to using a single modality only.

In Chapter 4 the focus is shifted from activities of daily living to physical activity in general. In this chapter, it is investigated whether it is possible to predict if people participating in a lifestyle activity program drop out before the program's comple-

tion. For the duration of a lifestyle activity program, the amount of daily physical activity of a participant is monitored through a small body-worn sensor, and can be viewed online by the participant. In addition, the participant has a tailored set of daily activity targets, and can contact a personal coach through email. The aim of the program is to slowly increase the amount of physical activity over its duration, hopefully leading to a permanent increase once the program ends. Participants who terminate their participation early for whatever reason (referred to as 'dropouts') risk their initial aims of achieving a more active lifestyle. If potential dropouts can be addressed in a timely fashion, interventions might still be possible. In this chapter it is investigated which factors (called markers) contribute to dropout probability using a database of 950 participants of a twelve-week lifestyle activity program. In addition, a method based on genetic programming is discussed to combine existing markers into a new set of markers with a higher level of predictive power. Using principal component analysis and a k-nearest neighbor classifier, results show that without the use of genetic programming to combine markers, classification accuracy is approximately 64%. By combining markers through genetic programming, classification class accuracy increases to 72%, which equates to a reduction in the number of errors of approximately 23%.

Looking closer at how the amount of physical activity can be determined, Chapter 5 investigates methods for step detection and cadence estimation. Step detection refers to the detection of individual footsteps when a person is walking, and forms the functional part of a pedometer. In addition, cadence estimation and step detection have important applications in medicine and healthcare, where changes in gait can be indicative of underlying health problems such as the onset of cognitive decline. The focus of this chapter is on comparing various cadence estimation techniques based on tri-axial accelerator data. Here, four cadence estimation methods (autocorrelation, QRS, dynamic time warping and iterative step detection) are explored for various wearing positions of the accelerometer, namely the pendant and wrist wearing positions. In addition, the performance of the algorithms is investigated for a variety of walking speeds, ranging from 1.5 to 6 km/h. As expected, performance seems to deteriorate for lower walking speeds - this is particularly relevant for elderly or people with gait abnormalities, as these populations often walk considerably slower compared to 'healthy' populations. In addition, the results suggest that pendant-based cadence estimation is more accurate compared to wrist-based cadence estimation.

In addition to the topics described in the individual chapters, we addressed the following main research question:

> *How can segmentation, classification, and regression be applied to problems that involve the tracking and interpretation of human motion?*

In light of the main research question, the following three related research questions were addressed:

- Can activities of daily living (ADL) be unobtrusively tracked and recognized?
- Can analyzing the behavior of people trying to be more physically active help predict if they will drop out of a lifestyle physical activity program?
- Is it possible to determine (psycho)motor skills such as gait accurately using wearable sensors?

With regard to the first research question, can activities of daily living (ADL) be unobtrusively tracked and recognized, a model to unobtrusively recognize ADLs was developed, and described in Chapter 3. To address this research question, both the segmentation problem and the classification problem discussed in Section 1.6 need to be solved. The results from Chapter 3 suggest that the problem of unobtrusive ADL tracking can be solved with reasonable accuracy, however, a number of challenges still remain. One of these challenges pertains to the quality of the measured sensor data. As discussed in Chapter 3, the recognition results based on manually annotated data are generally superior to those on automatically classified sensor data, indicating that the feasibility of ADL recognition at least in part depends on the accuracy of the employed scene analysis techniques. Another aspect is what sensor modalities are employed; as can be seen in the results of Chapter 3, some sensor modalities yield better results compared to others. In addition, in many cases it may be beneficial to combine several sensor modalities, as this may provide better results compared to any individual modality.

Another remaining challenge is in distinguishing multiple persons in such a system, or how to discard sensor inputs not created by the intended user of the system. This remains an open question for now. One solution is to let the intended users wear tags or other items that help identify them. However, this will likely reduce the unobtrusiveness of the system, and introduces problems when the user forgets to wear their tag. Other options include more unobtrusive means of identification, such as the use of a camera for face recognition. Such methods generally still have limitations, for example, a user must look in the direction of the camera for face recognition to work. Finally, the results in Chapter 3 also demonstrate the added complexity introduced when both the classification and segmentation problems need to be solved simultaneously. This can be seen by comparing the results from Section 3.4, where only the classification problem is considered, to the results from Section 3.5, where both problems are considered.

In terms of the second research question, can analyzing the behavior of people trying to be more physically active help predict if they will drop out of a lifestyle physical activity program, a large set of data from such a program was analyzed to determine if the event of a participant dropping out can be predicted in advance. The results of this study are described in Chapter 4, and indicate that the analysis of the

participants' behavior and characteristics can assist in providing an early warning to coaches regarding potential dropouts. However, the results also illustrate some of the challenges with such approaches in general. In particular, the large number of data markers typically available in such data sets can make it challenging to find the correct markers for the prediction problem at hand. The genetic programming algorithm used can help with this issue to some extent, as it performs a semi-guided search of the marker space to determine which markers may be valuable for a particular prediction problem. There also exist several other methods for marker selection, for example stepwise selection for regression models. Some care must be taken when using such methods however, as they should not preclude critical thinking by the researcher with regard to which markers may be relevant for inclusion and as to why certain markers may have been selected.

In addition, the study in Chapter 4 shows that if the markers in a data set have little predictive power individually, it may be worthwhile to attempt to combine markers, and as a result create a new set of markers. More generally, any algorithm for predictive analysis can only perform as well as what information is provided by the available markers. In some cases, there is more to be gained from exploring new markers that can be derived from the overall data set, than by attempting to improve the algorithms used for the predictive analysis over the data markers (which is not to say the latter should not also be attempted).

With regard to the final research question, is it possible to determine (psycho)motor skills such as gait accurately using wearable sensors, we show in Chapter 5 that it is possible to accurately determine cadence using a single accelerometer worn at the wrist or around the neck. However, we also show that there are a number of factors that impact how accurately gait can be determined. In particular, walking speed has a considerable impact on how accurately cadence can be determined using an accelerometer. At lower walking speed, the accelerations due to walking become less pronounced, making accurate estimation more difficult. It seems likely that the estimation of other gait features, such as walking speed or step variance, would also suffer from this effect at lower walking speeds. In addition, the wearing position of the accelerometer also plays a role in the accuracy of gait estimation. A wearing position that is further detached from the source of the walking accelerations is more likely to be affected by noise due to other movements. This is also illustrated in Chapter 5, where using the wrist as a wearing position yields inferior performance compared to the pendant position in most cases. At lower walking speeds, the effects of the wearing position are often exacerbated. For applications designed for populations with low expected walking speeds, such as for instance elderly, it may be worthwhile to consider carefully the wearing position of a gait estimation sensor.

6.1 Discussion on human motion tracking

This thesis concerns the field of human motion tracking. While there are many applications in this field (including security, sports, and social interaction), we expect the field to be increasingly driven by applications in healthcare and healthy lifestyles, as the world continues to age [Mathers et al., 2015] and global health-related costs continue to increase [WHO, 2011]. As the pressure on the world's healthcare systems continues to mount, we will increasingly have to look at technology solutions to assist and lower the burden on healthcare professionals and carers. While there are promising innovations, there are also still challenges to overcome [Wiederhold, Riva, and Graffigna, 2013; Free et al., 2013; de Joode et al., 2012]. As we have seen throughout this thesis, applications in human motion tracking have a role to play in these developments.

In addition, the field is, and will likely continue to be, driven by the increasing ubiquity of sensors in our daily lives; the main contributors to this being the proliferation of smartphones and smartwatches, and the developments surrounding the Internet of Things (IoT) [Gubbi et al., 2013]. In particular, smartphones and smartwatches offer widespread sensor platforms already carried daily by a large portion of the population; for some applications, this precludes the need of wearing a specialized sensor platform. In addition, as platform and sensor capabilities expand, there will be increasingly less need for a commercial application to develop its own sensor solution, making the development of such applications more accessible and less costly.

There are already a number of publications on the topic of gait analysis describing applications tailored for smartphone use [Bradjic and Harle, 2013; Huang et al., 2010; Li et al., 2012; Mladenov and Mock, 2009], as well as on the topic of human activity recognition [Mitchell, Monaghan, and O'Connor, 2013; Ronao and Cho, 2016]. Indeed, the algorithm described in Chapter 5 could also be implemented for a smartphone or smartwatch application. However, there are also downsides to these platforms that should be kept in mind, such as power consumption, differences between various models in terms of computational power and sensor availability, and low control over the system as a whole.

The Internet of Things, very generally speaking, refers to the trend of more sensor-capable devices becoming connected, and as such, allowing us to collect potentially very large amounts of data of a person's daily life activities and behavior. While there already is considerable literature on how the IoT can be deployed [Perera et al., 2014], there are still many challenges to be solved, and perhaps as a consequence there have been few practical applications in the field of human motion tracking. For this field, the most relevant extension of the IoT is at the moment the smart home; a home embedded with sensors that can be used to track and react to its residents.

The concept of the smart home is closely related to for example the work of Okour, Maender, and Basilakis [2015] or van Kasteren et al. [2008], using a large number of sensors embedded in a home environment. However, as few people actually live in a smart home at the time of writing, its practical applications are limited. While smart homes may have a big impact on the field of human motion tracking in the future, it seems currently more feasible for solutions based on environmental sensors to be restricted to a limited number of easy to install sensor boxes, such as the system described in Chapter 3 or by Urwyler et al. [2015].

With increasingly large amounts of data available from sensors, it is feasible that techniques from big data analytics will play a larger role in future applications of human motion tracking. Currently, however, our ability to collect all this data is limited by many mobile sensors, as the streaming of all data created is often prohibitive in terms of battery life and bandwidth. As such, for many mobile devices, we must still rely on local processing of the data.

To an extent, this can be addressed by temporarily storing data on the device itself, and then synchronizing with a local internet connection, as employed by the sensor described in Chapter 4. This does have disadvantages for some applications (for example, real-time data is not available), but is one of the approaches that allows us to capture large amounts of data using a mobile sensor. While the data set described in Chapter 4 is relatively small in terms of big data, it is already clear that finding information in the data set through manual inspection is very difficult, and techniques from data mining and big data analytics are required.

With regard to the model we described in our main research question, consisting of segmentation, classification, and regression, we have seen that the problems described in this thesis all fit with one or more components of this model. While the model describes the main components of most problems in the field of human motion tracking, it is not a sequence of steps that can be applied to solving all problems in this field; the main reason being that there are often domain-specific steps to be taken in for example preprocessing of data or the expected behavior of a set of measurements over time. This means that a practitioner of this field cannot simply apply a method from one application they are familiar with to a new domain, and expect to be immediately successful. As such, in the field of human motion tracking, one must draw from a wide variety of expertise, including machine learning, data analytics, signal processing, movement science, and psychology.

6.2 Contributions

Finally, we briefly describe the contributions made to the field of human motion tracking in this thesis.

- We show that the k-segmentation problem can be solved in $O(n^2k)$ time, even for sequences consisting of items with non-unit durations, or where the alignment requirement needs not hold.
- For the L1 and L2 error criteria, we show that between two item boundaries, the error function contains at most a single maximum, and no other stationary points.
- We show that the recognition of activities of daily life (ADL) in a kitchen environment is feasible using a single camera and microphone.
- We investigated the use of coupled hidden Markov models to the detection of ADLs based on video and audio data. In addition, we investigated the MA algorithm for parameter estimation in this context, a method that uses both positive and negative examples for parameter estimation.
- We investigate the contribution of each individual modality in audio and video-based ADL recognition.
- The performance of ADL recognition based on individual, manually segmented sequences of audio and video data is compared to the performance when input is provided as a continuous, unsegmented steam of audio and video data.
- We describe a genetic programming approach to combining individually weak features for predicting dropout risk of a population in a large, high-dimensional data set, and compare this method with the heuristic approach used in practice.
- We compare a number of different classes of accelerometry-based gait analysis algorithms in terms of performance for different walking speeds, under treadmill and free-walking conditions, for wearing positions at the wrist and pendant position.
- We introduce an algorithm based on iterative step detection for the detection of cadence.

Bibliography

Aggarwal, J. and Q. Cai [1997]. Human motion analysis: A review. In *Proceedings of the IEEE Non-Rigid and Articulated Motion Workshop*, pp. 90–102. IEEE. doi:10.1109/NAMW.1997.609859.

Aggarwal, J. K. and L. Xia [2014]. Human activity recognition from 3D data: A review. *Pattern Recognition Letters* **48**, 70–80. doi:10.1016/j.patrec.2014.04.011.

Aghabozorgi, S., A. S. Shirkhorshidi, and Y. W. Teh [2015]. Time-series clustering – A decade review. *Information Systems* **53**, 16–38. doi:10.1016/j.is.2015.04.007.

Alzantot, M. and M. Youssef [2012]. Uptime: Ubiquitous pedestrian tracking using mobile phones. In *IEEE Wireless Communications and Networking Conference*. IEEE. doi:10.1109/WCNC.2012.6214359.

Auger, F., M. Hilairet, J. Guerrero, E. Monmasson, T. Orlowska-Kowalska, and S. Katsura [2013]. Industrial applications of the Kalman filter: A review. *IEEE Transactions on Industrial Electronics* **60** (12), 5458–5471. doi:10.1109/TIE.2012.2236994.

Azad, R. K., J. S. Rao, W. Li, and R. Ramaswamy [2002]. Simplifying the mosaic description of DNA sequences. *Physical Review E* **66** (3), 031913. doi:10.1103/PhysRevE.66.031913.

Baldi, P. and Y. Chauvin [1994]. Smooth on-line learning algorithms for hidden Markov models. *Neural Computing* **6** (2), 307–318. doi:10.1162/neco.1994.6.2.307.

Bao, L. and S. S. Intille [2004]. Activity recognition from user-annotated acceleration data. In A. Ferscha and F. Mattern (Eds.), *Pervasive Computing*, Volume 3001 of *Lecture Notes in Computer Science*, pp. 1–17. Springer. doi:10.1007/978-3-540-24646-6_1.

Bataineh, M., T. Marler, K. Abdel-Malak, and J. Arora [2016]. Neural network for dynamic human motion prediction. *Expert Systems with Applications* **48**, 26–34. doi:10.1016/j.eswa.2015.11.020.

Bellman, R. [1961]. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM* **4** (6), 284. doi:10.1145/366573.366611.

Bingham, E. [2010]. Finding segmentations of sequences. In S. Dzeroski, B. Goethals, and P. Panov (Eds.), *Inductive Databases and Constraint-Based Data Mining*, pp. 177–197. Springer New York. doi:10.1007/978-1-4419-7738-0_8.

Bohannon, R. [1997]. Comfortable and maximum walking speed of adults aged 20–79 years: Reference values and determinants. *Age and Ageing* **26** (1), 15–19. doi:10.1093/ageing/26.1.15.

Bonato, P. [2010]. Wearable sensors and systems. *IEEE Engineering in Medicine and Biology Magazine* **29** (3), 25–36. doi:10.1109/MEMB.2010.936554.

Boulgouris, N., D. Hatzinakos, and K. Plataniotis [2005]. Gait recognition: A challenging signal processing technology for biometric identification. *IEEE Signal Processing Magazine* **22** (6), 78–90. doi:10.1109/MSP.2005.1550191.

Bradjic, A. and R. Harle [2013]. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 225–234. ACM. doi:10.1145/2493432.2493449.

Brand, M. [1997]. *Coupled Hidden Markov Models for Modeling Interactive Processes*. Technical report, MIT Media Lab Perceptual Computing /Learning and Common Sense, Technical Report 405.

Brand, M., N. Oliver, and A. Pentland [1997]. Coupled hidden Markov models for complex action recognition. In *Proceedings of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 994–999. IEEE. doi:10.1109/CVPR.1997.609450.

Bucks, R. S., D. L. Ashworth, G. K. Wilcock, and K. Siegfried [1996]. Assessment of activities of daily living in dementia: Development of the Bristol activities of daily living scale. *Age and Ageing* **25** (2), 113–120. doi:10.1093/ageing/25.2.113.

Bulling, A., U. Blanke, and B. Schiele [2014]. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys* **46** (3), 33. doi:10.1145/2499621.

Chen, D., J. Yang, R. Malkin, and H. D. Wactlar [2007]. Detecting social interactions of the elderly in a nursing home environment. *ACM Transactions on Multimedia Computing, Communications, and Applications* **3** (1), 6. doi:10.1145/1198302.1198308.

Chundi, P. and D. Rosenkrantz [2008]. Efficient algorithms for segmentation of item-set time series. *Data Mining and Knowledge Discovery* **17** (3), 377–401. doi:10.1007/s10618-008-0095-0.

Churchill, G. A. [1989]. Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology* **51** (1), 79–94. doi:10.1007/BF02458837.

Cochran, W., J. Cooley, D. Favin, H. Helms, R. Kaenel, W. Lang, G. Maling, D. Nelson, C. Rader, and P. Welch [1967]. What is the fast Fourier transform? *Proceedings of the IEEE* **55** (10), 1664–1674. doi:10.1109/PROC.1967.5957.

Cooley, J. and J. Tukey [1965]. An algorithm for the machine calculation of complex Fourier series. *American Mathematical Society* **19** (90), 297–301. doi:10.2307/2003354.

Dadlani, P., P. Markopoulos, D. van Bel, K. Smolders, M. Pijl, B. de Ruyter, and E. Aarts [2013]. Similarity awareness: Using context sensing to support connectedness in intra-family communication. *Journal of Ambient Intelligence and Smart Environments - Design and Deployment of Intelligent Environments* **5** (5), 425–441. doi:10.3233/AIS-130219.

de Joode, E. A., M. P. J. van Boxtel, F. R. Verhey, and C. M. van Heugten [2012]. Use of assistive technology in cognitive rehabilitation: Exploratory studies of the opinions and expectations of healthcare professionals and potential users. *Brain Injury* **26** (10), 1257–1266. doi:10.3109/02699052.2012.667590.

de Ruyter, B. E. R., S. E. Baha, M. J. Pijl, and P. Markopoulos [2011]. The role of empathy in making availability judgments from video and silhouette awareness information. *The Ergonomics Open Journal* **4**, 41–46. doi:10.2174/1875934301104010041.

Debes, C., A. Merentitis, S. Suhkanov, M. Niessen, N. Frangiadakis, and A. Bauer [2016]. Monitoring activities of daily living in smart homes: Understanding human behavior. *IEEE Signal Processing Magazine* **33** (2), 81–94. doi:10.1109/MSP.2015.2503881.

Deshpande, N., E. J. Metter, S. Bandinelli, J. Guralnik, and L. Ferrucci [2009]. Gait speed under varied challenges and cognitive decline in older persons: A prospective study. *Age and Ageing* **38** (5), 509–514. doi:10.1093/ageing/afp093.

Dijkstra, B., W. Zijlstra, E. Scherder, and Y. Kamsma [2008]. Detection of walking periods and number of steps in older adults and patients with Parkinson's disease: Accuracy of a pedometer and an accelerometry-based method. *Age and Ageing* **37** (4), 436–441. doi:10.1093/ageing/afn097.

Dishman, R. K. and W. Ickes [1981]. Self-motivation and adherence to therapeutic exercise. *Journal of Behavioral Medicine* **4** (4), 421–438. doi:10.1007/BF00846151.

Dishman, R. K., J. F. Sallis, and D. R. Orenstein [1985]. The determinants of physical activity and exercise. *Public Health Reports* **100** (2), 158–171.

Du, Y., W. Wang, and L. Wang [2015]. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1110–1118. IEEE.

doi:10.1109/CVPR.2015.7298714.

Duda, R., P. Hart, and D. Stork [2000]. *Pattern Classification, 2nd Edition*. Wiley.

Ellis, K., J. Kerr, S. Godbole, G. Lanckriet, D. Wing, and S. Marshall [2014]. A random forest classifier for the prediction of energy expenditure and type of physical activity from wrist and hip accelerometers. *Physiological Measurement* **35** (11), 2191–2203. doi:10.1088/0967-3334/35/11/2191.

Flanagan, J. A., J. Mantyjarvi, and J. Himberg [2002]. Unsupervised clustering of symbol strings and context recognition. In *Proceedings of the 2002 International Conference on Data Mining*, pp. 171–178. IEEE. doi:10.1109/ICDM.2002.1183900.

Folstein, M. F., S. E. Folstein, and P. R. McHugh [1975]. "Mini-mental state". A practical method for grading the cognitive state of patients for the clinician. *Journal of Psychiatric Research* **12** (3), 189–198. doi:10.1016/0022-3956(75)90026-6.

Fortune, E., V. Lugade, M. Morrow, and K. Kaufman [2014]. Validity of using triaxial accelerometers to measure human movement - Part ii: Step counts at a wide range of gait velocities. *Medical Engineering & Physics* **36** (6), 659–669. doi:10.1016/j.medengphy.2014.02.006.

Free, C., G. Phillips, L. Galli, L. Watson, L. Felix, P. Edwards, V. Patel, and A. Haines [2013]. The effectiveness of mobile-health technology-based health behavior change or disease management interventions for health care consumers: A systematic review. *PLoS Medicine* **10** (1), e1001362. doi:10.1371/journal.pmed.1001362.

Gao, L., A. K. Bourke, and J. Nelson [2014]. Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems. *Medical Engineering & Physics* **36** (6), 779–785. doi:10.1016/j.medengphy.2014.02.012.

Garcia, A. W. and A. C. King [1991]. Predicting long-term adherence to aerobic exercise: A comparison of two models. *Journal of Sport & Exercise Physiology* **13** (4), 394–410.

Gavrilla, D. [1999]. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding* **73** (1), 82–98. doi:10.1006/cviu.1998.0716.

Ge, X., W. Pratt, and P. Smyth [1999]. Discovering Chinese words from unsegmented text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 271–272. ACM. doi:10.1145/312624.313472.

Gedikli, A., H. Aksoy, N. Unal, and A. Kehagias [2010]. Modified dynamic programming approach for offline segmentation of long hydrometeorological time series. *Stochastic Environmental Research and Risk Assessment* **24** (5), 547–557. doi:10.1007/s00477-009-0335-x.

Gheyas, I. A. and L. S. Smith [2010]. Feature subset selection in large dimensionality domains. *Pattern Recognition* **43** (1), 5–13. doi:10.1016/j.patcog.2009.06.009.

Gionis, A. and H. Mannila [2003]. Finding recurrent sources in sequences. In *Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology*, pp. 123–130. doi:10.1145/640075.640091.

Godbehere, A. B., A. Matsukawa, and K. Goldberg [2012]. Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In *American Control Conference*, pp. 4305–4312. IEEE. doi:10.1109/ACC.2012.6315174.

Goris, A. and R. Holmes [2008]. The effect of a lifestyle activity intervention program on improving physical activity behavior of employees. In *PERSUASIVE 2008: Proceedings of the 3rd International Conference on Persuasive Technology*, Berlin, Heidelberg, pp. 23–34. Springer-Verlag. doi:10.1007/978-3-540-68504-3_3.

Graf, C. [2008]. The Lawton instrumental activities of daily living scale. *American Journal of Nursing* **108** (4), 52–62. doi:10.1097/01.NAJ.0000314810.46029.74.

Gubbi, J., R. Buyya, S. Marusic, and M. Palaniswami [2013]. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* **29** (7), 1645–1660. doi:10.1016/j.future.2013.01.010.

Guo, H., L. B. Jack, and A. K. Nandi [2005]. Feature generation using genetic programming with application to fault classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **35** (1), 89–99. doi:10.1109/TSMCB.2004.841426.

Gupta, P. and T. Dallas [2014]. Feature selection and activity recognition using a single triaxial accelerometer. *IEEE Transactions on Biomedical Engineering* **61** (6), 1780–1786. doi:10.1109/TBME.2014.2307069.

Gwadera, R., A. Gionis, and H. Mannila [2006]. Optimal segmentation using tree models. In *ICDM 2006 6th International Conference on Data Mining*, pp. 244–253. doi:10.1109/ICDM.2006.122.

Haiminen, N., A. Gionis, and K. Laasonen [2008]. Algorithms for unimodal segmentation with applications to unimodality detection. *Knowledge and Information Systems* **14** (1), 39–57. doi:10.1007/s10115-006-0053-3.

Haskell, W., I. Lee, R. Pate, K. Powell, S. Blair, B. Franklin, C. Macera, G. Heath, P. Thompson, and A. Bauman [2007]. Physical activity and public health: Updated recommendation for adults from the American college of sports medicine and the American heart association. *Circulation* **116** (9), 1081–1093. doi:10.1161/CIRCULATION.107.185649.

Hongeng, S., F. Bremond, and R. Nevatia [2000]. Representation and optimal recognition of human activities. In *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 818–825. IEEE. doi:10.1109/CVPR.2000.855905.

Huang, Y., H. Zheng, C. Nugent, P. McCullagh, S. McDonough, M. Tully, and S. Connor [2010]. Activity monitoring using an intelligent mobile phone: A validation study. In *Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments*. ACM. doi:10.1145/1839294.1839306.

Ifeachor, E. and B. Jervis [2002]. *Digital Signal Processing: A Practical Approach*. Pearson Education.

Iwama, H., D. Muramatsu, Y. Makihara, and Y. Yagi [2012]. Gait-based person-verification system for forensics. In *IEEE 5th International Conference on Biometrics: Theory, Applications and Systems*. doi:10.1109/BTAS.2012.6374565.

Jones, N. and P. Pevzner [2004]. *An Introduction to Bioinformatics Algorithms*. The MIT Press.

Kaghyan, S. and H. Sarukhanyan [2012]. Activity recognition using k-nearest neighbor algorithm on smartphone with tri-axial accelerometer. *International Journal of Information Models and Analyses* **1**, 146–156.

Karaman, S., J. Benois-Pineau, V. Dovgalecs, R. Mégret, J. Pinquier, R. André-Obrecht, Y. Gaëstel, and J. Dartigues [2014]. Hierarchical hidden Markov model in detecting activities of daily living in wearable videos for studies of dementia. *Multimedia Tools and Applications* **69** (3), 743–771. doi:10.1007/s11042-012-1117-x.

Ke, S., H. L. U. Thuc, Y. Lee, J. Hwang, J. Yoo, and K. Choi [2013]. A review on video-based human activity recognition. *Computers* **2** (2), 88–131. doi:10.3390/computers2020088.

Kehagias, A., E. Nidelkou, and V. Petridis [2006]. A dynamic programming segmentation procedure for hydrological and environmental time series. *Stochastic Environmental Research and Risk Assessment* **1** (2), 77–94. doi:10.1007/s00477-005-0013-6.

Keijsers, G. P. J., M. Kampman, and C. A. L. Hoogduin [2001]. Dropout prediction in cognitive behavior therapy for panic disorder. *Behavior Therapy* **32** (4), 739–749. doi:10.1016/S0005-7894(01)80018-6.

Kelley, R., A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis [2008]. Understanding human intentions via hidden Markov models in autonomous mobile robots. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction*, pp. 367–374. ACM.

doi:10.1145/1349822.1349870.

Kelly, M. E., D. Loughrey, B. A. Lawlor, I. H. Robertson, C. Walsh, and S. Brennan [2014]. The impact of exercise on the cognitive functioning of healthy older adults: A systematic review and meta-analysis. *Ageing Research Reviews* **16**, 12–31. doi:10.1016/j.arr.2014.05.002.

Keogh, E., S. Chu, D. Hart, and M. Pazzani [2001]. An online algorithm for segmenting time series. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pp. 289–296. doi:10.1109/ICDM.2001.989531.

Kocielnik, R., N. Sidorova, F. M. Maggi, M. Ouwerkerk, and J. H. D. M. Westerink [2015]. Smart technologies for long-term stress monitoring at work. In *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, pp. 53–58. IEEE. doi:10.1109/CBMS.2013.6627764.

Konig, A., C. F. Crispim-Junior, A. Derreumaux, G. Bensadoun, P. D. Petit, F. Bremond, R. David, F. Verhey, P. Aalten, and P. Robert [2015]. Validation of an automatic video monitoring system for the detection of instrumental activities of daily living in dementia patients. *Journal of Alzheimer's Disease* **44** (2), 675–685. doi:10.3233/JAD-141767.

Kotsiantis, S. B., C. J. Pierrakeas, and P. E. Pintelas [2003]. Preventing student dropout in distance learning using machine learning techniques. In V. Palade, R. J. Howlett, and L. Jain (Eds.), *Knowledge-Based Intelligent Information and Engineering Systems*, Volume 2774 of *Lecture Notes in Computer Science*, pp. 267–274. Springer. doi:10.1007/978-3-540-45226-3_37.

Koza, J. R. [2010]. Introduction to genetic programming tutorial: From the basics to human-competitive results. In *GECCO 2010: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, pp. 2137–2262. ACM. doi:10.1145/1830761.1830894.

Ladetto, Q. [2000]. On foot navigation: Continuous step calibration using both complementary recursive prediction and adaptive Kalman filtering. In *Proceedings of the 13th International Technical Meeting of the Satellite Division of the Institute of Navigation*, pp. 1735–1740.

Lara, O. and M. Labrador [2013]. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys & Tutorials* **15** (3), 1192–1209. doi:10.1109/SURV.2012.110112.00192.

Lee, S. and K. Mase [2002]. Activity and location recognition using wearable sensors. *Pervasive Computing* **1** (3), 24–32. doi:10.1109/MPRV.2002.1037719.

Li, F., C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao [2012]. A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 421–430. ACM. doi:10.1145/2370216.2370280.

Ligorio, G. and A. M. Sabatini [2015]. A novel Kalman filter for human motion tracking with an inertial-based dynamic inclinometer. *IEEE Transactions on Biomedical Engineering* **62** (8), 2033–2043. doi:10.1109/TBME.2015.2411431.

Lim, S. S., T. Vos, A. D. Flaxman, G. Danaei, K. Shibuya, H. Adair-Rohani, M. A. AlMazroa, M. Amann, H. R. Anderson, K. G. Andrews, M. Aryee, C. Atkinson, L. J. Bacchus, A. N. Bahilim, K. Balakrisnan, J. Balmes, S. Barker-Collo, A. Baxter, M. L. Bell, J. D. Blore, F. Blyth, C. Bonner, G. Borges, R. Bourne, M. Boussinesq, M. Brauer, P. Brooks, N. G. Bruce, B. Brunekreef, C. Bryan-Hancock, C. Bucello, R. Buchbinder, F. Bull, R. T. Burnett, T. E. Byers, B. Calabria, J. Carapetis, E. Carnahan, Z. Chafe, F. Charlson, H. Chen, J. S. Chen, A. T. Cheng, J. C. Child, A. Cohen, K. E. Colson, B. C. Cowie, S. Darby, S. Darling, and A. Davis [2010]. A comparative risk assessment of burden of disease and injury attributable to 67 risk factors and risk factor clusters in 21 regions 1990-2010: A systematic analysis for the global burden of disease study 2010. *The Lancet* **380** (9859), 2224–2260. doi:10.1016/S0140-6736(12)61766-8.

Lippmann, R. [1987]. An introduction to computing with neural nets. *IEEE ASSP Magazine* **4** (2), 4–22. doi:10.1109/MASSP.1987.1165576.

Logan, B., J. Healey, M. Philipose, E. Tapia, and S. Intille [2007]. A long-term evaluation of sensing modalities for activity recognition. In J. Krumm, G. Abowd, A. Seneviratne, and T. Strang (Eds.), *UbiComp 2007: Ubiquitous Computing*, Volume 4717 of *Lecture Notes in Computer Science*, pp. 483–500. Springer. doi:10.1007/978-3-540-74853-3_28.

Lombardo, L., M. Cama, C. Conoscenti, M. Märker, and E. Rotigliano [2015]. Binary logistic regression versus stochastic gradient boosted decision trees in assessing landslide susceptibility for multiple-occurring landslide events: Application to the 2009 storm event in Messina (Sicily, southern Italy). *Natural Hazards* **79** (3), 1621–1648. doi:10.1007/s11069-015-1915-3.

Long, X., S. Pauws, M. J. Pijl, J. Lacroix, A. H. C. Goris, and R. M. Aarts [2009]. Analysis and prediction of daily physical activity level data using autoregressive integrated moving average models. In *Proceedings of the 3rd Workshop Behaviour Monitoring and Interpretation*, pp. 1–15.

Long, X., S. Pauws, M. J. Pijl, J. Lacroix, A. H. C. Goris, and R. M. Aarts [2011]. Predicting daily physical activity in a lifestyle intervention program. In B. Gottfried and H. Aghajan (Eds.), *Behaviour Monitoring and Interpretation – BMI*, Volume 9 of *Ambient Intelligence and Smart Environments*, pp. 131–146. IOS Press. doi:10.3233/978-1-60750-731-4-131.

Long, X., M. J. Pijl, S. Pauws, J. Lacroix, A. H. C. Goris, and R. M. Aarts [2014]. Towards tailored physical activity health intervention: Predicting dropout par-

ticipants. *Health and Technology* **4** (3), 273–287. doi:10.1007/s12553-014-0084-9.

Lovric, M., M. Milanovic, and M. Stamenkovic [2014]. Algorithmic methods for segmentation of time series: An overview. *Journal of Contemporary Economic and Business Issues* **1** (1), 31–53.

Luštrek, M. and B. Kaluža [2009]. Fall detection and activity recognition with machine learning. *Informatica* **33** (2), 197–204.

Lykourentzou, I., I. Giannoukos, V. Nikolopoulos, G. Mpardis, and V. Loumos [2009]. Dropout prediction in e-learning courses through the combination of machine learning techniques. *Computers & Education* **53** (3), 950–965. doi:10.1016/j.compedu.2009.05.010.

Madabhushi, A. and J. K. Aggarwal [1999]. A Bayesian approach to human activity recognition. In *Second IEEE Workshop on Visual Surveillance*, pp. 25–32. IEEE. doi:10.1109/VS.1999.780265.

Mamitsuka, H. [1997]. Supervised learning of hidden Markov models for sequence discrimination. In *Proceedings of the 1st Annual International Conference on Computational Molecular Biology*, RECOMB 1997, pp. 202–208. ACM. doi:10.1145/267521.267551.

Mannila, H., H. Toivonen, and A. I. Verkamo [1997]. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery* **1** (3), 259–289. doi:10.1023/A:1009748302351.

Mannini, A. and A. Sabatini [2012]. Gait phase detection and discrimination between walking-jogging activities using hidden Markov models applied to foot motion data from a gyroscope. *Gait & Posture* **36** (4), 657–661. doi:10.1016/j.gaitpost.2012.06.017.

Maquet, D., F. Lekeu, E. Warzee, S. Gillain, V. Wojtasik, E. Salmon, J. Petermans, and J. L. Croisier [2010]. Gait analysis in elderly adult patients with mild cognitive impairment and patients with mild Alzheimer's disease: Simple versus dual task: A preliminary report. *Clinical Physiology and Functional Imaging* **30** (1), 51–56. doi:10.1111/j.1475-097X.2009.00903.x.

Márquez-Vera, C., A. Cano, C. Romero, and S. Ventura [2013]. Predicting student failure at school using genetic programming and different data mining approaches with high dimensional and imbalanced data. *Applied Intelligence* **38** (3), 315–330. doi:10.1007/s10489-012-0374-8.

Marschollek, M., M. Goevercin, K. Wolf, B. Song, M. Gietzelt, R. Haux, and E. Steinhagen-Thiessen [2008]. A performance comparison of accelerometry-based step detection algorithms on a large, non-laboratory sample of healthy and mobility-impaired persons. In *30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE.

doi:10.1109/IEMBS.2008.4649407.

Mathers, C. D., G. A. Stevens, T. Boerma, R. A. White, and M. I. Tobias [2015]. Causes of international increases in older age life expectancy. *The Lancet* **385** (9967), 540–548. doi:10.1016/S0140-6736(14)60569-9.

Maurer, U., A. Smailagic, D. P. Siewiorek, and M. Deisher [2006]. Activity recognition and monitoring using multiple sensors on different body positions. In *International Workshop on Wearable and Implantable Body Sensor Networks*, pp. 113–116. IEEE. doi:10.1109/BSN.2006.6.

McCallum, A. and K. Nigam [1998]. A comparison of event models for naive Bayes text classification. In *Workshop on Learning for Text Categorization*, pp. 41–48. doi:10.1.1.65.9324.

McNamee, A. [2005]. *Ethical Issues arising from the Real Time Tracking and Monitoring of People using GPS-Based Location Services*. Technical report, University of Wollongong.

Megiddo, N. and K. J. Supowit [1984]. On the complexity of some common geometric location problems. *SIAM Journal on Computing* **13** (1), 182–196. doi:10.1137/0213014.

Megiddo, N., E. Zemel, and S. L. Hakimi [1981]. The maximum coverage location problem. *SIAM Journal on Algebraic Discrete Methods* **4** (2), 253–261. doi:10.1137/0604028.

Meinhold, R. J. and N. D. Singpurwalla [1983]. Understanding the Kalman filter. *The American Statistician* **37** (2), 123–127.

Mezghani, N., S. Husse, K. Boivin, K. Turcot, R. Aissaoui, N. Hagemeister, and J. de Guise [2008]. Automatic classification of asymptomatic and osteoarthritis knee gait patterns using kinematic data features and the nearest neighbor classifier. *IEEE ASSP Magazine* **55** (3), 1230–1232. doi:10.1109/TBME.2007.905388.

Michael, K., A. McNamee, and M. Michael [2006]. The emerging ethics of humancentric GPS tracking and monitoring. In *International Conference on Mobile Business*, pp. 34. doi:10.1109/ICMB.2006.43.

Mitchell, E., D. Monaghan, and N. E. O'Connor [2013]. Classification of sporting activities using smartphone accelerometers. *Sensors* **13** (4), 5317–5337. doi:10.3390/s130405317.

Mladenov, M. and M. Mock [2009]. A step counter service for java-enabled devices using a built-in accelerometer. In *Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware*, pp. 1–5. ACM. doi:10.1145/1554233.1554235.

Muller, M. [2007]. Dynamic time warping. In *Information Retrieval for Music and*

Motion. Springer Berlin Heidelberg. doi:10.1007/978-3-540-74048-3_4.

Must, A., J. Spadano, E. H. Coakley, E. A. Field, G. Colditz, and W. H. Dietz [1999]. The disease burden associated with overweight and obesity. *The Journal of the American Medical Association* **282**(16), 1523–1529. doi:10.1001/jama.282.16.1523.

Nuynh, T. and B. Schiele [2005]. Analyzing features for activity recognition. In *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies*, pp. 159–163. ACM. doi:10.1145/1107548.1107591.

Okour, S., A. Maender, and J. Basilakis [2015]. An adaptive rule-based approach to classifying activities of daily living. In *International Conference on Healthcare Informatics*, pp. 404–407. IEEE. doi:10.1109/ICHI.2015.57.

Oliver, N., E. Horvitz, and A. Garg [2002]. Layered representations for human activity recognition. In *4th IEEE International Conference on Multimodal Interfaces*, pp. 3–8. IEEE. doi:10.1109/ICMI.2002.1166960.

Oner, M., J. Pulcifer-Stump, P. Seeling, and T. Kaya [2012]. Towards the run and walk activity classification through step detection - An Android application. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. doi:10.1109/EMBC.2012.6346344.

op den Buijs, J., T. Smits, M. Pijl, M. Simons, and L. Schertzer [2015]. Predictive modeling of emergency hospital transport using medical alert pattern data: Retrospective cohort study. In *iproc*, Volume 1 of *Connected Health Symposium*, pp. e19. JMIR Publications. doi:10.2196/iproc.4772.

Pan, J. and W. J. Tompkins [1985]. A real-time QRS detection algorithm. *IEEE Transactions on Biomedical Engineering* **32**(3), 230–236. doi:10.1109/TBME.1985.325532.

Pantelopoulos, A. and N. Bourbakis [2009]. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **40**(1), 1–12. doi:10.1109/TSMCC.2009.2032660.

Patel, S., H. Park, P. Bonato, L. Chan, and M. Rodgers [2012]. A review of wearable sensors and systems with application in rehabilitation. *Journal of NeuroEngineering and Rehabilitation* **9**(21). doi:10.1186/1743-0003-9-21.

Patterson, D. J., D. Fox, H. Krautz, and M. Philipose [2005]. Fine-grained activity recognition by aggregating abstract object usage. In *Proceedings of the 9th IEEE International Symposium on Wearable Computers*, pp. 44–51. IEEE. doi:10.1109/ISWC.2005.22.

Perera, C., A. Zaslavsky, and P. C. D. Georgakopoulos [2014]. Context aware computing for the Internet of Things: A survey. *IEEE Communications Surveys &*

Tutorials **16** (1), 414–454. doi:10.1109/SURV.2013.042313.00197.

Philipose, M., K. P. Fishkin, M. Perkowitz, and D. J. Patterson [2004]. Inferring activities from interactions with objects. *Pervasive Computing* **3** (4), 50–57. doi:10.1109/MPRV.2004.7.

Pijl, M., J. Lacroix, S. Pauws, and A. Goris [2009]. Prediction of successful participation in a lifestyle activity program using data mining techniques. In *21st Benelux Conference on Artificial Intelligence*.

Pijl, M., S. van de Par, and C. Shan [2010]. An event-based approach to multi-modal activity modeling and recognition. In *IEEE International Conference on Pervasive Computing and Communications*, pp. 98–106. IEEE. doi:10.1109/PERCOM.2010.5466986.

Pijl, M. J. [2015]. Method of estimating the position of a device and an apparatus implementing the same. U.S. Patent Application 20150173037.

Pijl, M. J. and H. Baldus [2016]. Method and apparatus for determining the risk of a patient leaving a safe area. U.S. Patent Application 20160113591.

Pijl, M. J., P. M. Fulton, and H. Baldus [2015]. Method of controlling a device implementing the same. U.S. Patent Application 20150087332.

Pijl, M. J., C. Shan, L. Wang, and S. L. J. D. E. van de Par [2014]. Method of selecting an optimal viewing angle position for a camera. U.S. Patent 8659655.

Pijl, M. J. and T. Smits [2016]. A comparison of accelerometer-based cadence estimation algorithms at different walking speeds. Submitted to Gait & Posture.

Poularakis, S., K. Avgerinakis, A. Briassouli, and I. Kompatsiaris [2015]. Computationally efficient recognition of activities of daily living. In *IEEE International Conference on Image Processing*, pp. 247–251. IEEE. doi:10.1109/ICIP.2015.7350797.

Preis, J., M. Kessel, M. Werner, and C. Linnhoff-Popien [2012]. Gait recognition with Kinect. In *1st International Workshop on Kinect in Pervasive Computing*, pp. 1–4.

Rabiner, L. R. [1989]. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77** (2), 257–286. doi:10.1109/5.18626.

Rashidi, P. and D. J. Cook [2009]. Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* **39** (5), 949–959. doi:10.1109/TSMCA.2009.2025137.

Ravi, N., D. Nikhil, P. Mysore, and M. L. Littman [2005]. Activity recognition from accelerometer data. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence*, pp. 1541–1546. AAAI Press.

Ronao, C. A. and S. Cho [2016]. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications* **59**, 235–244. doi:10.1016/j.eswa.2016.04.032.

Ryan, C. G., P. M. Grant, W. W. Tigbe, and M. H. Granat [2006]. The validity and reliability of a novel activity monitor as a measure of walking. *British Journal of Sports Medicine* **40** (9), 779–784. doi:10.1136/bjsm.2006.027276.

Ryberg, C., E. Rostrup, M. B. Stegmann, F. Barkhof, P. Scheltens, E. C. W. van Straaten, F. Fazekas, R. Schmidt, J. M. Ferro, H. Baezner, T. Erkinjuntti, H. Jokinen, L. O. Wahlund, J. O'Brien, A. M. Basile, L. Pantoni, D. Inzitari, and G. Waldemar [2007]. Clinical significance of corpus callosum atrophy in a mixed elderly population. *Neurobiology of Aging* **28** (6), 955–963. doi:10.1016/j.neurobiolaging.2006.04.008.

Ryu, U., K. Ahn, E. Kim, M. Kim, Y. Chang, B. Kim, and S. Woo [2013]. Adaptive step detection algorithm for wireless smart step counter. In *International Conference on Information Science and Applications*, pp. 1–4. IEEE. doi:10.1109/ICISA.2013.6579332.

Saar-Tsechansky, M. and F. Provost [2007]. Handling missing values when applying classification models. *Journal of Machine Learning Research* **8**, 1623–1657.

Scherder, E., L. Eggermont, D. Swaab, M. v. Heuvelen, Y. Kamsma, M. d. Greef, R. v. Wijck, and T. Mulder [2007]. Gait in ageing and associated dementias; Its relationship with cognition. *Neuroscience & Biobehavioral Reviews* **31** (4), 485–497. doi:10.1016/j.neubiorev.2006.11.007.

Seelig, H. and R. Fuchs [2011]. Physical exercise participation: A continuous or categorical phenomenon? *Psychology of Sport and Exercise* **12** (2), 115–123. doi:10.1016/j.psychsport.2010.10.004.

Shaeffer, D. [2013]. MEMS inertial sensors: A tutorial overview. *IEEE Communications Magazine* **51** (4), 100–109. doi:10.1109/MCOM.2013.6495768.

Sheikh, Y., M. Sheikh, and M. Shah [2005]. Exploring the space of an action for human action recognition. In *10th IEEE International Conference on Computer Vision*, pp. 144–149. IEEE. doi:10.1109/ICCV.2005.90.

Shinkai, S., S. Watanabe, S. Kumagai, Y. Fujiwara, H. Amano, H. Yoshida, T. Ishizaki, H. Yukawa, T. Suzuki, and H. Shibata [2000]. Walking speed as a good predictor for the onset of functional dependence in a Japanese rural community population. *Age and Ageing* **29** (5), 441–446. doi:10.1093/ageing/29.5.441.

Sorbral, A. and A. Vacavant [2014]. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding* **122**, 4–21. doi:10.1016/j.cviu.2013.12.005.

Stefanov, D. [2004]. The smart house for older persons and persons with physical disabilities: Structure, technology arrangements, and perspectives. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* **12** (2), 288–250. doi:10.1109/TNSRE.2004.828423.

Stergiou, N. and L. M. Decker [2011]. Human movement variability, nonlinear dynamics, and pathology: Is there a connection? *Human Movement Science* **30** (5), 869–888. doi:10.1016/j.humov.2011.06.002.

Stevens, G. A., G. M. Singh, Y. Lu, G. Danaei, J. K. Lin, M. M. Finucane, A. N. Bahalim, R. K. McIntire, H. R. Gutierrez, M. Cowan, C. J. Paciorek, F. Farzadfar, L. Riley, and M. Ezzati [2012]. National, regional, and global trends in adult overweight and obesity prevalences. *Population Health Metrics* **10** (22). doi:10.1186/1478-7954-10-22.

Storm, F. A., B. W. Heller, and C. Mazzà [2015]. Step detection and activity recognition accuracy of seven physical activity monitors. *PLoS ONE* **10** (3), e0118723. doi:10.1371/journal.pone.0118723.

Storti, K. L., K. K. Pettee, J. S. Brach, J. B. Talkowski, C. R. Richardson, and A. M. Kriska [2008]. Gait speed and step-count monitor accuracy in community-dwelling older adults. *Medicine & Science in Sports & Exercise* **40** (1), 59–64. doi:10.1249/mss.0b013e318158b504.

Suryadevara, N. and S. Mukhopadhyay [2012]. Wireless sensor network based home monitoring system for wellness determination of elderly. *IEEE Sensors Journal* **12** (6), 1965–1972. doi:10.1109/JSEN.2011.2182341.

Tao, W., T. Liu, R. Zheng, and H. Feng [2012]. Gait analysis using wearable sensors. *Sensors* **12** (2), 2255–2283. doi:10.3390/s120202255.

Tatti, N. [2013]. Fast sequence segmentation using log-linear models. *Data Mining and Knowledge Discovery* **27** (3), 421–441. doi:10.1007/s10618-012-0301-y.

Terzi, E. [2006]. *Problems and Algorithms for Sequence Segmentations*. Technical report, Department of Computer Science, Helsinki University.

Terzi, E. and P. Tsaparas [2006]. Efficient algorithms for sequence segmentation. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pp. 316–327. doi:10.1137/1.9781611972764.28.

Thompson, P. D. [2003]. Exercise and physical activity in the prevention and treatment of atherosclerotic cardiovascular disease. *Arteriosclerosis, Thrombosis, and Vascular Biology* **23** (8), 1319–1321. doi:10.1161/01.ATV.0000087143.33998.F2.

Torrence, C. and G. P. Compo [1998]. A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society* **79** (1), 61–78. doi:10.1175/1520-0477(1998)079<0061:APGTWA>2.0.CO;2.

Toshev, A. and C. Szegedy [2014]. DeepPose: Human pose estimation

via deep neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653–1660. IEEE. doi:10.1109/CVPR.2014.214.

Tran, T., W. Luo, D. Phung, S. Gupta, S. Rana, R. L. Kennedy, A. Larkins, and S. Venkatesh [2014]. A framework for feature extraction from hospital medical data with applications in risk prediction. *BMC Bioinformatics* **15** (425). doi:10.1186/s12859-014-0425-8.

Tudor-Locke, C. and D. Rowe [2012]. Using cadence to study free-living ambulatory behavior. *Sports Medicine* **42** (5), 381–398. doi:10.2165/11599170-000000000-00000.

Urwyler, P., L. Rampa, R. Stucki, M. Büchler, R. Müri, U. P. Mosimann, and T. Nef [2015]. Recognition of activities of daily living in healthy subjects using two ad-hoc classifiers. *BioMedical Engineering OnLine* **14** (54). doi:10.1186/s12938-015-0050-4.

Valle, M. A., S. Varas, and G. A. Ruz [2012]. Job performance prediction in a call center using a naive Bayes classifier. *Expert Systems with Applications* **39** (11), 9939–9945. doi:10.1016/j.eswa.2011.11.126.

van der Maaten, L., E. Postma, and J. van den Herik [2009]. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research* **10**, 66–71.

van Halteren, A. T., J. P. W. Lacroix, G. Gelijnse, M. J. Pijl, P. K. Saini, M. C. Kaptein, J. L. G. Ferron, and R. Holmes [2014]. Coaching system that builds coaching messages for physical activity promotion. U.S. Patent Application 20140122104.

van Iersel, M. B., W. Hoefsloot, M. Munneke, B. R. Bloem, and M. G. M. O. Rikkert [2004]. Systematic review of quantitative clinical gait analysis in patients with dementia. *Zeitschrift für Gerontologie und Geriatrie* **37** (1), 27–32. doi:10.1007/s00391-004-0176-7.

van Kasteren, T., A. Noulas, G. Englebienne, and B. Kröse [2008]. Accurate activity recognition in a home setting. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, pp. 1–9. ACM. doi:10.1145/1409635.1409637.

Verghese, J., M. Robbins, R. Holtzer, M. Zimmerman, C. Wang, X. Xue, and R. B. Lipton [2008]. Gait dysfunction in mild cognitive impairment syndromes. *Journal of the American Geriatrics Society* **56** (7), 1244–1251. doi:10.1111/j.1532-5415.2008.01758.x.

Viard, K., M. Fanti, G. Faraut, and J. Lesage [2016]. An event-based approach for discovering activities of daily living by hidden Markov models. In *15th IEEE International Conference on Ubiquitous Computing and Communica-*

tions. IEEE.

Vögele, A., B. Krüger, and R. Klein [2014]. Efficient unsupervised temporal segmentation of human motion. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 167–176.

Waite, L. M., D. A. Grayson, O. Piguet, H. Creasey, H. P. Bennett, and G. A. Broe [2005]. Gait slowing as a predictor of incident dementia: 6-year longitudinal data from the Sydney Older Persons Study. *Journal of the Neurological Sciences* **229–230**, 89–93. doi:10.1016/j.jns.2004.11.009.

Warburton, D., C. Nicol, and S. Bredin [2006]. Health benefits of physical activity: The evidence. *Canadian Medical Association Journal* **174** (6), 801–809. doi:10.1503/cmaj.051351.

Ware, L. J., R. Hurling, O. B. W. B. Fairley, L. T. Hurst, P. Murray, L. K. Rennie, E. C. Tomkins, A. Finn, R. M. Cobain, A. D. Pearson, and P. J. Foreyt [2008]. Rates and determinants of uptake and use of an internet physical activity and weight management program in office and manufacturing work sites in England: Cohort study. *Journal of Medical Internet Research* **10** (4), e56. doi:10.2196/jmir.1108.

Welch, G. and G. Bishop [1995]. *An Introduction to the Kalman Filter*. Technical report, University of North Carolina at Chapel Hill.

WHO [2011]. *Global Health and Ageing*. Technical report, World Health Organization, US National Institute of Aging.

Wichit, N. and A. Choksuriwong [2015]. Multi-sensor data fusion model based Kalman filter using fuzzy logic for human activity detection. *International Journal of Information and Electronics Engineering* **5** (6), 450–453. doi:10.7763/IJIEE.2015.V5.577.

Wiederhold, B., G. Riva, and G. Graffigna [2013]. Ensuring the best care for our increasing aging population: Health engagement and positive technology can help patients achieve a more active role in future healthcare. *Cyberpsychology, Behavior and Social Networking* **16** (6), 411–412. doi:10.1089/cyber.2013.1520.

Wittwer, J. E., K. E. Webster, and H. B. Menz [2010]. A longitudinal study of measures of walking in people with Alzheimer's disease. *Gait & Posture* **32** (1), 113–117. doi:10.1016/j.gaitpost.2010.04.001.

Yang, J. and V. Honavar [1998]. Feature subset selection using a genetic algorithm. In H. Liu and H. Motoda (Eds.), *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Volume 453 of *The Springer International Series in Engineering and Computer Science*, pp. 117–136. Springer US. doi:10.1007/978-1-4615-5725-8_8.

Ye, J., S. Dobson, and S. McKeever [2012]. Situation identification techniques in

pervasive computing: A review. *Pervasive and Mobile Computing* **8** (1), 36–66. doi:10.1016/j.pmcj.2011.01.004.

Ying, H., C. Silex, A. Schnitzer, S. Leonhardt, and M. Schiek [2007]. Automatic step detection in the accelerometer signal. In *Proceedings of the 4th International Workshop on Wearable and Implantable Body Sensor Networks*, pp. 80–85. Springer. doi:10.1007/978-3-540-70994-7_14.

Zhang, Y., K. G. M. Beenakker, P. M. Butala, C. Lin, T. D. C. Little, A. B. Maier, M. Stijntjes, R. Vartanian, and R. C. Wagenaar [2012]. Monitoring walking and cycling of middle-aged to older community dwellers using wireless wearable accelerometers. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 158–161. IEEE. doi:10.1109/EMBC.2012.6345895.

Zijlstra, W. and A. L. Hof [2003]. Assessment of spatio-temporal gait parameters from trunk accelerations during human walking. *Gait & Posture* **18** (2), 1–10. doi:10.1016/S0966-6362(02)00190-X.

Acknowledgements

Although somewhat cliché, it is nonetheless true that this thesis would not have become a reality without the large group of people who have guided, supported, and inspired me along the way.

First and foremost, I would like to thank my copromotor and day-to-day coordinator Jan Korst, and my promotors Emile Aarts and Max Louwerse. Thank all of you for challenging, encouraging and motivating me to achieve the results that can be found in this book.

Jan, I cannot thank you enough for the untold amounts of hours you have dedicated, the innumerable times you critically went through my writings, and for all the feedback and guidance you have given me over the years. I would say that I will miss our weekly talks and discussions on work and also topics of a more random nature, but if we are ever going to solve that k,h-segmentation problem, we're probably going to need a lot more coffee...

Emile, thank you for your energy and enthusiasm, from our first meeting to the final print of this thesis, never dulled despite gradually increasing geographical separation. Throughout that time, your ability to instantly grasp a topic and condense it into its essence has never ceased to amaze me.

Max, even though you joined at a later stage of the journey, your enthusiasm, ideas and feedback have been invaluable. Thank you for providing a new and unique perspective to this work that made this thesis much better than it otherwise would have been.

I would also like to extend my gratitude to the members of the reading committee: Catholijn Jonker, Johan Lukkien, Wim van Petegem, Eric Postma, and Boris de Ruyter for reviewing this thesis, and for challenging me to further improve the contents of this work. In the same vein, I would like to thank Verus Pronk for reviewing the thesis, and Ramon Clout for reviewing the contents of Chapter 2. Thank you for your comments, and in particular, for painstakingly checking all my mathematical equations.

I would also like to thank everyone I worked with at Philips Research. In particular, but by no means exclusively, my fellow project members who worked with me on the topics described in this thesis: Steven van de Par, Caifeng Shan, Pavan Dadlani, Lu Wang, Boris de Ruyter, Steffen Pauws, Xi Long, Joyca Lacroix, Privender Saini, Roger Holmes, Annelies Goris, Tine Smits, Paul Fulton, Alan Davie, Patrick Kechichian, Laura Klaming, and Heribert Baldus.

In addition, I want to thank everyone at Philips Research with whom I had the pleasure to work over the years, whether as a project member or otherwise, as well as those at Tilburg University who helped during the final stages of this thesis. Further, I would like to thank all those who participated in the studies described in this thesis; the tracking of human motion would have been very difficult without you.

Finally, I would like to thank my friends and family, particularly Sip Jan, Agnes, Steven and Esther, for supporting and encouraging me, and distracting me when needed.