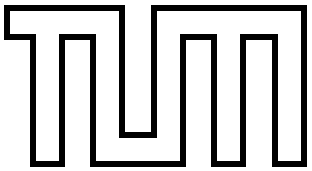


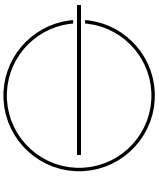
FAKULTÄT FÜR INFORMATIK

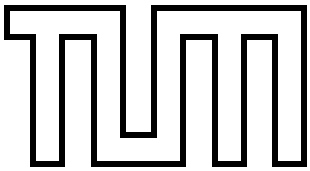
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

# Determining Body-Orientation from Sensors in Mobile Devices

Philip Daubmeier





FAKULTÄT FÜR INFORMATIK

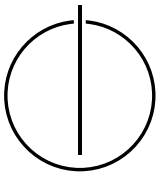
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

# Determining Body-Orientation from Sensors in Mobile Devices

# Gewinnung von Daten zur Körperorientierung aus Sensoren in Mobilgeräten


| | |
|---|---|
| Author: | Philip Daubmeier |
| Supervisor: | Prof. Dr. Johann Schlichter |
| Advisor: | Dr. Georg Groh |
| Date: | July 16, 2012 |

I assure the single handed composition of this master's thesis only supported by declared resources.

München, July 16, 2012 Philip Daubmeier

> The hardest thing is to go to sleep at night, when there are so many urgent things needing to be done. A huge gap exists between what we know is possible with today's machines and what we have so far been able to finish.
> — *Donald Knuth, author of "The Art of Computer Programming"*

## Acknowledgements

Tina, thank you so much for your patience with me during all the time and for constantly motivating me. Thank you, I love you!

Thanks a lot to my fellow students and my parents for proofreading the work and giving me helpful suggestions for improving the composition.

Thanks to all volunteers for their participation in the experiment to collect training data. Also, I want to thank all interviewees of the survey, discussed in section 6.1, for taking the time to answer all the questions.

Niklas and René, thanks for your readiness to help me with the Kinect SDK and its restrictions.

Alex, thank you. I know we didn't meet very often, but you helped me more than you might think. Your diploma thesis was a great assistance and inspiration for me.

Georg, special thanks to you as my valued advisor and mentor in many ways. Thank you for your support, all the ideas and particularly for the inspirational discussions we had. You gave me the chance to actively develop the topic of this thesis together with you and the freedom of making the work turn into what it is now. Having started a few months back, several restrictions and problems occured over and over, some of which I could and some of which I couldn't influence or work around. In the end, however, with steady effort in improving the method with your help, some of the results were even better than we expected.

Most importantly, I want to thank my supervisor Professor Schlichter, for giving me the opportuniy to realize this work. With the need to apply knowledge from all sorts of areas inside the domain of computer science, some of which I had to study especially for this work, I could not think of a more exciting and challenging subject for my master's thesis. I personally learned a lot during this time and hope that I could —to some degree— contribute to this area of research.

# Abstract

This thesis presents a method to extract body geometry information of a person by solely using sensor data of a mobile phone, carried in the user's trouser pocket. This is achieved by utilising a fitted regression model and adequate preprocessing, exploiting domain knowledge about the sensors' properties and constraints of human anatomy. The model is constructed from a training data set of simultaneously measured inertial sensor values and skeleton data, which is in turn gained from a depth-sensing camera. It is examined, if the presented method can sufficiently model the underlying relation in order to be able to reconstruct important parts of the body geometry from phone sensor readings alone. Finally, the precision of those predictions is shown in an empirical analysis, before discussing the opportunities which arise from this work.

# Contents

## Acronyms

| | |
|---|---|
| CMOS | Complementary Metal Oxide Semiconductor |
| DFT | Discrete Fourier Transformation |
| DoF | Degrees of Freedom |
| FFT | Fast Fourier Transformation |
| ID | Identifier |
| IMU | Inertial Measurement Unit |
| IR | Infra-Red (light spectrum) |
| Lerp | Linear interpolation |
| MEMS | Micro-Electro-Mechanical System |
| MSN | Mobile Social Networking |
| NED | North-East-Down (coordinate system) |
| NFC | Near Field Communication |
| NUI | Natural User Interface |
| RGB | Red-Green-Blue (colour space) |
| RSS | Residual Sum of Squares |
| SDK | Software Development Kit |
| Slerp | Spherical Linear Interpolation |
| SN | Social Networking |
| Squad | Spherical cubic interpolation |
| TCP | Transmission Control Protocol |
| ToF | Time of Flight |

| | |
|---|---|
| USB | Universal Serial Bus |
| WiFi | Wireless network |

# Introduction

In the scale of history of mankind, computers are one of the most recent inventions. Yet, they have undergone a rapid development, from large room-filling calculators to highly integrated micro devices in the more recent past.

With the advent of the internet, people started to use them for communication and social interaction. Web-based social networking platforms emerged, which enabled users to connect by explicitly establishing friendship relations to each other. The machines became part of the social life. Moreover, structures of social interaction were mapped in a machine-processable form by people themselves, which in turn enabled new types of services.

Efforts were made to generate such structures automatically. Those attempts are slowly maturing, being supported by the fact that mobile devices are getting more and more powerful as well as being equipped with a broad range of sensors. However, social signals, which convey essential information about the social relationship, are complex and yet subtle. They are the sum of non-verbal behavioural cues like gesture, posture, interpersonal distance, vocal behaviour and the like [VSP09]. Humans are perfectly able to either subconsciously or deliberately perceive those cues and interpret them, while machines still struggle with such tasks.

While such social signals comprise details about the kind of relationship, it is essential to know between whom those relations exist in the first place. Therefore, the task is to observe which social interactions take place between people. The research field of Reality Mining takes the approach of collecting little digital bread-crumbs, which are left by people's everyday activities, over longer periods of time [Pen09]. One key information that can be gained this way is a trace of locations of different persons, that can be used to draw conclusions about where and when people met. Wearable and location-aware devices, like smartphones with their sensors, were utilised for the collection of these data. This is a very unobtrusive way, as people carry such devices anyhow.

Still, these methods often suffer from not being fine-grained enough on small scales. To obtain details about the nature of various social situations, more precise interaction geometry is needed. Sensors of mobile devices provide more information about the users body geometry than their raw data gives reason to expect. It can be extracted by exploiting domain knowledge about human anatomy, wearing preferences and typical patterns of activity.

In this thesis, a method is presented, that makes use of these facts to compute certain body geometry features. A supervised machine learning approach is used to train a mapping from smartphone sensor data to the body posture, as seen from a supervising motion capturing sensor. To make the method reliable, various preprocessing steps are applied to the sensor data. In doing so, again domain knowledge is getting embedded into the system.

Therefore, central goals of this work are the identification of patterns that a preprocessing can capitalise on, as well as the selection of a suitable machine learning algorithm. Fur-

thermore, an answer to the question, which body geometry features can be deduced with sufficient quality, should be found.

The discussion of related work in both the realms of social signal processing and motion capturing establishes the basis for subsequent chapters. Next, all used sensors of both the mobile device and the training supervisor are presented. After this, the selection of the machine learning method is discussed. Within that context, the advantages of special preparation of the raw values is elaborated. The various steps of this preprocessing are then derived on the basis of experiments in the following. The results are evaluated, before concluding with a recapitulation of the method and a discussion of the prospects and opportunities, which arise from this work.

# 1. Concept

Before presenting the approach that was taken in this thesis, an overview over related research topics will be given in the following. First and foremost, social situations are examined with emphasis on social networks and social signal processing, to lay out the foundation upon which the work is built. It will be investigated which parameters of body geometry are especially of interest and promise to be able to be inferred from sensor values. Previous methods are enumerated, which use the same kind of sensors to reason about body geometry and types of activity. A proposed new approach is concluded from the weaknesses that accompany those prior methods. Associated sensors and mechanisms for detecting body orientation are put in the context of related techniques, before going into great detail in the next chapters.

## 1.1. Social Networks

This work is mainly motivated by a growing research interest in bringing social networks and social interactions into a computer science context. Social networks, in their original meaning, are graph structures that model individuals and dyadic connections between them. Dyads are mid- or long-term relationships between actors in such a network, and can incorporate additional details about the type or strength of the connection [MG11]. Social networks and their analysis is therefore an interdisciplinary field of sociology, social psychology and graph theory.

### 1.1.1. Social Networking Platforms

*Web-based Social Networking Platforms* emerged from online communities, enriching them with the addition of a mapping to just such a social network graph. Users of these services are establishing relationships to user profiles of people they know. Depending on the platform, they can even specify the type or weight of their relation, e.g. 'acquaintance', 'close friend', 'family member', etc. This way, the users are explicitly creating a social network graph inside the system of the service provider. Growing public interest in those platforms and their related research topic mutually drive each other. While the former allows for analysis of social networks in a new scale, scientific findings of the latter are flowing back into the development of platforms.

Real-world social networks are much more dynamic, however. For instance, after leaving school or changing the employer, friends or colleagues are no longer met everyday, which causes these dyads to become weaker until possibly completely losing touch to some. With

couples breaking up or falling out with former friends, dyads may stay but get negative connotations. Other examples of these dynamics could be friends in sports, for which the relationship and frequency of encountering may vary seasonally.

All those changes over time and many others are not nearly mirrored in social networking platforms on the web yet. As investigations and surveys of these showed [NBW06, GD10], users are adding new relations on average around three to four times more often than deleting dyads. This may be caused by the majority of users conceiving and using these websites as platforms for staying in contact with former acquaintances. Even with the possibility given to categorize relations, it is questionable whether users would actually explicitly update their profile to keep it consistent with their current real-world social networks. And if they did, those manually updated graphs are likely to be biased and error-prone. This is not surprising, given the nature of humans, who are not perceiving the world from a point of view as neutral as machines are doing.

### 1.1.2. Mobile Social Networking

*Mobile Social Networking* (MSN) platforms promise to resolve these shortcomings. With computers getting ubiquitous, mobile and even wearable, they are getting more and more involved into real-life environments. *Smartphones* accompany users in their daily routines and are able to sense their environment. Each device iteration is getting equipped with more accurate and more diverse sensors. Besides higher computational power and a different user interface that is centred around a larger touch display, the sensors are one of the key factors that distinguishes smartphones from so called *feature phones*. In western Europe, sales of smartphones already surpassed those of all other cell phones types [IDC11].

Together with the smartphone trend, mobile applications of services with a social networking context became more and more popular in the recent past. As of April 20, 2012, the currently largest network *Facebook* reached a total of over 900 million users, with over 500 million accessing the platform via their mobile devices at least once a month [Fac12]. Using a conventional social network service via a mobile user interface does not make a true MSN, however. To exploit their full potential, mobile phones can be used as *Social Sensors*. Based on the automatically gained information, new services could be built which enhance the functionality or make existing services more convenient for users.

### 1.1.3. Smartphones as Social Sensors

In the context of the research topic of *Human Behaviour Understanding*, Salah et al. [SL11] showed an overview of relevant data that can be gained from smartphones which enables them to be classified as *Social Sensors*. They can sense the following features:

- **Location.** Using various satellite navigation systems or wireless network triangulation, the phone can determine its position to a certain degree.

- **Other devices in physical proximity**. Via Bluetooth scanning or Near Field Communication (NFC), other devices in mid or near distance can be recognised.
- **Movement patterns.** Inertial sensors allow to draw conclusions about physical activities.
- **Communication.** The mobile phone knows *with whom* the user calls or texts, *when* this happens and *how long* such dialogues take. In theory, even contents could be analysed in addition to this.
- **Device status and interaction.** This includes for example network coverage, whether or not the device is charging or how the phone is used (games are played, user is surfing, etc.).
- **Peripheral devices.** The sensing capabilities of the device can be extended further by connecting to peripherals, e.g. watches with heart rate sensors or many others.

All those data can be used to examine the users behaviour and activities on comparatively short time and spacial scales. These observations can in turn be used to reason about social interactions of the person as well. Previous work in the domain of *Reality Mining* focused on collecting a multitude of raw data of those aforementioned classes and analysing them with *Data Mining* methods. While with this broad generic approach a lot of insights could be gained already, it does not make use of domain knowledge in the collection process.

### 1.1.4. Inference on Social Situations

Other preceding researches cover the location feature, which plays a major role in the multitude of possible behavioural cues. One of the key indicators for a social interaction taking place showed to be the knowledge about the positions of several persons at a point in time, which is also a result of reality mining and is further explored in [Leh09].

On a large scale it may be sufficient to know only about the location of the present actors. To examine smaller scales in both time and space, however, other features play a more important role. To estimate with whom a person interacts, a key factor is to know where both participants are looking at. This angle with which both are oriented to each other together with the knowledge about their distance, can be exploited to reason about the presence of an interaction and to some degree also about their type of relationship. These tuples of relative orientation and distance are explored in more detail in [GLR+10].

Another topic is the inference of types of activity from smartphone sensed data. The classification of activities enables to make more detailed assumptions about the likelihood of the social situations taking place. Activities are therefore another key to decide whether a social situation takes place and determine the precise parameters of the interaction. This is especially true in combination with the aforementioned method. Two interacting persons' relative orientation is different, if they are standing face-to-face to each other or jogging side-by-side, for example.

The aforementioned topics could also be grouped into two main aspects, which will be further detailed in the following, as they are especially related to the topic of this work:

- **Motion capturing.** The measurement of relative orientation of persons falls into the larger topic of motion capturing, as it involves reasoning about the geometry of the actors bodies. This can be as coarse-grained as only tracking this very orientation angle or as fine-grained as tracking all movable body parts or even facial expressions.
- **Activity classification.** This topic has gained a lot of contributions lately, as sensors are getting smaller and more affordable. It has not only applications in the realm of social signal processing, but in many other different contexts.

There is a multitude of related work on the topic of activity classification by inertial sensors. This includes applications in medicine [SCCD], sports [ABMp+] and life-logging [KL07, RDML]. Sensors are also used to determine modes of transport, such as walking, running, cycling or driving a car [NSY, BI04]. However, with these approaches, the activities are classified into predefined states, leaving no room for activities which do not fit in any of these. The problem that is inherent to classifying in general is also that there is no statement about those states which lie in between them or about transitions.

A continuous mapping can be, depending on the specific application, more expressive. Such observations are made by motion capturing, which will be regarded in the following.

## 1.2. Motion capturing

Motion capturing comprises a large field of different techniques and areas of application. An overview over current methods is given in [WF02]. For this work, it will be concentrated on tracking of human body geometry, which is only one of the branches in this area. There are very different approaches, which target on the reconstruction of continuous features of body geometry, the complete skeleton or a surface mesh of a given body. These methods are typically applied in real-time, i.e. not on static subjects but focusing on motion, as the name already implies.

A short overview over the existing approaches can be given with the following categorisation:

- **Optical**:
  - **Single camera** and computer vision, e.g. body mesh fitting
  - **Multiple cameras** for three-dimensional information, e.g. stereoscopic image recognition
  - **Marker-supported** camera systems
  - **Depth sensing** devices like time-of-flight cameras or structural pattern scanners
- **Inertial sensor** based, measuring forces or orientation of body parts they are attached to.

This list is by far not exhaustive and just illustrates a selection of very different techniques that exist for gathering body geometry data. It can be seen, however, that the systems can be grouped in two basic categories: ones that are measured from the outside, such as camera systems and some which are sensing from the inside, such as inertial sensors.

Many traditional motion capturing techniques are based on markers, which are attached to the persons to track. These markers are either actively sending out signals or light or are illuminated and reflecting the light again. The scene is then viewed from several angles simultaneously, allowing for calculating back the position by regarding their projections to all cameras, of which the position is known. This or similar techniques require a complex system, that has to be configured or calibrated and is therefore constrained to the static location of this system. The accuracy on the other hand is unbeaten by any of the other methods mentioned in this work. The main problem with these approaches is, however, that for tracking persons and observing social situations, this is not a very natural environment.

Approaches that are much less obtrusive, as they do not require any body-worn markers and only require a regular video camera are computer vision methods. Here, the body geometry is tried to be recognised from a given image, or sequence of images. This tries to emulate what the human brain does as well when looking at another person: it can comprehend in which state and pose his body is by just the eyes' image. Advanced techniques try, for example, to fit a body mesh onto a recognised person, which can additionally be supported by constraining the model with help of inverse kinematics, as presented in [PLGR12]. These class of methods typically work in any environment as long as enough light is available, but yield results of comparatively less accuracy.

Depth sensing devices provide a compromise between traditional and solely image based tracking methods. The reason for this is, that with 3-dimensional images containing depth information, the persons can be detected much easier and therefore the body be tracked. The tracking accuracy is generally expected to be higher than with 'flat' image recognition, though still only a single observing device is needed. Some depth sensing devices may have problem with outdoor environments, however. Sensors of this type will be discussed in detail later in this work.

An also noteworthy approach was taken by Schwarz et al., by combining the motion capturing principle with making use of inertial sensors. They presented a method to estimate poses in [SMN11], where six body-worn orientation sensors are used for tracking persons. While continuously checking for the most probable activity, the pose is estimated via a mapping of the selected activity-specific model. The activity can be one of several predefined discrete states in this method. In some parts this is even superior to complex camera and marker installations for motion capturing: occlusions are negligible as each sensor can measure independently from any line of sight. Also, the tracking works everywhere as it does not have to rely on any measurements from the outside. However, it is yet limited to certain trained pose classes and the user still has to wear several sensors all over his body.

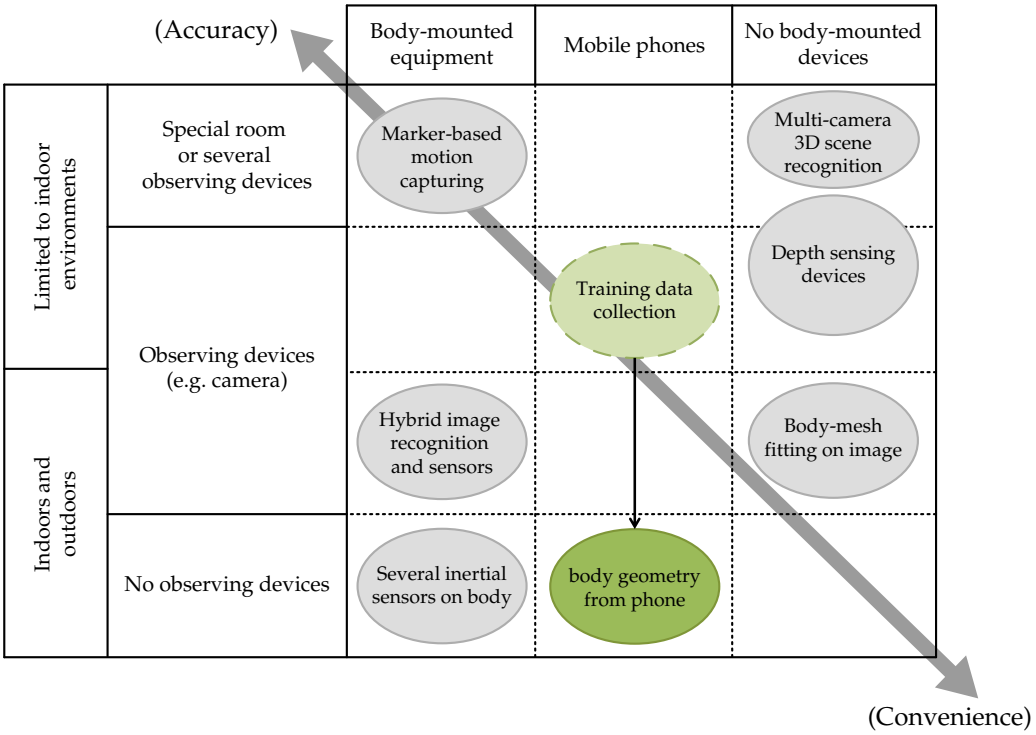



Figure 1.1.: Classification of different motion capturing techniques and the overall effect of either maximising accuracy *or* convenience. The approach of this work is depicted in the green circles.

## 1.3. Approach of this work

All of the previously mentioned related techniques for determining body geometry have the disadvantage of needing either to observe situations from the outside or requiring the user to have several devices mounted to his body. For a more unobtrusive approach, the idea of using the smartphone as a social sensor, as mentioned above, will be picked up again.

The method presented in this work is outlined as follows: inertial sensor measurements of the mobile device, which is carried by a person, are taken as a basis for determining certain features of the user's body. To enable this, a machine learning algorithm is trained to transform those given sensor measurements to the target body features. It is trained by using values from a full-skeleton motion capturing technique as examples. The skeleton tracking is done by a depth sensing device and an appropriate software component. Once trained, the mobile phone can ideally act like a motion capturing device itself, tracking several key geometry features of the user's body while being carried.

Figure 1.1 clarifies the goal once more by arranging several motion capturing methods in several dimensions. The different methods can be compared in the vertical axis in terms of complexity of their systems, with some needing many observing cameras, others requiring

none at all. Also, some are limited in their environment in which they can be applied.

The horizontal axis describes the devices which have to be worn on the body. A compromise can be found in the middle, where a smartphone has to be worn, which is not as obtrusive however, as it is carried anyhow by many persons. Previously discussed approaches are shown in this comparison, as well as the one presented in this thesis, which is splitted into the trained and the training phase. When being trained, both the mobile phone and the depth scanning device have to run their measurements synchronously to generate a training data set of sample pairs.

### 1.3.1. Goals

Figure 1.1 also shows a general rough trend of convenience for the user and accuracy of tracking results being rather contradictory. The body posture information yielded by the smartphone will not be able to compete with motion capturing techniques that involve cameras or even markers, as they have insights about the whole body. Nonetheless, it shall be answered in this thesis, which body features can be determined reliably with smartphone sensors alone and how accurately they can be tracked.

Another goal of this work is to examine if the outlined method is suitable to infer the relative orientation of two persons to each other, as already mentioned was a key component in [GLR+10] to conclude about social situations. This, in turn, can be inferred if two peoples headings in a common reference system are known. These headings can be described by two vectors. If both persons are situated on the same elevation, those vectors lie in the same plane, parallel to the ground. The system can then be simplified by reducing them both to angles relative to a virtually infinitely far reference point, e.g. the magnetic north pole. Such angles in a common local reference coordinate system will be examined in this work. The use of combining sensor data with real body geometry allows for a new point of view onto the data, as the persons heading can be very different from the devices heading.

The motivation for this work is also, that even though several possible error sources are introduced when transforming sensor data into body geometry, the output could be enriched with new information. This is potentially achievable, because even though no additional information is introduced directly, the transformation can be supported by implicit knowledge about human motion and limitations of the skeleton and its joints as well as typical patterns of movement of humans. Whether or not this domain knowledge can be used to receive more expressive data through this transformation shall be shown in this work as well.

If such a method proves to be reliable, it can be applied to many more areas as well rather than only for social situation explorations. In medical applications, for example, the method could be used for monitoring purposes. Related work in this area shows that there were already similar approaches taken, such as the monitoring of movement disorders by observing body positions with inertial sensors [KJvdK98] or by raising alarm if elder people have accidents [ZA07], which is also determined by worn sensors. It could also be applied in areas like life-logging or as an enrichment for reality mining.

### 1.3.2. Assumptions

A study of the mobile phone location in public spaces has shown that the majority of male phone owners carry their devices within their trouser pockets [ICG05]. The study also showed, that most women carry their phones in their handbags. Unfortunately, it is very hard to infer anything about the persons body posture from the devices sensors, if it is lying inside a bag as the attitude of the device is varying each time it gets taken out of and being put back into the handbag. Perhaps, the forces that can be measured from the phone could be used to distinguish between different modes of movement, such as walking, running or standing. However, a more fine-grained differentiation between body postures with a low level of energy is hard to make. Even more so, telling apart if the handbag is carried by a person standing still or not being carried at all could be difficult as well. These assumptions are not further verified in this thesis, as the trouser pocket was chosen to define a scope for this work.

As another indicator for this being a good choice, it was shown that for different positions of inertial sensors on the body, the trouser pocket proved to be very suitable for determining different activities, compared to other locations such as the wrist or on a necklace [MSSD06].

However, even with the restriction of only addressing mobile devices that are carried in the pocket, many variables are still to be determined by the algorithm, such as the exact pocket (left or right, back or front) and the orientation of the device inside it. This will be discussed in chapter 6.1 in detail.

### 1.3.3. Further considerations

There are two different modes, such a learning algorithm can be trained: individually for each person that is using the application (I) or with a broad range of training sets from different people that are aggregated and used together as the training data (II). While the second method is more convenient, individual training could yield more accurate results. Each person has different body metrics and slightly different patterns of their typical movement. These metrics and patterns could then be used to further restrict the learning algorithm. This in turn could allow for building a more exact model of valid body postures, that can be produced by the algorithm. Furthermore, user-specific facts can be included in such a model, like the preferred pocket, the mobile device is located.

This is especially useful, as a person's pocket location typically does not vary over a long period of time, as detailed later in this work and confirmed by earlier results in [ICG05]. However, even with all those advantages, the process of training the persons peculiarities to the algorithm can be inconvenient to the user himself. He has to have access to a depth scanning device, configure the device with a computer and the special software, while at the same time connect his mobile device to the same PC and software. Having set up the training environment, he ideally has to perform all typical motions that should be detected by the learning algorithm. This process is time consuming and error-prone, if not done in right way. In the worst case, an incorrect or insufficient training data set is produced, that

produces poorer results than having used a generic, user-independent data set in the first place.

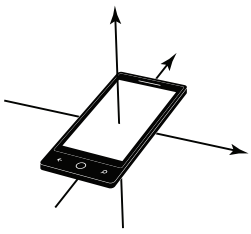
These consideration have led to taking the approach of building up a training set from sufficiently many different persons to form a generic data set. It should be mentioned, that even though this approach was taken, the basic concept presented in this work can be applied together with an individual learning phase just as well. The impacts of this decision will be further discussed on the basis of actual test results in the evaluation chapter at the end of this thesis.

# 2. Mobile Device Sensors

As the general concept has been outlined, first the sensors of the mobile device will be presented in detail, as these will be used in both the training phase as well as in a trained state to infer about the actual body features. To achieve a high success rate with the machine learning algorithm, it is fundamental to feed it with as much data from the sensors as required and at the same time trying to only provide information, that is relevant for extracting the target values out of it. If some data is omitted, it could be the case that the algorithm does not work with the accuracy that it could, because it could be the missing piece that is needed to improve the desired accuracy. The opposite case, i.e. too much data, could as well have negative causes: The computational power that is needed increases, without being able to gather additional information. It also adds to the risk of getting into a state where too much irrelevant data is used, thus resulting in an overfitting of the model, as explained later.

Therefore, it is essential to take a closer look at the properties of the sensors, available in the device. This enables to determine what exactly they can measure and, equally important, what they can not. Furthermore, each of the three presented sensor types can be implemented differently in hardware. This means there are different accuracies and operating ranges. Before discussing which exact parameters of the sensors are taken into account for this work and in which exact form, all sensors and their functionalities are described in the following. For each type of sensor, the most common implementation for mobile devices and their corresponding tolerances and operating ranges are given.

## 2.1. Accelerometer



An accelerometer is a sensor device, capable of measuring accelerations and therefore forces that affect the device. In today's smartphones, accelerometers are realised as *micro-electro-mechanical systems* (MEMS), measuring forces in all 3 axes. A 3D Vector can be deduced from those three single measurement values, that represents the direction and strength of the force that is applied to the device. With the device lying still, the vector describes the gravitational force, as it is the only acceleration that is present. It has a magnitude of about 1g and is pointing to the centre of the earth, which is the downwards direction in the local *north-east-down* (NED) coordinate system. As this way the down direction is known, it directly allows to infer the 2 degrees of freedom (DoF) orientation of the device relative to the ground plane.

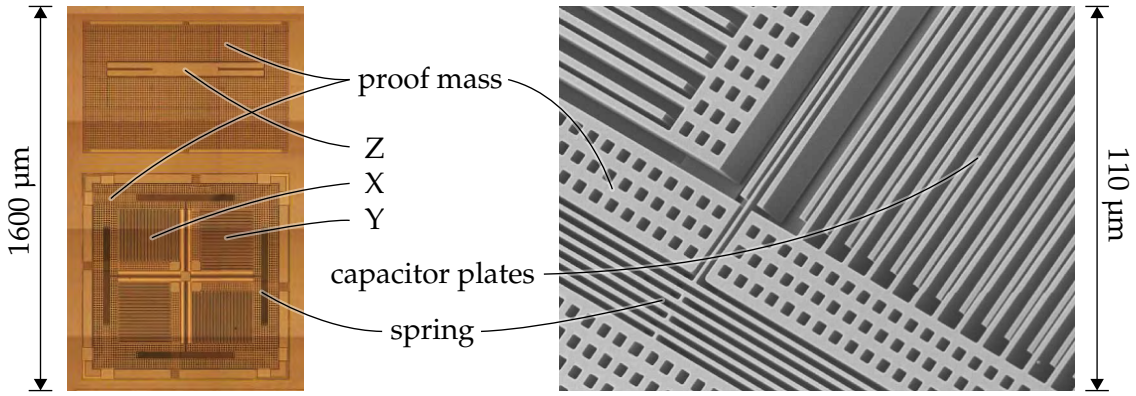



Figure 2.1.: Scanning electron microscope (SEM) pictures of the LIS331DLH accelerometer of STMicroelectronics, with 3 degrees of freedom (Original images from [DW10])

In case the phone is being moved or shook, the device is subject to not only gravity, but also forces that are the cause of these rotational and linear motions. The acceleration, measured by this sensor is therefore a composition of different types of forces. In such a case, the gravitational component and therefore the orientation of the phone can only be estimated, based on previous measurements. This is typically done by applying a low-pass filter on the sensor output signal, which drops all high-frequency influences, like short movements and shaking of the device. This holds a critical disadvantage, however, as the responsiveness of the measured orientation is getting worse. The signal lags behind quick rotation motions of the device, as the filter cancels them out like they were disruptions. The signal will asymptotically approximate the right value eventually, but it takes longer than it would have without the filter. The combination with a gyroscope will solve this issue, as detailed at the end of this chapter.

The structure of a MEMS accelerometer is typically planar, with loose structures that are held with springs and equipped with proof masses, all etched out of silicon. Attached thereon, a multitude of capacitor plates are placed in a comb-like structure to actually measure motion of the spring-mounted mass. Figure 2.1 shows these structures. The part for measuring the $z$-axis is laid out differently to account for reacting to forces perpendicular to the chip.

Typical operating ranges of MEMS accelerometers are about $\pm$ 2.5 g, with a zero-level offset of up to 40 mg and a sensitivity of under 20 mg [STM08], whereas newer prototypes even allow for much higher accuracy [AAA07].

## 2.2. Magnetometer


A magnetometer is integrated into more and more smartphones today [Col11]. Such sensors are typically able to measure magnetic flux density in all three orthogonal spacial directions. The main purpose of the sensor as a whole is to detect the direction of the earth's magnetic field lines, and therefore act like a digital compass. The direction in form of a 3D vector can be deduced by combining the three single flux density measurements. As the field strength is adequately equal on most locations on earth, it can also be reasoned about the presence of electro-magnetic culprits or the need for recalibration of the sensor due to bias.

As of now, there are mainly Hall-effect based magnetometers in current mobile devices [TLH11]. The effect that is exploited by this type of sensors is named after its discoverer Edwin Hall. He described a voltage shift across an electrical conductor that is orthogonal to both the electric current and the magnetic field that is applied [Hal79]. On a flat square conductor, for instance, this effect can be used to detect a magnetic field that is perpendicular to the plane by applying a current between two opposing edges and measuring the voltage difference between the two other edges. The sign and amplitude of this measured shift is then directly related to the direction and strength of the magnetic field.

The advantage of such Hall-effect based magnetometer sensors is that they can be produced in the form of a CMOS chip, which can be fabricated inexpensively in existing production facilities. In principle, the sensors can only detect fields that are perpendicular to the chip's surface, like already explained. To enable the detection in all three directions, an additional magnetic concentrator is used. This coat, made out of special ferromagnetic material, bends the field lines to allow the detection of magnetic fields which have originally been parallel to the sensor's planar surface. Figure 2.2 shows the layout of such a sensor: The 8 square structures are the electrical conductors which are used for measurement, some of which are covered by magnetic concentrators in a layer above (not seen in this image).

Newer development has also lead to Lorentz force based magnetometers, which will be integrated in smartphones in the near future [GR11]. These sensors are MEMS-based and have higher sensitivity and less sensitive to temperature shifts [RWY+09]. The Lorentz force, which is also responsible for the Hall-effect in electrical conductors, is exploited to measure the magnetic field strength in these sensors.

Independent from the type of magnetometer, these sensor structures are combined into a package, containing feedback control systems and a complete interface logic for digital connections to the outside of the sensor device. Current sensors have a measurement range of $\pm$ 1200 µT, or depending on the sensor even up to $\pm$ 8000 µT, with a resolution of 0.3-0.8 µT [Asa10, STM11].

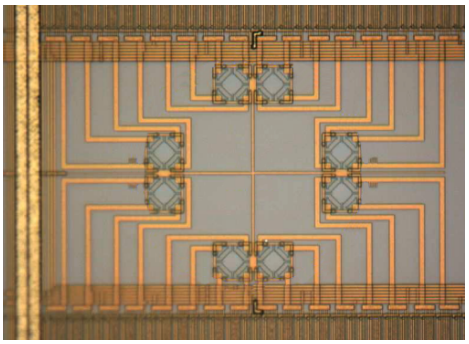
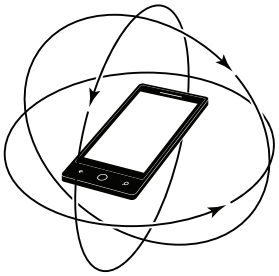


Figure 2.2.: Microscopic image of the AK8973 Hall-effect based magnetometer of Asahi Kasei Microsystems, which is built into a 2010 released smartphone. The width of the whole image section is about 500 µm in the real chip. (Image taken from [DW11a])

## 2.3. Gyroscope



A micro-electro-mechanical gyroscope, which is built-in to more and more mobile devices [Col11], can measure angular rates in 3 dimensions, i.e. the rotation speed around each axis. Since it can only sense angular velocity, it only tracks relative changes of the device's attitude, not the absolute orientation related to a reference frame. Those values alone are not particularly valuable in most situations. However, in combination with the aforementioned sensors, the gyroscope becomes useful to stabilize measurements and get more accurate results. This will be elaborated in detail later.

The fabrication processes and structures of a MEMS gyroscope are very similar to those of its accelerometer counterpart. However, the working principle is different and so is its layout. The fundamental principle that is used for this type of sensor is that it acts like a tuning fork gyroscope [Nas05]. It consists of pairs of proof masses that are loosely attached to the surrounding silicon with springs. The capacitor plates are this time not only serving the purpose of measuring but also for actively driving the mass to a vibration at a certain frequency. The physical effect that is exploited, like in many types of gyroscopes, is the Coriolis effect. When the device is rotated, the Coriolis force generates a vibration that is orthogonal to the one that was produced actively by the sensor. The amplitude of this orthogonal vibration can be measured by capacitive electrodes and enables to conclude the angular rate which was applied to the device.

Figure 2.3 shows the structures of two different such gyroscopes. In the upper part of the illustration, the sensor consists of four 'wings' inside the circular structure in the middle of the silicon. The whole structure is oscillated in a rotational manner around the centre by an array of capacitive driving plates that are located to both the left and right sides. Each two opposing wings form a pair of masses that are bent by either a perpendicular acceleration or by the Coriolis forces. In case of a linear acceleration, both wings are bent

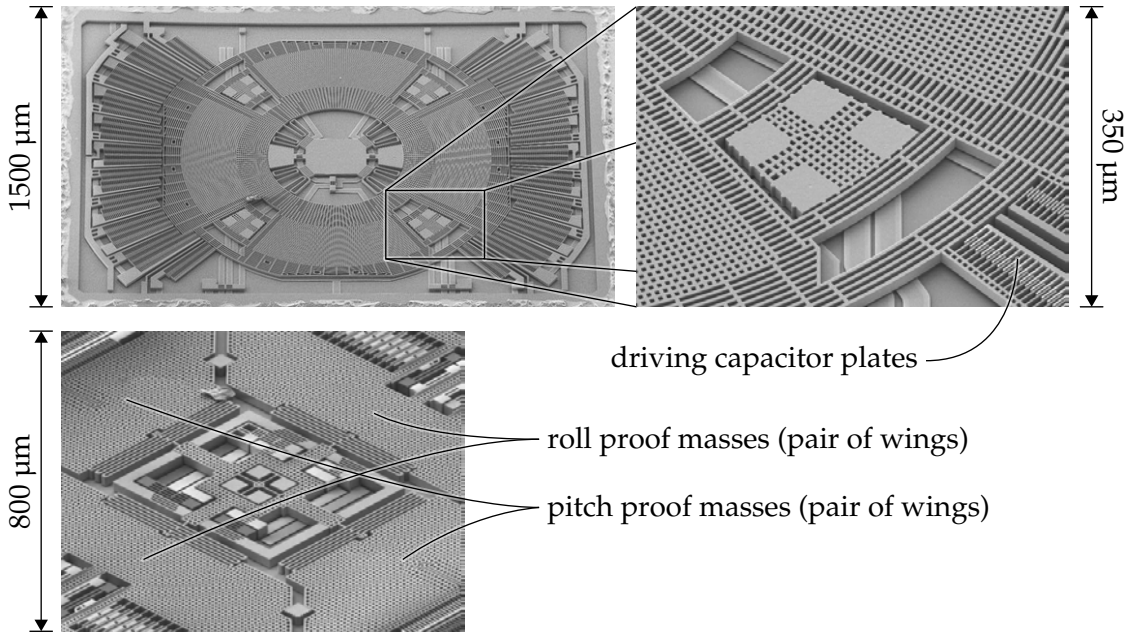



Figure 2.3.: SEM images of the MEMS gyroscopes LYPR540AH (above) and L3G4200D (below) of STMicroelectronics (Images taken from [BD10] and [DW11b])

in the same direction. If an angular rate is applied, however, the wings are moved to opposing sides. Therefore, the difference of the two wing's measurements is taken, which then only contains information about the angular rate and is completely insensitive for any linear forces.

As already mentioned, MEMS gyroscopes can not measure absolute orientation, unlike classical mechanical gyroscopes which exploit the inertia of a spinning mass. When integrating the angular rate over time, the relative change in orientation to a certain previous point in time can be calculated. This way, however, bias errors of single measurements are summed up to a large drift, getting worse the longer the measurement sequence takes. To account for this drift, it has to be corrected with absolute values regularly, like explained in the sensor fusion section shortly.

Typical measurement ranges of current MEMS gyroscopes are at angular rates of about $\pm$ 270°/s with a sensitivity of 0.005°/s to 0.01°/s [STM12].

## 2.4. Sensor fusion

In this context, the process of *sensor fusion* connotes the logical combination of all three presented sensor types into one virtual sensor. At first, the accelerometer lets infer the current attitude of the device. This, however, still leaves one rotational degree of freedom around the up-down-axis. This final DoF can be constrained with data gathered from the magnetometer. The final result is the device's complete attitude, relative to the local NED reference system.

However, the fusion of these two sensors alone yields to a very noisy signal that also has a poor dynamic response with quick movements resulting in overshoot of the signal. Also, as mentioned earlier, with a lot of forces applied by the user, the gravity vector can no longer be determined exactly but rather only be estimated. Furthermore, magnetometers typically have a poor response time to quick movements and noticeably lag behind the correct magnetic heading.

The gyroscope, however, does not have all those weaknesses. In fact, it has a very short response time and is much more accurate in perceiving motions, even quick ones, than the magnetometer. As already mentioned, with integrating the angular rate values of the gyroscope, the attitude of the device can be obtained. This attitude is only relative to the orientation at the start of the sequence, but it is much less noisy than the one that resulted from the accelerometer values and has an accurate dynamic response to quick movements as well. It should be noted, that the raw gyroscope signal is noisy and biased as well. The difference comes from the integration, where these are summed up to a drift over time while the noise is damped significantly at the same time.

With having noisy but absolute values on the one hand and accurate but relative and drift-afflicted values on the other hand, it seems reasonable to combine these two signals and take the best of both worlds. There are, amongst others, basically two ways to achieve this:

1. The first and simpler approach is to take the low-pass filtered attitude signal of the accelerometer and compass and add it to a high-pass filtered integrated signal of the gyroscope, like described in [Col07]. The low-pass ensures that noise is removed from the signal and the longer-term absolute values are preserved, whereas the high-pass removes the proportion that is subject to drift and leaves the quick and accurate short-time responses of the gyroscope.
2. A second way is to use a *Kalman filter* with a suitable model of the sensor inputs and their noises. The application of such a filter to combine inertial sensors is described in [Leh09].

To avoid the *gimbal lock* problem of Euler angles, which are directly deduced from these triaxial sensors, rotation quaternions are used as an output of sensor fusion algorithms to express the device's attitude.

One more information can be extracted from the combination of the inertial sensors: after having a good approximation of the attitude, it allows for the back calculation of the gravity vector. Subtracting this from the raw accelerometer values, the linear acceleration

information is left over. Effectively, the combination of just the accelerometer and the gyroscope via sensor fusion allows for separation of gravity from all other accelerations and therefore solves the basic problem that was present with only having the accelerometer by itself.

There are already complete *Inertial Measurement Units* (IMU) in the size of a single one of the previously mentioned sensor chips. They have all three inertial MEMS sensors and their integrated circuitry in a single package. Even a processing unit for computing the sensor fusion calculations is built into the chip package [Inv12]. These are, to the best knowledge of the author, not yet integrated in devices on the market today, but will certainly be in the near future. Aside from saving space in mobile devices, which are getting smaller and smaller, they also help saving power due to the further miniaturisation.

Some of today's smartphone operating systems provide an implementation of such a sensor fusion mechanism in the OS itself. This additional hardware abstraction layer above the underlying sensors supersedes the need for knowing which sensors are actually available. If only an accelerometer and magnetometer is available, the orientation output will be calculated only by filtering the acceleration values and combining them with the compass readings. In case an additional gyroscope is recognised in the device, the advanced sensor fusion method, described above, can be used. Even if a hardware integrated IMU is present, the interface does not change, but instead the values are just passed through to the consuming application. Moreover, developers do not have to start from scratch, handling raw sensor values, but can instead focus on processing more meaningful values like the device orientation in a common reference frame or the linear acceleration without the gravity part.

Such a complete interface in the operating system to sensor fusion was introduced in Microsoft Windows Phone 7.1 with the *Combined Motion API* in April 2011 [Mic11c]. Google's Android features an interface for the so called *Rotation Vector Sensor* that used to combine only accelerometer and magnetometer data since OS version 2.3, but was updated to incorporate gyroscope values with Android 4.0 as well [Goo11], which was released in October 2011. Apple iOS now also supports such functionality since the iOS 5.0 version that was released in March 2012 [App12].

# 3. Supervision Sensor

The sensors of the mobile device provide the data, which will be used to reason about the user's body geometry, as outlined in the concept before. To enable this, a reference of true target values is needed to train against. For this purpose, a motion capturing sensor system will be used to collect the corresponding body geometry in real-time together with the mobile sensors.

In this work, a *Kinect* depth sensor will be utilised for this, together with an adequate motion capturing software component. Released by Microsoft in 2010, this sensor device was developed with the use of 3D-sensing technology of PrimeSense, a Tel Aviv based company [Mic10]. Its original purpose was to be a gaming sensor to enable controller-less casual games, but was recently also officially released as a *natural user interface* (NUI) input device for the PC. Together with this, a slightly modified version of the Kinect was released specially for the PC, that has a broader field of sight and a nearer depth range covering a smaller area. For this work however, where a more distant and therefore also larger range is more suitable, the original version was used. A picture of the device can be seen in figure 3.1.

The Kinect sensor provides with the following integral sensors and actuators, some of which will be described in more detail later:

- **Camera:** A CMOS image sensor with a Bayer-Pattern colour filter that outputs a stream of 30 RGB images per second, i.e. the same technology like built into usual webcams.
- **Depth sensor:** A structured-light 3D scanner system, consisting of an infra-red (IR) projector, an IR image sensor and a processor. It outputs a stream of about 30 depth frames per second.
- **Audio processing:** A microphone array consisting of 4 microphones and a sound processor for noise suppression and echo cancellation. Beyond this, the array enables the computation of virtual directed microphones via beamforming to be able to only 'listen' into a certain direction. This can also be used backwards to locate from which direction a sound source came.
- **Orientation sensor:** An accelerometer for measuring the devices current orientation by looking at the direction of earth's gravity vector, similar to the one described for the mobile device in the previous chapter.
- **Tilt adjustment:** A stepper motor to tilt the device up and down to a certain degree for adjusting the vertical range of sight and help identify the ground plane.


Figure 3.1.: The original version of the Kinect sensor device. Both the RGB and IR cameras are located in the middle, the IR projector can be seen to the left. (Image taken from Mircosoft press release)

## 3.1. Depth imaging

The important part of the sensor for this work is the depth sensing ability. For this purpose, the IR projector sends out a specially design dot pattern of light points, which are then reflected by the objects and persons in front of the device. Depending on how far or near the subjects are located from the sensor, the pattern in this region is going to be more densely or sparsely spread from the device's point of view. The dot pattern appears to be random, but indeed has a logical structure which allows to reconstruct the location where it belongs to. Furthermore the pattern can be broke down into two patterns which are self-similar, one being more sparse with brighter points and a more dense one consisting of dots of less luminance. This allows for achieving a better depth resolution in nearer areas of the image, whereas the bright and sparse one extends the measurable range for distant parts.

To detect this projected and reflected pattern, an IR camera is built into the device which is formed from a common CMOS image sensor that sits behind a filter to only let light from the infra-red spectrum pass. A processor, integrated in the device, computes an estimated distance for each pixel of the frame. It does so by analysing the reflected pattern. In this process, the fact is exploited that the device knows how the pattern is structured, which is fixed and calibrated for each device. Therefore, this computation follows the principles of a structured-light 3D scanner.

In contrast to other techniques, such as time-of-flight (ToF) cameras, this allows to utilise low-cost hardware components and yet get a real-time stream of depth images. ToF cameras measure the distance based on the time the light needed to travel from the projector to the object, be reflected and travel back to the sensor. The distance travelled is then simply this measured time multiplied with the speed of light in the medium of air. To distinguish when the light was emitted, it is sent out in periodic pulses with a sufficiently large period. This principle enables the measuring of the distance of each pixel directly. In reality, however, sophisticated systems with high quality components have to be built to control the projector pulses and to read out the sensor to achieve sufficient accuracy. This causes the system to cost multitudes more than comparable structured-light scanners.

The main disadvantage of structured-light 3D scanners, the required computation of the depth map is compensated by the Kinect by having specialized hardware built into the

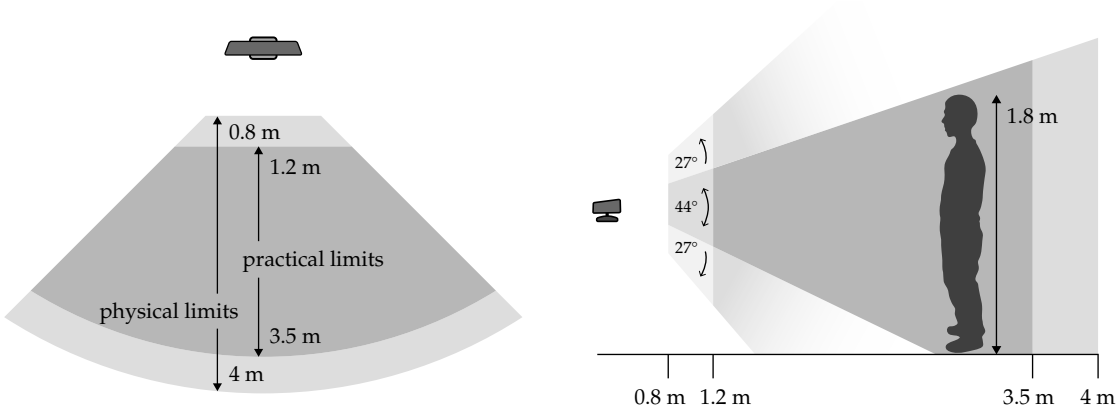



Figure 3.2.: The operating ranges for the depth sensor of the Kinect are shown in a top and side view. The angles to which the sensor can tilt up and down with its motor are depicted in the side view as well.

device that handles the processing in real-time. This relieves the work from the host computer as it gets the already processed depth stream as a sequence of 2D grey-scale images. Each pixel of such an image contains a value proportional to the estimated depth and lies in the range of 11 bit.

## 3.2. Depth sensing constraints

For later measurements to collect training data for this work's method, it is important to know what the Kinect sensor can measure and where its limits are.

First of all, due to the nature of the structured light scanner, its depth sensing ability is restricted to certain distances. If an observed object is to near, the distance can no longer be inferred. This is due to technical limitations such as the focusing of the camera and the fact that the IR projector and its camera counterpart are located apart from each other. If the distance gets to large, the dot pattern will be shrinking to a small cluster in which single points can no longer be distinguished. Physical and practical operating ranges of the first version of the sensor are shown in figure 3.2. The values for this illustration were taken from the Kinect SDK documentation [Mic11b]. Note, that the PC version of the sensor has different operating ranges, which are not of special interest for this work, however.

The digital resolution of a single pixel is linearly proportional to the distance. However, the precision depends strongly on the distance, which has a normally distributed error with standard deviations of $0.5$cm for a $1$m distance up to $2$cm for an object as far as $3.5$m away [Kho11].

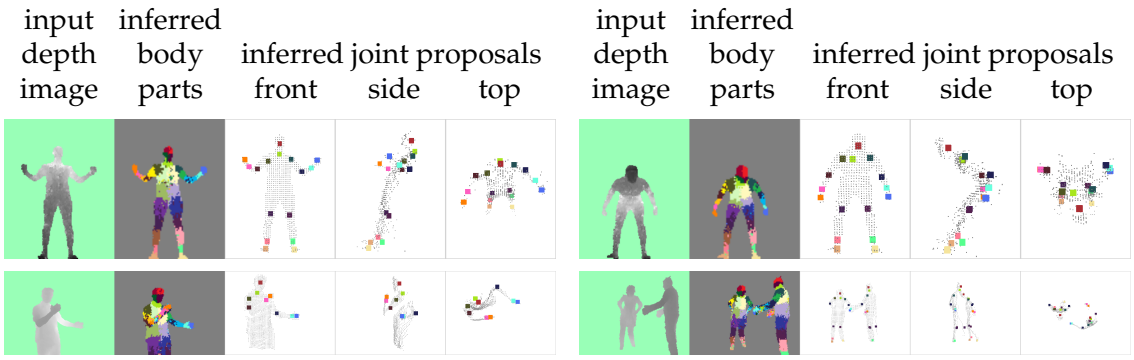



Figure 3.3.: Intermediate states of skeleton joint inference. (Images taken from [SFC+])

## 3.3. Frameworks

From these depth images, it is possible to recognise persons and even provide motion capturing functionality by fitting a skeleton into the sensed depth map of a subject. There are several software frameworks and drivers for the Kinect sensor. Some of them are open source projects, such as *libfreenect*, which were derived from reverse engineering the USB communication. They are only providing access to the sensor's depth stream, however.

The *OpenNI* framework, in contrast, can be used together with the so called *NiTE* middleware to compute such skeleton data. This project was mainly developed by PrimeSense for their reference depth sensing system before the Kinect was even available. In this framework, the skeleton data is estimated based on the difference to previous frames. Before recognising a person's skeleton, it has to be initialised with a special pose each time the user enters the captured space.

As Microsoft licensed PrimeSense's hardware technology for the use of the Kinect sensor, an own motion capturing algorithm was developed. It does not depend on previous states but regards each depth frame independently, which makes the recognition more robust. The algorithm, which was first only used for the Xbox gaming console was recently also released as part of an official framework for PCs, the so called *Kinect SDK* [Mic11a]. It includes the drivers for the device, a middleware component for skeleton estimation and interfaces for third-party software to access all these data. This framework was used in this work, as it provides a reliable skeleton estimation method.

The algorithm used in the Kinect SDK is described in [SFC+]: The object recognition is done by a per-pixel classification into body parts. For this classification, decision trees were built from a large dataset. This dataset in turn was gained from skeleton data of a traditional motion capturing system and depth pictures which were synthesised from these skeletons. As the depth maps were generated, their respective true skeleton was exactly known. In the synthesis of depth images, different body sizes, shapes and clothing were created to make the final classification more robust. If this trained decision tree is applied to each pixel of a Kinect depth frame, each one is classified to belong to a respective

body part with a certain confidence. The joints are then inferred from these body parts. The procedure is illustrated with examples in figure 3.3, where the raw depth image, the classified body parts and finally the inferred joints can be seen.

As with this method, the type of body part and therefore the type of joint is inherently classified at the same time, the skeleton bones can be simply connected between their corresponding joints. Even though the algorithm is even able to reconstruct parts of the skeleton if the person is not completely pictured inside the depth image, there are several restrictions with this method. Only two persons can be skeleton tracked at the same time. Furthermore, a person that is facing away from the Kinect can not be distinguished from one facing towards it. Also, occlusions that happen when a person stands sideways to the sensor makes the skeleton estimation difficult. Therefore, the best results are achieved if the person faces very roughly into the Kinect's direction. These restrictions were accounted for in collecting the test data, as can be seen later.

# 4. Machine Learning

As seen before, there are given measurement values from the sensors of the mobile device and the corresponding target values, also measured by a sensor in the learning process. These target values are taken as the ground truth to train a suitable machine learning algorithm. This kind of training is called *supervised learning*, where the algorithm is provided with pairs of input and output parameters which have to be brought into relation with each other. In contrast to *unsupervised learning*, where clusters, patterns, etc. are searched within the given data, there is a 'teacher' that provides the correct function value to a given input. The task of the learning algorithm is now to find the unknown function that produces these output values. It can directly verify the quality of its attempts by comparing its function values with the actual values from the teacher.

Several considerations have to be taken beforehand, however, that apply to any of the classes of supervised learning approaches. There is a trade-off between the computational cost, i.e. either the asymptotic or the actual runtime behaviour of the algorithm, and the accuracy that can be achieved. The processing time needed to train mainly depends on the type of algorithm that was selected for the task, if the number of dimensions is held constant. However, having decided for a specific approach, the number of dimensions not only has influences on the computation time, but can also have other impacts. Working with high-dimensional spaces, it is possible that some parameters that are in fact irrelevant, appear by change to be useful to the learner. Such a so called *overfitted* model is not only computationally more expensive, but also introduces a new error source: new data, that is not exactly matched by any previously learned data set, can lead to results that are much farther away from the expected value than with a lower-dimensional better fitted model, just because the irrelevant dimension interferes with the process. Following the principle of Ockham's razor, if multiple consistent models can be constructed from the input data, the simplest should be preferred. In this case simplicity can be defined as the number of dimensions, where less are to be preferred.

A similar argument applies to the selection of the machine learning algorithm. During the researches of this work, it became apparent that the selection of various different learning algorithms does not affect the accuracy of the results nearly as much as the addition of assisting domain knowledge. Such knowledge can be brought into the system by defining constraints, transforming the input data into more suitable representations and tuning the various parameters of the respective algorithm. These methods and their impacts will be presented and discussed throughout the next chapters of this work. It turned out, that algorithms with a concise and clear functional principle shall be preferred, if more complex methods are not significantly improving the overall accuracy. The results of a more complex algorithm are harder to keep track of and the chance of introducing new sources of error increases.

## 4.1. Algorithm selection

As the goal of this work is to reconstruct body geometry from inertial sensor data, it is essential to clarify in which form the input is available, what the desired output should look like and which classes of supervised learning algorithms are consequently suitable for this task.

The presented inertial sensors as well as the virtual fused sensor deliver a constant stream of floating-point numbers. Although limited by the finite and discrete floating-point representation in memory and ultimately even more by the accuracy of the sensors themselves, those numbers can be assumed continuous values for the purpose of this discussion.

At first glance, the raw measurement values of the triaxial sensors do not seem to have an obvious relation to the device's orientation and therefore seemingly do not allow to draw conclusions about body geometry. However, as already described by the sensor fusion process, this information is contained in the raw streams and can be extracted by it. Other features can be extracted as well by using implicit knowledge about the situation of the mobile device and correlations between various sensors. These transformations will be described in the next chapter.

As will be seen, these will mainly directly linear relationships between those features and the desired body geometry features. For these types of problems, where continuous mappings of linear or non-linear relationships are underlying, the class of *regression* algorithms is suitable. A special case of these is *linear regression*, where an optimal estimation for an underdetermined linear system of equations is searched, as an exact solution can not be found. In contrast to what the name might imply, the data itself that is used for training and later on for prediction does not have to have a linear relationship. Instead, the coefficients which are trained go into the model in a linear fashion. First off, the principle behind linear regression will be detailed in the following. It is also worth noting that an excellent introduction to this topic is given in [RN10, chapter 18].

## 4.2. Linear regression

Linear regression is a way to analyse a relationship between a dependent scalar variable, also called *response*, and a set of *regressor* values. In case such a relationship exists and has the required properties, the response values are sufficiently explained by a subset of the regressors or all of them. This means, that an function $\mathbb{R}^n \rightarrow \mathbb{R}$ exists that describes this relationship, which is not known however. Given only a limited set of examples for this relation, i.e. training data which is also subject to measurement errors, the tasks for linear regression are:

- Fitting a model, such that this relationship is fully described.
- Qualifying the importance of each of the regressor variables for the model or determining regressors which have no relationship with the response at all.

### 4.2.1. Hypothesis space

To find the best fit in the training data set, a function $h(x)$ is searched. This function should be used later on to predict response values from given input values by simply evaluating it. Those input values, or regressors, shall be defined as $x_{j,i}$, where $i = 1 \dots n$ enumerates the number of dimensions, which is the number of features. Furthermore, a dummy element $x_{j,0} = 1 \ \forall j \in \mathbb{N}$ is introduced for convenience, as can be seen later. The index $j$ determines the location of the $n$-dimensional vector $\mathbf{x}_j$ in the dataset, which in the case of this work is also the index of the time frame. Each time frame has a response value that was also measured and represents the target value, the trained algorithm shall yield given the regressor values. The training data is therefore a set of examples $E$ and will be defined as:

$$E = \{(y_0, \mathbf{x}_0), (y_1, \mathbf{x}_1), \cdots, (y_N, \mathbf{x}_N)\}$$

$$\text{with} \quad \mathbf{x}_j := \begin{pmatrix} x_{j,0} \\ x_{j,1} \\ \vdots \\ x_{j,n} \end{pmatrix}$$

The training data can be visualized by a point cloud in a $n+1$-dimensional space. Those $n+1$ dimensions are due to the $i = 1 \dots n$ input features that map to a single target dimension, i.e. the $y_j$ values. For example, a training set with $n = 2$ input dimensions that map to a response can be depicted in a 3D graph, with each element of the set being described by a point. The task is to find a function $h_{\mathbf{w}}(\mathbf{x}_j)$ that is fitted to lie right inside this point cloud and therefore having the least possible distance to each of the points. This vague intuition can be formalized by restricting how such a function may look like and by defining what is considered a good fit.

In the case of linear regression problems, all input parameters have a linear proportionality to the response values. Then, the hypothesis space is defined as the set of function of the form:

$$h_{\mathbf{w}}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = w_0 + \sum_{i=1}^{n} w_i x_{j,i}$$

However, other than the name may suggest, linear regression can also be used for non-linear relations in the data. A linear model only has to be linear in the parameters that need to be estimated. A model of the form $w_0 + w_1 x_j + w_2 x_j^2$ is quadratic in $x$, but is a linear model in the parameters $w_0$, $w_1$ and $w_2$. In this case all $x_j$ can be substituted by $x_{j,1}$ and $x_j^2$ by $x_{j,2}$ in both the learning and predicting phase.

The coefficients $w_i$ are real numbers and represent the weights that should actually be learned. A special case is $w_0$, the intercept or constant term. To get a more consistent

syntax, the dummy element $x_{j,0}$ that was introduced before, was defined to always be 1. This way, all weight coefficients can be grouped into a vector $\mathbf{w}$ and the function can be written as the following sum, or even simpler in the form of a matrix product:

$$h_{\mathbf{w}}(\mathbf{x}_j) = \sum_{i=0}^{n} w_i x_{j,i} = \mathbf{w}^\top \mathbf{x}_j$$

Later in this work, the following abbreviated notation will also be used for defining the hypothesis space with their concrete response and regressor variables: e.g. $y \sim x + x^2$ which denotes a quadratic relation between the single regressor variable $x$ and the response $y$.

### 4.2.2. Errors and Loss

The ultimate goal is to minimize the error between a predicted value using the model and the actual measured value. There are infinitely many possible metrics to quality such an error, however. Therefore, a suitable function $L(y_j, \hat{y}_j)$ is required that describes the *loss* between the measured value $y_j$ and the predicted $\hat{y}_j = h_{\mathbf{w}}(\mathbf{x}_j)$. Such a function encapsulates the metric for the error and should meet the requirements of linear regression. $L(y, y)$ is defined to always be zero, as there is no loss if the response is guessed exactly right.

The simplest example of a concrete loss function is the 0/1 loss, where $L_{0/1}(y_j, \hat{y}_j) = 0$ for $y_j = \hat{y}_j$ and 1 otherwise. It can be utilised in classifications, where $y$ holds one of several discrete states. In this context however, this is clearly unsuitable, as both the input and output dimensions are error-prone continuous measurements, which makes $L_{0/1}$ yield 1 for almost all pairs of values.

For real-valued data, other metrics have to be found therefore. As the general intuition is that small errors are better than large ones, the absolute difference can be taken for this reason, i.e. the absolute value loss $L_1(y_j, \hat{y}_j) = |y_j - \hat{y}_j|$. Another very important loss function is the squared error loss $L_2(y_j, \hat{y}_j) = (y_j - \hat{y}_j)^2$, which gives large differences between the two values much more weight than smaller deviances.

The *empirical loss* with respect to a loss function $L$ of a hypothesis $h$ is then defined as the total loss over all $N$ examples $\in E$:

$$EmpLoss_{L,E}(h) = \frac{1}{N} \sum_{j=0}^{N} L(y_j, h(\mathbf{x}_j))$$

### 4.2.3. Model fitting

To obtain the best vector of weights $\mathbf{w}^*$, which in turn corresponds to the best fitting, the empirical loss has to be minimized (the constant factor $\frac{1}{N}$ can be dropped for this purpose):

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{j=0}^{N} L(y_j, h_{\mathbf{w}}(x_j))$$

The temporal inaccuracies or noise of the sensor input values are expected to be normally distributed. As such, they behave like uncorrelated random variables with an expected value of zero and equal variances. The best linear unbiased estimator of the coefficients is then given by the least squares estimator, as the Gauss-Markov theorem states (elucidated in [Pla50]). The optimal vector of weights $\mathbf{w}^*$ is therefore found by letting the loss function be the squared error loss $L_2$:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{j=0}^{N} (y_j - \mathbf{w}^\top \mathbf{x}_j)^2$$

The minimisation is typically done with a *gradient descent* algorithm, as there is not necessarily a way to do this in a closed form in higher-dimensional spaces. The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is a vector denoted by $\nabla f$ and gives the magnitude and direction of the steepest slope:

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n} \right)$$

Starting at an arbitrary point $\mathbf{w}^{(0)}$ in the weight space, a weight candidate $\mathbf{w}^{(\tau)}$ it is iteratively updated by going in the direction of the steepest descent. $\tau$ denotes the iteration number. At some point, a local minimum will be reached eventually, where the algorithm converges and the final weight settles. The step size is usually adjusted by multiplying the gradient with a factor $\eta$, which is in this context also called *learning rate*. This has to be carefully chosen, as too slow steps lead to a very slow convergence, whereas with too large steps it may step over a relevant minimum completely. An update to a new step can therefore be described by:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \, \nabla EmpLoss_{L_2,E} \left( h_{\mathbf{w}^{(\tau)}} \right)$$

There are many other variations of the gradient descent algorithm [Gar84] that improve this basic method by more likely finding a *global* minimum than only a local one or which are converging faster. Furthermore, other minimising algorithms, like for instance *simulated annealing* [KGV83] amongst others, can be used to solve this optimisation problem as well. In the end, there is always a trade-off between the quality of the results and the runtime of the algorithm. More detailed discussions of gradient descent can be found in [Sny05] and especially with emphasis on machine learning in [RN10] and [Bis06].

As the model is fitted by now, i.e. the supervised learner is trained, the estimated response $\hat{y}$ can be predicted from a set of new regressor values $\mathbf{x}_{\text{new}}$ by simply using these trained coefficients $\mathbf{w}^*$ and evaluating:

$$\hat{y} = \mathbf{w}^{*\top} \mathbf{x}_{\text{new}}$$

# 5. Preprocessing and Reconstruction

Before running the machine learning algorithm itself, the input data has to be transformed and processed. The reasons for this are:

- A lot of the raw values are not applicable to linear regression, as well as to other machine learning approaches, without a preceding transformation. Such a conversion is not necessarily lossy. In contrary, many of them are bijective mappings and are reversible in the scope of the machines numerical accuracy, as can be seen in the next sections.
- Domain knowledge can be used to transform input values in a way, that irrelevant information is excluded thus only data of interest remain. The implicit knowledge lies in the selection and design of the transformation function. This can also be useful for reducing dimensionality in advance, as data of interest is sometimes contained within a combination of several raw parameters. If done carefully, this can significantly enhance the quality of the learners response, as there is a reduced chance of including information into the learning process that is falsely classified as relevant, i.e. overfitting the model. The risk with such an approach is missing relevant data and therefore a decrease in overall result quality, as will be discussed in more detail later on.
- The dependency of the response variable on chronologically preceding regressor values shall be examined in this work. As the time window needs to be sufficiently large to inspect the influences on the results, the dimensionality would explode. Therefore, the only feasible possibility is to reduce all values of the window to a set of characteristic features. Those transformations will be presented later in this section.

## 5.1. Multiplexing

The first step of preparing the raw sensor streams for the learning process is multiplexing. The word multiplexing is commonly used in the context of electronic switching or telecommunication, where several signals are combined into a single sequence that is transmitted over a shared medium before being decomposed again into the original signals by a demultiplexer. In this work, the name was chosen for a method that is related to this, but has a slightly different meaning. The goal here is not to form a stream that can be demultiplexed again, but solely to combine all input sequences to a single one that can be processed further. The digital signal processing tasks of this specific multiplexer break down into the following three steps:

1. Account for applying corrections to constant time offsets, i.e. delays that are always present due to the duration of transmission.
2. Assure the synchronicity of measured events in all signals. This includes correcting small deviations from an ideally constant frequency as well as harmonising different periodicities of the various time-discrete streams.
3. Apply transforms and filters onto the latest $n$ time frames of each input stream, as discussed later, to obtain the values to pack into a new time frame of the output stream. The packaging procedure represents the core of the multiplexing mechanism.

### 5.1.1. Delay correction

For the first step, the correction of constant delays, several experiments were made to determine those offsets of the various streams. Beginning with the sensor stream, its delay originates from several effects: the sampling and digital signal processing in the sensor hardware itself, the polling frequency of the mobile operating system, the sensor fusion process, and the delivery of those chunks of data to the mobile application. They take time and add up to the delay. In addition, the transmission of data to the computer over WiFi network plays an important role.

For this work, however, it is irrelevant to know which exact effects are involved and how they contribute to the overall result. Of importance are metrics about the total duration, from the point in time when a physical force moved the device, to receiving the corresponding data at the multiplexer. Moreover, it needs to be examined if this duration is sufficiently constant or varies under certain circumstances.

Therefore, a small measurement environment was set up with a custom made software tool that measures delays between an incoming sensor data frame and a keystroke. The mobile phone was connected to the software and repeatedly carefully dropped onto the keyboard. The sensor data frames were then inspected and the first one with a deceleration above a certain threshold was taken as the point in time the device was not only hitting the key, but already pressing it. Together with the timestamp of the key press, which has a delay in the magnitude of only hundredths of seconds, it became possible to infer the timespan between the two events. With the knowledge that a keystroke timestamp is very close to the physical event, the delay of the sensor data could be deduced. After nearly hundred measurements, it was evident that the delay is sufficiently constant and approximately 150 ms. The errors of the delay samples, as a consequence from the experiments, are normally distributed.

Following measurement of the delay of the sensor stream, the delay of the Kinect stream was to be determined as well. As stated earlier, this depth sensor delivers a handful of different streams. In the context the only one of interest is the sequence of sets of skeleton joint coordinates. The skeleton is extracted from the depth frame by software in the Kinect framework. The depth frame, in turn, was generated by a hardware unit in the device, using infra-red images of the camera for this purpose. All these processing steps require time

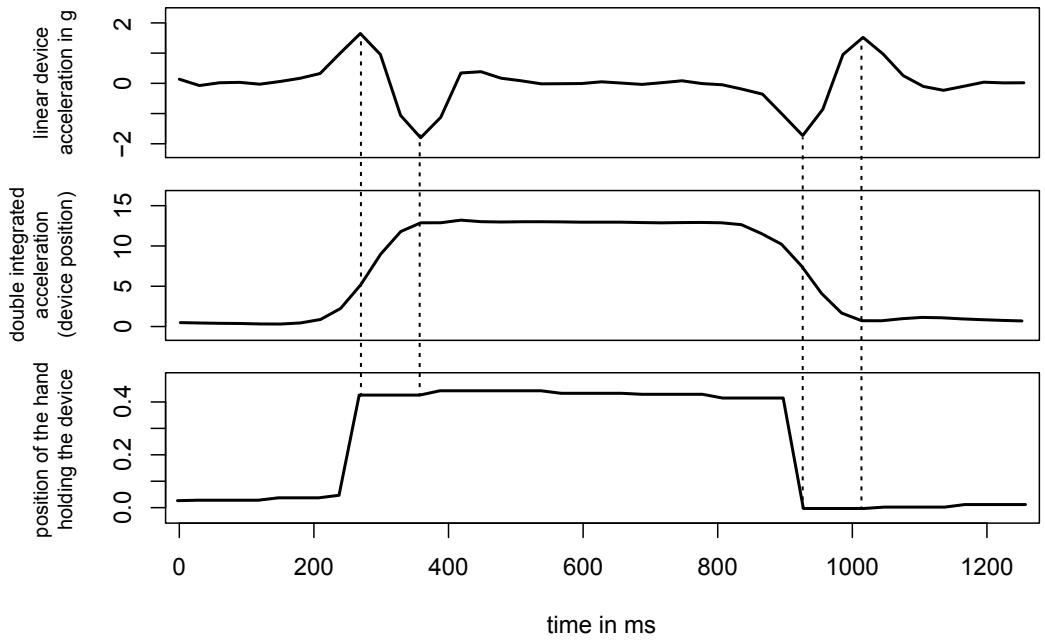



Figure 5.1.: The plots show a delay between the sensor and Kinect values at the input of the multiplexer. Note that the function values of the phone position and the hand position are not of the same magnitude due to using different coordinate systems.

and produce a certain delay between a physical event and the arrival of the corresponding skeleton data at the multiplexer. Because of this, another test was set up.

For this reason, another test was set up. While observed by the depth sensor, the device was held in the right hand, while the arm was moved up as quickly as possible. After a short pause, the arm was lowered again and the whole process was repeated several times. At the input of the multiplexer, the packet arrival times of both streams were saved as well as the two parameters of interest: the $y$-coordinate of the right hand and the linear acceleration values of the respective axis of the phone. The acceleration values are hereby rectified to cancel out the force of gravity.

Figure 5.1 shows both these values plotted over the recorded timestamps in the topmost and bottommost plots. To get a better comparison between them, the acceleration was integrated twice to get the position, after an intermediate step of the velocity (first integration). This can be seen in the middle of the two other plots. Note, that the sequence was slightly adjusted by piecewise addition of constant summands, purely for illustration purposes. The double integration multiplies small errors in the acceleration measurements to huge drifts. The sampling resolution as well as the tolerances of the sensor are not sufficiently fine grained and would not allow for plots of this precision. The temporal progress, however, was not modified by these adjustments, i.e. both the ascent and descent are located

exactly at the same point in time in the original, non-adjusted plot.

The figure shows a representative section of the series of measurements. At the first local maximum of the acceleration graph, the Kinect skeleton stream already reports the new position of the hand. At the point in time, the sensor also returns the new position, some time is already elapsed. The same can be seen in the opposite direction later in the graph. The smoothness of the double integrated graph is due to filtering in the sensor fusion process as well as due to inertia of the sensor. After inspecting a multitude of measurements, it turned out the offset value in question is constantly at about 80 ms. A slightly modified experiment using rotational values about the attitude of the device yielded almost exactly the same delay. In this second test, the changes in direction for the measured sensor values have even been easier to identify.

The Kinect therefore responds around twice as fast to movements that sensor readings do, with about 70 ms delay compared to 150 ms, respectively. It is assumed that for the most part this delay is caused by the sensor hardware and software stack on the mobile phone itself. However this will not be verified further in this thesis. To compensate for this difference in delays, the multiplexer needs to ensure it buffers incoming skeleton frames and for multiplexing with a sensor frame, takes one that is about 80 ms older.

### 5.1.2. Frequency harmonisation

As frames of the input streams that should be aggregated by the multiplexer are not guaranteed to arrive at the same frequency, this has to be taken care of before multiplexing. There are several ways to approach this issue:

1. One stream represents the primary stream. Any time a frame arrives from this stream, a new output frame is packaged and released. The frequency of the multiplexer is then equivalent to the primary stream's rate.

2. The multiplexer runs at some designated frequency and polls the last arrived frame of each stream periodically to package a new output frame. The multiplexers frequency is then both constant and independent from any of the input streams.

3. Aligning the various input frequencies by resampling them into a common period. Again, the multiplexer has its own clock generator, but this time all values of each stream are manipulated and forced onto this global period.

The advantage of the first two methods is clearly the simplicity, which particularly makes them computationally inexpensive. However, any stream which is not consistently periodic is left untouched, which could cause problems with extracting time-dependent features that require equidistant time-series as an input, as can be seen later in this chapter. This issue is addressed by the third method, that not only harmonises all involved frequencies but also regularises the periods within each stream. This can only be achieved by resampling the streams, which in turn implies that the values contained in the frames can not be taken as they arrive, but are to be interpolated in some way that conserves the original signal.

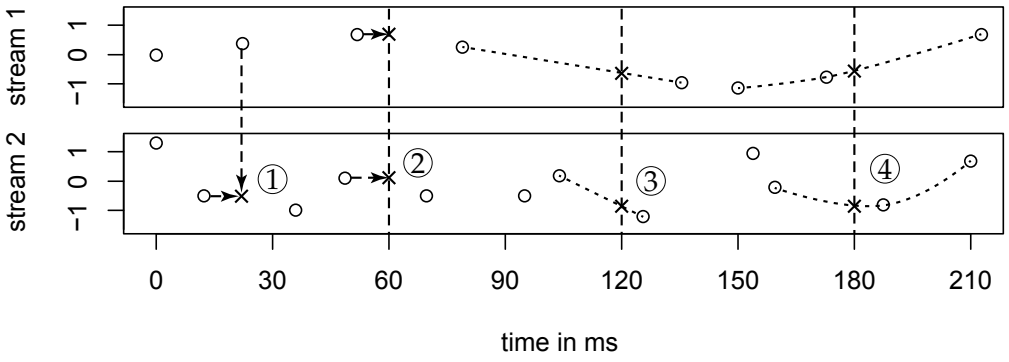



Figure 5.2.: Two non-equidistant data sequences over time. Circles depict original values of the stream, crosses mark inferred values. Four different methods of harmonisation are shown: either the first stream is set as the primary stream ① or both streams are forced to a new frequency without ②, with linear ③ or with cubic spline ④ interpolation.

Resampling in the scope of digital signal processing is usually done by upsampling the signal by an integral factor, filtering out alias spectra in the frequency domain and finally downsampling it again with a so called decimator [Por11, DKL98]. This can be approximated by interpolating between $n$ points in the time domain for those basically non-periodic signals, that are given in this context. That way, it can even be accounted for non-equidistantly spaced discrete signals. Figure 5.2 shows two such interpolation methods: linearly between two data frames ③ and via cubic spline interpolation over three or more points ④.

It can be seen, that the multiplexer needs to look back in time to a certain degree, having both data that are older and newer than the point in time the frame should be extracted. If at some stream, a newer frame did not arrive in time, it is to be extrapolated, which is not desirable at all. Extrapolation in this case means to predict the future. It is thus not guaranteed to yield useful results when applied to such non-deterministic streams. To minimize the probability of not having a newer data frame at the multiplexers disposal, the timespan to look back needs to be chosen sufficiently large. This, in turn, implies that additional delay is introduced into the overall system.

Even worse, there is a second issue with the interpolation approach: apart from creating delay, it is non-trivial to interpolate between complex higher-order objects like skeletons, quaternions and the like. Non-trivial in this context means both high computation cost and several possible approaches to interpolate each of those complex types. Each of them would have its own advantages and disadvantages and different impacts on the overall result of the machine learning process.

Complex structures pose difficulties, as do seemingly simpler sets of numbers. Interpolating between angles, for example, needs special treatment, as there are two possible ways the rotating motion could have lead from the first to the second angle. In this case, usually

a circular interpolation method is used, that is presented later in this chapter in another context. This method always interpolates over the shorter path around the circle. This however, still does not solve the problem entirely. As a contrary example, one could imagine a series of measurements of an angle $\in (-\pi, \pi]$: The sequence begins with some values near $-\pi$ and suddenly jumps to a little under $\pi$. Interpolation between the two points before and after the jump would yield an angle very close to either $-\pi$ or $\pi$, assuming the series wrapped around its boundaries, as this is the shorter path. In reality, however, it was just a rapid jump around nearly the whole circle, with a true intermediate value of around $0$. The algorithm can impossibly know which is the right choice and therefore takes the more probable one.

From a signal processing perspective, it could be said that the ambiguity was already introduced with the initial sampling where the signal had a frequency higher than the Nyquist frequency. A resampling can then impossibly compensate this, instead more than likely makes it even worse. In conclusion, it can not be guaranteed that this does not introduce errors that would not have occurred otherwise. This example illustrates, that even with an appropriate interpolation method, data is likely generated, that was actually never measured *and* is erroneous. Those values would then be taken as truth by the machine learning algorithm and could suddenly lead to false conclusions.

The issue gets even worse when regarding two sets of Euler angles in three dimensional space. Here, the gimbal lock effect can cause unexpected interpolation results. To solve this problem, quaternions are used to store the devices attitude, like already presented, which are transformed back into Euler angles at the latest possible point in time. Rotational quaternions, which lie on the unit hypersphere, can not be interpolated by treating their four components independently, because the result is no longer guaranteed to be a unit quaternion. However, there exist methods to do a linear (Slerp) and even a cubic (Squad) interpolation between two quaternions, like described by Dam et al. in [DKL98].

A naïve approach for getting an intermediate version between two skeletons can also lead to problems. When only regarding the joints separately and interpolating between the two joint positions in Cartesian coordinates, the bones between them can be skewed in length. To obtain reasonable results, the kinetic chain is to be processed hierarchically. This is realised by transforming the vector between each two connected joints into polar coordinates, interpolating the angles and length separatelly before recomposing it back to the resulting skeleton.

With all those aforementioned difficulties, a solution that drops the need for interpolation altogether is favourable over one that requires it. This is especially true in the context of this work, where tests have shown that the input streams have a sufficiently fine grained temporal resolution as well as stability in frequency, such that possible improvements by resampling can not rectify the uncertainties and errors the method introduces. Indeed, it has shown that the Kinect skeleton stream is very constant in its frame rate. The timespan between two arbitrary frames was measured to be normally distributed with a mean of 32 ms and a standard deviation of about $1.2$, meaning that there is a $90\%$ change of this duration only varying by at most 6 ms to each side. With the frequency of this stream being this stable, it was taken as the primary stream that sets the pace of the output sequence, superseding the need for an own clock generator in the multiplexer. The output

frequency, resulting to a rate of about 31 time frames per seconds is also a good choice for the following reasons:

- This corresponds to the frequency of the Kinect skeleton stream, like stated before, which renders the introduction of an extra clock generator unnecessary.
- The frequency is low enough to not be disturbed significantly by task scheduling on modern computer hardware and can therefore be processed approximately in real-time, i.e. with fixed equal periods.
- At the same time, the frequency is high enough to reach sampling rates that are required to observe movement patterns of persons. Following the Nyquist-Shannon theorem [Nyq28, Sha49], the sampling frequency has to be at least twice the largest frequency that should be reconstructable from the input signal. Movements of interest for this topic happen in the magnitude of at most tenths of seconds and therefore require a sampling rate of approximately 20 times per second or more.
- If the resulting output stream needs to be visualized in any form, a frequency of about 25 to 30 images per second is just sufficient to produce a fluid moving picture, as this is also the common range of frame rates of video sequences [RM00].

The sensor stream, that is sent over the network to the computer to arrive at the multiplexer, was measured to have a stronger deviance from the ideal equidistant pattern. This is mostly due to side effects of the wireless network transmission. Also, on the device a pulling rate for new sensor values of 50 ms was chosen. At higher frequencies, the signal begins to become staircase-shaped, as it reaches the limits of the underlying subsystem of sensor hardware and software stack. Although this is the fastest possible rate, it is still slower than the chosen output frequency of the multiplexer, in particular by a factor of about $1.5$.

Due to these more disadvantageous prerequisites, it was tested whether treating specifically this stream with interpolation yielded improved results. The effect proved to be negligible, however. For the sake of simplicity, the overall harmonisation method was decided to just take the last respective frame of the sensor stream, leaving it unmodified for packaging.

### 5.1.3. Packaging

Having adjusted all input streams by their constant delay and corrected their frequencies to be equal and synchronous to each other, the actual multiplexing can be done by packaging them into the output stream. Right before this is the only possible time of calculating features that depend on information of past time frames, as they are now equidistantly spaced. The $n$ last frames with $m$ values each are then reduced to $k < mn$ time-dependent features, using methods elaborated later in section 5.3. After each multiplexed output frame, the majority of those input values stay the same, with only being joined by $n$ new values of the next input frames and dropping the oldest $n$ values in return. This way it represents a sliding window over the input streams. Figure 5.3 illustrates this process.

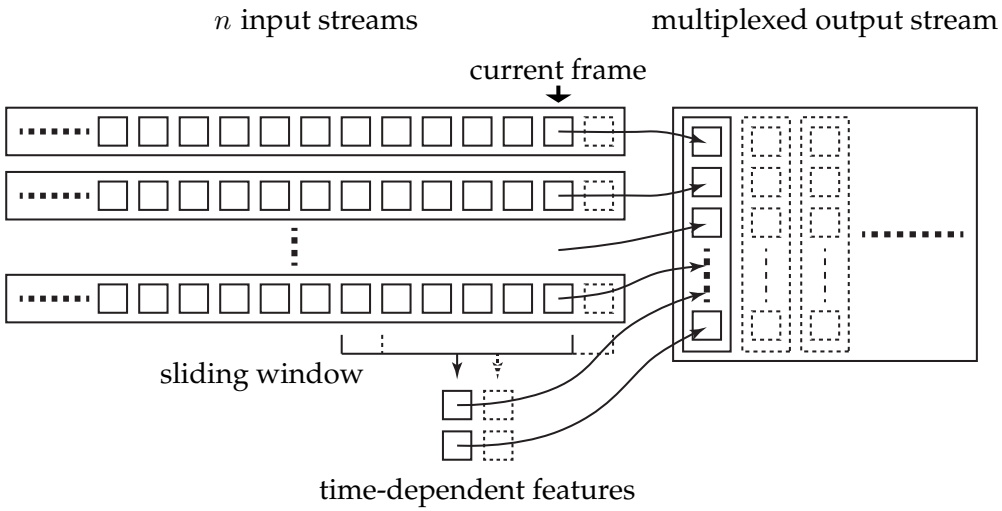



Figure 5.3.: Adjusted and harmonised input streams with their frames being filtered and packaged into frames of the output stream in the multiplexer. A sliding window over the $n$ newest time frames is used for extraction of time-dependent features.

Finally, a stream of entirely independent frames was generated. The effects like delay and irregularity have been compensated and therefore abstracted from the following processing steps. Required information about past measurements is not lost, but included in those time-dependent features that have been calculated. This also reduces dimensionality drastically. The multiplexed frames can be processed further without need to keep the order. This enabled an approach to do tests on the regression results later in the evaluation: the set of multiplexer output frames can be arbitrarily partitioned without any problem.

In the following, different features that were studied in the context of this work are presented, together with the methods to calculate them. Because of the multiplexer, those features can be regarded and computed detached from chronological dependencies. The first feature that is presented, which is also one of the most important ones, is going to be the heading.

## 5.2. Heading calculation

The sensor fusion algorithm provides attitude information, that has all 3 rotational degrees of freedom in the euclidean space. It outputs a rotation quaternion, as this is free of effects like gimbal lock. A very key feature that is included in the attitude is the heading information, also called bearing. This is a single angular value that specifies the direction the device is facing at, on the $x, y$-plane, which is the ground plane. The heading is essential not only to allow to reason about the direction the person is facing at, but also to make

the attitude invariant of the bearing. This is an important simplification for the machine learning process, as can be seen later.

However, there is no common definition for the derivation of such a value from the attitude. A function that will yield the heading, $f_h\colon \mathbb{H} \to (-\pi, \pi]$ basically projects a unit 3-sphere in 4-dimensional space onto the unit circle, a compact set which in turn can be easily mapped to the $(-\pi, \pi]$ interval, finally resulting in a value of 1 dimension. There are many possible ways to construct such a function. To get results that are meaningful, i.e. follow the intuition behind the bearing, the following requirements to such a mapping were defined:

- The heading must be consistent with the rotation around the $z$-axis. This means, given an arbitrary attitude, represented by the unit quaternion $\mathbf{q}_a$, its rotation around the $z$-axis by some arbitrary angle $\theta$ has to add the same angle to the corresponding heading. Therefore, $\mathbf{q}'_a = \mathbf{q}_r\mathbf{q}_a \Rightarrow f_h(\mathbf{q}'_a) = f_h(\mathbf{q}_a) + \theta \pmod{2\pi}$ must hold, where $\mathbf{q}_r = \cos(\frac{\theta}{2}) + 0i + 0j + \sin(\frac{\theta}{2})k$ is the unit quaternion that defines a rotation around the $z$-axis with angle $\theta$. The only exception is given if the attitude is located right at a singularity of $f_h$ and therefore $f_h(\mathbf{q}_a)$ is undefined.
- The required function $f_h$ has to be stable for almost all attitudes $\mathbf{q}_a$, i.e. small changes in any of the quaternions dimensions should lead to only small changes in the final heading. This is essential for obtaining a meaningful mapping, as functions with deterministic chaotic behaviour are practically useless, as the input values are subject to errors and noise, which would then be multiplied in the output values.

Nonetheless, in the second requirement the stability can not be assured for each and every possible attitude, because there will always be poles in such mappings. This is due to effects like the gimbal lock, that are unapparent with rotation quaternions but emerge when projecting them onto lesser-dimensional spaces. The target is therefore to design a mapping that has those singularities at locations that are typically not touched to reduce occurrences of jumps in the output values. This is possible, because the probability distribution of attitudes that are likely to occur when carrying a mobile phone in the pocket is not evenly spread over all possible values. Consequently, a metric was searched for, that models this matter of fact.

### 5.2.1. Characteristic vectors

The starting point for modelling such a function is to look at typical attitudes of the mobile phone. One of them is when the device lies flat on the ground. It can be rotated around inside this plane like a compass. One could argue that it is reasonable to say it points towards the upper side of the display then. This characteristic vector that points from the bottom to the top inside the display plane shall be called $\vec{\mathbf{w}}$. Considering the two other orthogonally aligned rotation axes, the heading should stay constant while rolling the device around the $\vec{\mathbf{w}}$ vector, as well as pitching it up and down to a certain degree.

The conclusion according to this would be that the vector $\vec{\mathbf{w}}$ just has to be projected onto the ground plane and the heading can be directly inferred by the direction this vector

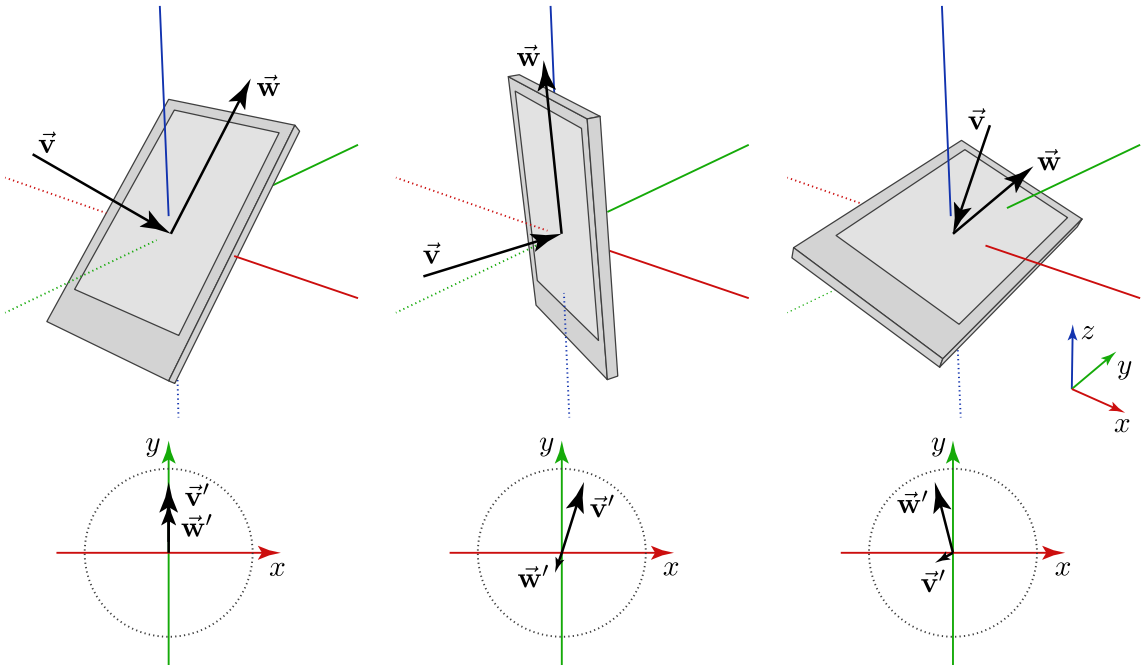



Figure 5.4.: Different device attitudes together with the respective characteristic unit vectors and their projections onto the $x, y$-plane are shown. Unit circles are depicted with a dotted line.

is pointing at. This certainly is a practicable model. However, it suffers from a major disadvantage: the singularity of the mapping appears right when the device is standing upright. In this situation, the vector is orthogonal to the plane which implies that the projection contracts to a point. This very attitude of the device, however, is a very common case.

To compensate for this, the consideration goes further by focusing on this case isolated from the previous one. A reasonable heading of an upright standing device can be seen in the direction of a vector, called $\vec{v}$ in the following, that pierces through the display to the backside of the device, i.e. the direction a user is looking at if he watches the display. Here, again the same issue appears, except for the singularity occurring when the phone is lying flat on the ground.

### 5.2.2. Reduction to single angle

The solution for avoiding those singularities at common attitudes is simply to combine the two methods, using the first one if the device is oriented approximately parallel to the ground or the second one if it is rather perpendicular to it, respectively. Orientations in between those can then be interpolated to create a smooth transition. For such a linear interpolation, also called lerp, a metric is needed to determine which vector should

have more influence in a particular case and how much exactly. Taking the length of the projected vector $\vec{\mathbf{w}}'$ has proven to be an applicable choice for this.

It resembles the scheme mentioned just before: If the device is nearing an orientation parallel to the floor, the projected vector approaches its maximum length, while shrinking to a minimal length of $0$ when reaching the upright position. The opposite is true for the projected vector $\vec{\mathbf{v}}'$. If laying on one side, both projections of the vectors have maximal length. For this reason it was decided to just regard the $\vec{\mathbf{w}}$ vector as the primary one and let the secondary only grow in influence, if the primary is shrinking at the same time. It seemed natural to let both vectors $\vec{\mathbf{w}}$ and $\vec{\mathbf{v}}$ be unit vectors to make sure the projections euclidean norms are $\in [0, 1]$ and can be used as interpolation weights. Figure 5.4 shows a device in certain attitudes, the corresponding vectors and their projections to the floor plane to visualize the whole process.

To formalize the described method, the vectors $\vec{\mathbf{w}}$ and $\vec{\mathbf{v}}$ are now found by taking the basic unit vectors $(0, 1, 0)^\top$ and $(0, 0, -1)^\top$ and rotating them with the given rotation quaternion:

$$\begin{aligned}\mathbf{q}_a(0i + 1j + 0k)\mathbf{q}_a^* &= w_x i + w_y j + w_z k \\ \mathbf{q}_a(0i + 0j - 1k)\mathbf{q}_a^* &= v_x i + v_y j + v_z k\end{aligned}$$

Where $\mathbf{q}^*$ denotes the quaternion conjugate of $\mathbf{q}$ and $\mathbf{quq}^*$ is the Hamilton product that rotates a vector $\mathbf{u}$ with the rotation given by $\mathbf{q}$. The projections onto the $x, y$-plane are then given by omitting the $z$-components to form the 2D-vectors:

$$\begin{aligned}\vec{\mathbf{w}}' &= (w_x, w_y)^\top \\ \vec{\mathbf{v}}' &= (v_x, v_y)^\top\end{aligned}$$

The heading angle of both those 2D-vectors can be found with the help of the two-argument arcus tangent function $\operatorname{atan2}(y, x)$, which directly yields an angle in the interval of $[-\pi, \pi]$ and is defined as:

$$\operatorname{atan2}(y, x) := \begin{cases} \arctan\frac{y}{x} & \text{if } x > 0 \\ \arctan\frac{y}{x} + \pi & \text{if } x < 0,\ y \geq 0 \\ \arctan\frac{y}{x} - \pi & \text{if } x < 0,\ y < 0 \\ +\pi/2 & \text{if } x = 0,\ y > 0 \\ -\pi/2 & \text{if } x = 0,\ y < 0 \\ 0 & \text{if } x = 0,\ y = 0 \end{cases}$$

The weight for the interpolation is given by the length of $\vec{\mathbf{w}}'$, i.e. $\alpha = \|\vec{\mathbf{w}}'\|$, as already mentioned before. The heading function is therefore defined as:

$$f_{h1}(\mathbf{q}_a) = \alpha \operatorname{atan2}(w_y, w_x) + (1 - \alpha) \operatorname{atan2}(v_y, v_x)$$

There is a problem with this function, however: it does not take into account the cyclic nature of the two angles to interpolate. As a simple example, the arithmetic cyclic mean of

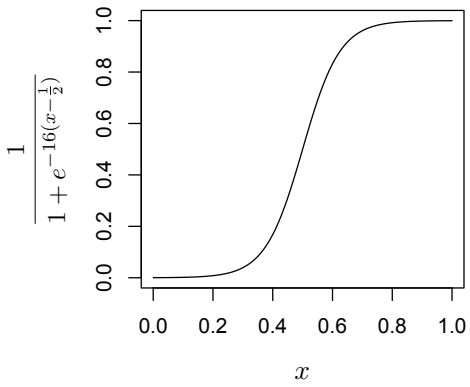



Figure 5.5.: The adjusted logistic function for refining the linear interpolation weight. $\omega$ is set to $16$ for this plot.

$-\pi$ and $\frac{\pi}{2}$ is $\frac{3\pi}{4}$ instead of $\frac{-\pi}{4}$, which the linear interpolation would yield. Such a wrapped lerp is equivalent to first interpolating the vectors and extract the heading out of the resulting vector, i.e. interpolating in cartesian coordinates instead of polar coordinates. The function is then defined as:

$$f_{h2}(\mathbf{q}_a) = \operatorname{atan2}(\alpha w_y + (1-\alpha)v_y, \alpha w_x + (1-\alpha)v_x)$$

The impacts of a non-cyclic interpolation will be discussed in more detail later on and are going be visualized with the aid of an example.

### 5.2.3. Further refinement

With linear interpolation, the vector with smaller weight can influence the result much stronger than intended, as can be seen in an example later on. A possibility to counteract this would be to adjust the weight $\alpha$ used in the interpolation process, such that values of $\alpha$ near the boundaries of $[0, 1]$ are pushed further against these. A specially modified version of the general logistics function $P(x) = 1/\left(1 + e^{-x}\right)$, that is not centred around $0$, but around $\frac{1}{2}$ is designed to achieve this. It maps values of $\alpha$ in the interval $[0, 1]$ to itself (figure 5.5 shows a plot of the function). It can be used to produce a new weight $\alpha' = P'(\alpha)$ that can then be used inside the lerp function. It is described by:

$$P'(x) = \frac{1}{1 + e^{-\sigma(x-\frac{1}{2})}}$$

Just like the general logistic function, it has two asymptotes, as the function converges to $0$ for $x \to -\infty$ and $1$ for $x \to \infty$, respectively. This function is used to adjust the weight of the linear interpolation between the two angles, where the normal linear interpolation would have to much weight on the farther located value, if it is very large. It maps the interval

of valid interpolation weights (not regarding extrapolation) onto the same interval again, but giving values near 0 or near 1 a stronger weight.

As it showed in several experiments, the horizontal scaling factor $\omega$ proved to produce the most suitable results for this purpose in the range of about $12$ to $18$. In practice, the selection of this parameter was a trade-off between the slope of the function at around $x = \frac{1}{2}$ and the actually beneficial weight that is added by applying the function. With choosing smaller values, the function becomes very similar to the identity map $f(x) = x$ in this interval, which nullifies the purpose of the whole process. Larger values for the scaling factor $\omega$ lets it approximate a function that cuts input values at a threshold at $x = \frac{1}{2}$ and returns only values very near to 0 or 1. This, in turn, nullifies the need for an interpolation altogether.

The factor $16$ was therefore chosen, as it lies in between those boundaries and has the advantage of being a power of two. This way it allows for easy multiplication with the value in the implementation, as the floating point representation in current computers is based on a binary mantissa and exponent. The addition of 4 to the values exponent is therefore equivalent to the multiplication of the number with 16, moreover even leaving the mantissa untouched and avoiding numerical errors completely.

### 5.2.4. Discussion and analysis

To illustrate the practical advantages of the aforementioned method for determining the heading, common positions and transitions in between them will be elaborated in the following. Figure 5.6 helps hereby, with showing exemplary time series of bearings of the two projected candidate vectors $\vec{\mathbf{v}}'$ and $\vec{\mathbf{w}}'$ and the weight variable, deviated from the euclidean norm of $\vec{\mathbf{w}}'$, i.e. $\|\vec{\mathbf{w}}'\|$. These three parameters are calculated from real attitude measurements via the just presented method and are plotted to the left-hand side. They are then used for the interpolation results. The results of each of the three methods can be seen in the right-hand plots, listed underneath each other for direct comparison.

First off, the assumption, that linear interpolation between the two candidate vectors is sufficient, can be rejected because it generates inconsistent result angles for some attitudes. This is the case, if the projected vectors do not point in the same direction and especially becomes apparent if the weight is near $0.5$, where both fractions have influence on the result. This can be easily explained by considering that this naive approach treats bearings, i.e. cyclic quantities, like elements of a linear interval. The impact is severe: it violates the basic requirement of the resulting heading value being invariant under rotating around the $z$-axis.

Figure 5.6 shows an example of this in about the first 3 seconds of time. The upright standing device was slightly tilted to the side for this purpose, i.e. around the $y$-axis. This way, the two vectors point in directions $90°$ displaced from each other and being about equally weighted. With this attitude held stable, the device was rotated around the $z$-axis to move alongside different headings. It can be seen in the resulting plot at point ①, that there is an instant jump by a magnitude of $\pi$ as the angle of $\vec{\mathbf{w}}'$ wraps around

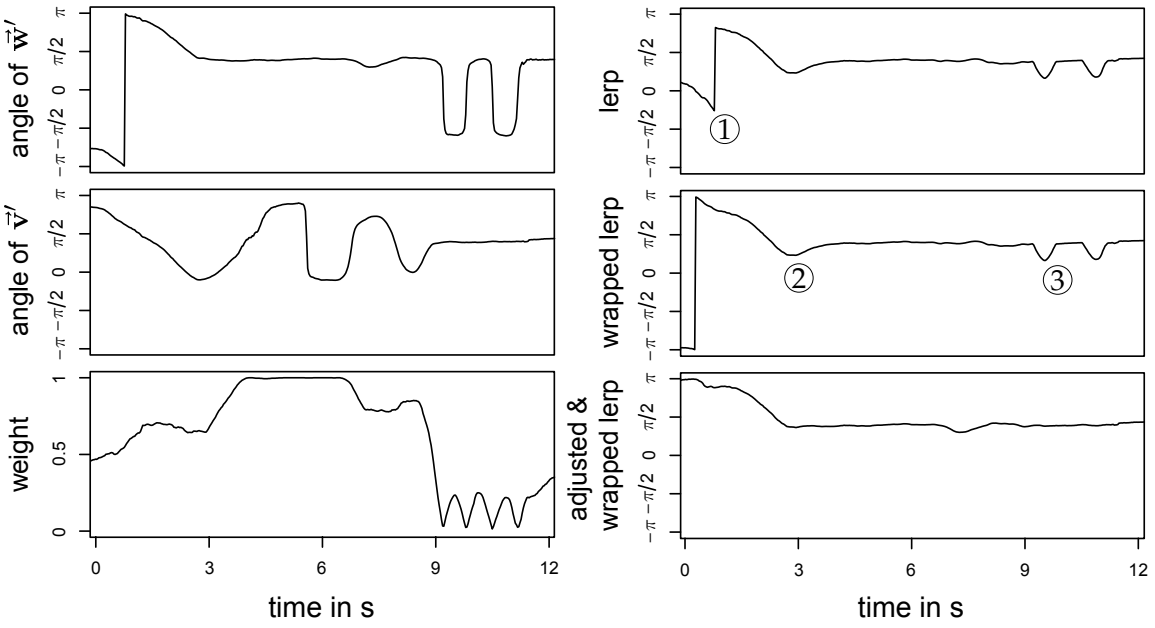



Figure 5.6.: Example measurements of transitions which show critical cases of the heading calculation. The input parameters, used by all three approaches are given, together with each methods resulting values for comparison.

its boundaries. Even though it is just a slight movement, it turns the resulting bearing completely around to the opposite side.

This issue is addressed by an interpolation method that takes into account the cyclic nature of the involved angular values, as presented before. This wrapped lerp method, like shown in the middle right plot, has a completely continuous line for the given input sequence. The only seemingly discontinuous location right at the start is just due to the representation of angles in this particular plot.

The example continues after 3 seconds with holding the heading of the device constant, laying the device flat on the ground and rotating it back and forth around its longitudinal axis, i.e. the vector $\vec{w}$. In a third phase, it was put upright again to slightly tilt it forwards and backwards between the 9 and 11 second marks. Both those phases have a strong weight on either of the two vectors, first on $\vec{w}'$, then on $\vec{v}'$. However, the markers ② and ③ show various points which are distorted from the expected constant bearing value. This is due to the rapid jumps of the angle around $180°$ while it passes the pole, where its corresponding original vector is aligned orthogonally to the ground plane. Even though this vector has minimal weight and therefore the other one is privileged to be used to determine the bearing, those large jumps influence the result and produce those little bumps in the plot.

The solution to this problem was presented as a readjustment of the weight parameter

with the modified logistics function. The results of the application of this function before applying the wrapped lerp can be seen in the bottom-most plot on the right-hand side. The line of bearing values over time is now continuous and very smooth, without even applying any lossy filters or frequency passes. This way it is stable and reacts to changes rapidly at the same time, because it is still time-invariant.

Now that the heading of the phone is known, it can be used as the starting point of calculation for several regression features, as can be seen in the following section.

## 5.3. Regression feature extraction

One of the most important goals of this work was to find significant features, that allow for a well suited linear regression model. They can be classified into two groups: the regressor features that are established from raw sensor values and the response features that will be used as a target of the regression hypothesis. In the following, the set of regressor features will be detailed. The question of which of these features are better suited for the overall goal will be left open for the evaluation chapter.

### 5.3.1. Elementary features

The very first feature was already presented: the heading angle of the mobile device. This is the most important feature for inferring the users heading, as further described later in section 5.4.2. It is furthermore used for making some of the other features independent of the heading.

The next basic feature is the attitude of the device, which is also one of the most essential ones. The mobile device can be regarded 'attached' to the users body as he carries it in his pocket. This way, each pose is invariant of any absolute heading of both the user and the device, as both are in the same local reference frame. If this frame is rotated in the ground plane, both the person and its phone are rotated at the same time. This forms an equivalence class containing all possible headings. As a consequence, this dimension can be disregarded for pose estimation purposes and only be used for the user heading estimation. The omission of this dimension showed to have a large positive impact on the stability and generalisability of the pose estimation. Any pose that is trained looking in a certain direction is recognised in any other direction, too, without the need for being trained separately. This is a prime example for the exploitation of domain knowledge: it is a simplification that the machine learning algorithm can not make, as it does not know the whole system and its properties it is embedded into.

However, since the heading is not only an isolated dimension, but is also contained inside the attitude information of the fused sensors, it has to be filtered out of there in order to get an attitude that is invariant of the heading. This is possible by using the isolated heading angle to rotate the attitude in the opposite direction by the same amount, effectively cancelling out the heading. The heading is defined to represent a rotation inside the ground plane, which therefore has the planes normal vector as the axis of rotation, which is the

$z$-axis. Its rotation quaternion is therefore described by $\mathbf{q}_r = \cos(\frac{\theta}{2}) + 0i + 0j + \sin(\frac{\theta}{2})k$, with $\theta$ being the heading angle. The heading-invariant attitude quaternion $\hat{\mathbf{q}}_a$ can finally be determined by rotating the attitude $\mathbf{q}_a$ by the inverse of the heading $\mathbf{q}_h^*$:

$$\hat{\mathbf{q}}_a = \mathbf{q}_h^* \mathbf{q}_a$$

It might seem strange that accelerometer readings are first combined with magnetometer values via sensor fusion, just to separate both again later on. Yet, the presented separation is not the exact inverse process of the fusion. There are many details that are gained using these two processing steps. The triaxial magnetometer data alone does indeed hold the heading information, but it is much harder to extract without knowing the orientation of the device. It would imply that the orientation has to be deduced from the magnetometer measurements as well, which is possible but much more unreliable than using the accelerometer. It is also true that the gravity vector of the accelerometer already represents a form of heading-invariant attitude, but this in turn misses the stabilisation of the gyroscope. Furthermore, the sensor fusion is required anyhow, as the attitude is used for other features, presented in the following. As a last reason, the very specially designed heading calculation yields a result which is slightly different from the raw magnetometer heading. Using this angle together with the fused sensor attitude for the calculation of $\hat{\mathbf{q}}_a$ preserves the best properties of both to avoid singularities and have a smooth, responsive and accurate attitude stream.

Having the heading-invariant attitude, it is finally converted to Euler angles. These three angles will then be used as regression features in the machine learning process. The conversion of a quaternion $q = q_0 + q_1 i + q_2 j + q_3 k$ into the pitch $\phi$, roll $\theta$ and yaw $\psi$ angles is, like described in [Die06], defined as

$$\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} \operatorname{atan2}\left(2(q_0 q_1 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)\right) \\ \arcsin\left(2(q_0 q_2 - q_3 q_1)\right) \\ \operatorname{atan2}\left(2(q_0 q_3 + q_1 q_2), 1 - 2(q_2^2 + q_3^2)\right) \end{pmatrix}$$

### 5.3.2. Statistical features

The three aforementioned features, i.e. the attitude 3-tuple of Euler angles, are in turn going to be the starting point for further features that are dependent over time. The following statistical features are formed using all of these 3-tuples in the time window, which is $n$ elements large:

- The mean over all $n$ sets for each angles
- The standard deviation over all $n$ sets for each angles
- The *Pearson Correlation Coefficient* [Lee12] between each two different angles over all $n$ sets

This results in nine additional features: the mean and standard deviation of the past $n$ values of pitch, roll and yaw as well as three features about the correlation between the past $n$ pairs of pitch-roll, pitch-yaw and roll-yaw respectively.

For an efficient processing, the calculation of these features was implemented by directly accessing *circular buffers*, which are in turn iteratively filled with the most recent attitude angles. Using this data structure, values of the sliding window do not have to be copied each time, but are staying in place while the newest value overwrites the oldest one in each time frame, thus avoiding many memory accesses.

### 5.3.3. Periodicity features

Additionally to these statistical features, which are calculated with regard to the time domain, there will be three ones that are extracted from the frequency domain, one for each attitude angle. The goal is to get information about periodic movements that are made, for instance walking or running. These features help estimating the state of the leg opposite to the one the mobile device is located. Therefore, the $n$ values of the time window are taken for each attitude angle and converted into their discrete spectrum via a *Discrete Fourier Transformation* (DFT) [OSB99].

Efficient implementations for such a conversion are available with the so called *Fast Fourier Transformation* (FFT) [Bri88]. Furthermore, it is possible to calculate the spectrum over the sliding window even more efficiently. In each time frame, only two values in the time domain are changed as the oldest one is dropped at the tail-end and a new one is appended in front. By exploiting the circular shift property [Spr88], a *Sliding Window DFT* [JL03] can use previous calculations of a spectrum for updating it to a new one instead of having to calculate the whole transformation each time.

In the frequency domain, the periodicities can be seen by finding the largest of the discrete frequency parts. However, if there is no significant movement, noise of the measurements results into all frequencies being about equal which could lead to the selection of a very low or very high frequency. As these frequencies contain no useful information in this context, only frequencies in the middle are regarded. The range of human walking frequencies starts at about 1 Hz, while running reaches up to about 162 steps per minute [CF86], which is nearly 3 Hz. All frequencies outside this or a slightly larger range are omitted. Due to the nature of FFT, which returns $\frac{n}{2} - 1$ discrete frequency buckets for $n$ input values, the time window has to be chosen sufficiently large to allow for an adequate frequency resolution. Furthermore, if no single frequency has a noteworthy amplitude, i.e. there is no periodicity of interest in the signal, the feature should take a zero value. Therefore, a suitable threshold has to be found in experiments.

There exist more advanced methods for determining significant periodicities in signals, which use extended analysis of the signal in time domain [VM04] or utilise the circular autocorrelation of a signal to refuse false periodicity candidates or refine the frequency resolution for true candidates [VCY05]. These methods were not tested in the course of this work, however, due to their complex implementation.

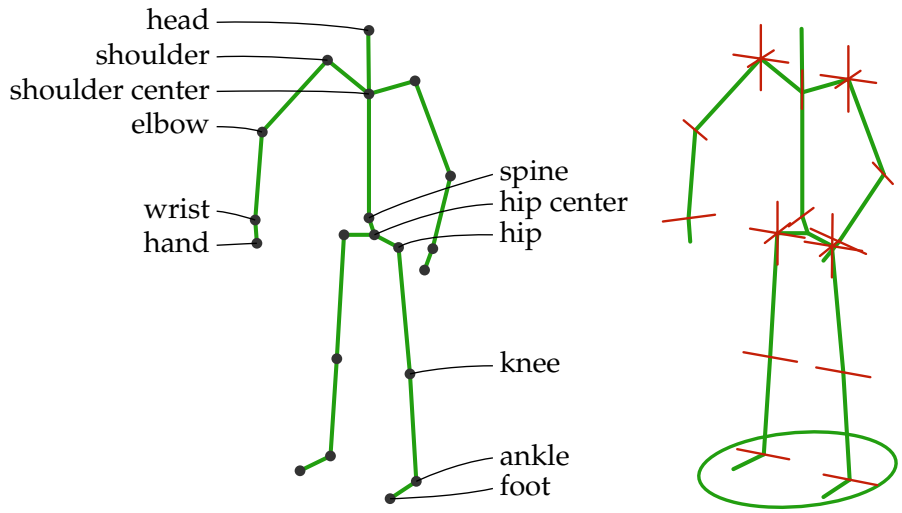



Figure 5.7.: The skeleton joints and their identifiers. Pairs of symmetric joints are distinguished by the additional identifier suffixes 'right' or 'left'. The degrees of freedom of each joint with their rotational axes are shown in the right illustration.

## 5.4. Response feature extraction

Having discussed only features that are inferred from the mobile devices sensor data, representing the regressor values for the machine learning process, a key component is still missing: the set of response values that will be used to supervise the learner. They represent the true body posture variables that shall be derived from the sensor measurements alone. This is the feedback needed to train and fit the regression model as well as to verify the accuracy of the results and test the statistical significance of various dimensions in the fitted model. As outlined before, the Kinect skeleton stream will be used for this purpose to extract features that relate to the regressor parameters.

Therefore, the principle topology of such a skeleton data structure will be outlined first. Each skeleton is formed by a set of joints, each having holding the cartesian coordinates of the joint in 3D space. Every joint has assigned a unique identifier out of those 20 defined ones that can be seen in the left part of figure 5.7. The hierarchical structure of the skeleton is given implicitly with these IDs. The bones connect those joints and represent the vertices between the nodes of the tree formed skeleton, as can be seen in the illustration. It starts at the root joint, the 'spire', which holds together two branches for each the upper and lower parts of the body, until these fork into the left and right limbs as well as the head, respectively.

### 5.4.1. Revolute joint model

Each of the joints between two bones acts like a hinge that enables motion in the skeleton. A consecutive series of such joints, which are connected by bones, form kinematic chains. Examples for these chains are all four limbs. Putting everything together leads to a 1 degree of freedom (DoF) revolute joint skeleton model. Both shoulders and hips are special cases, as they have 3 degrees of freedom and can be modelled by collapsing three consecutive 1 DoF revolute joints in the same position. Another way of dealing with those special joints is to use quaternions for modelling spherical joints, like presented in [MR09]. The rotational axes of each joint in such a model are illustrated in figure 5.7 to the right. These models are used within a lot of motion capturing techniques.

This model enables simplifications by reducing the number of values that are needed to describe it. Instead of requiring 60 vector component values in the joint coordinate representation, it can be boiled down to 22 angular values in the revolute joint representation. It should be noted, that this applies only when assuming that the lengths of all bones stay constant, which is a plausible assumption. This model transformation is similar to the conversion of vectors from cartesian into polar coordinates. Furthermore, this allows to separate the following four principal properties of the skeleton from each other, which are inseparable in the cartesian joint model:

- **Location.** The position of the person that is tracked in euclidean space. This can either be relative to a certain known location or absolutely in a given common coordinate system. As positioning information is not part of this thesis, it can be dropped in the context of this work. However, it is important to note that the skeleton can be regarded independent of the location.

- **Orientation.** The basic orientation of the skeleton inside the ground plane, that therefore corresponds to the heading information of the phone. A more generalized approach would be to allow the orientation to express the complete attitude in 3D space instead of only inside the ground plane. In this work, the orientation will be defined by the normal vector of the plane that is specified by the spine, right- and left-shoulder joints. This is reasonable, because this plane represents the torso of the skeleton and the shoulders give the only possible approximation of the direction the person is looking at.

- **Posture.** Having stripped off location and orientation of the skeleton, the core information is left over, that is the actual posture. This is described by the state of the kinematic chains formed by the revolute joint model.

- **Stature.** Besides the three dynamic properties, this contains the static information of the skeleton, which is the length of the bones. This stays constant for a specific person, but varies between people. This is the reason, it does not have to be stored in every single time frame and can be disregarded completely for the machine learning process.

### 5.4.2. Relevant features

As described, stature and location information is not going to be used for features that are passed on to the learner. The orientation, on the other hand, is one of the core goals of this work. Also, there are several posture related features, inferable from sensor data, that will be used further. All these will be discussed in the following.

Starting with the orientation, the goal is to find a feature that can be used for linear regression. Recall that the multiplexer has already packaged the current skeleton and all related sensor features, including the calculated heading, in a single frame. This means, the phone heading is known and the corresponding heading of the skeleton can be deduced from the shoulder joints. There are now two possible ways of dealing with this data:

The first possibility would be to take the skeleton heading angle $h_{\text{skel}\,j}$ as the response variable and make it dependent on the mobile device heading angle $h_{\text{dev}\,j}$ as well as on $k$ sensor features $s_{j,1} \dots s_{j,k}$. Again, $j$ denotes the index of the time frame. The dependence on sensor features is really important here, as they contain implicit information about the pose, which in turn influences the deviation of the phone heading from the skeleton heading. Therefore, they have a strong relation to each other that has to be reproduced in the model. This way, a linear relation $h_{\text{skel}\,j} \approx c_0 + c_1 h_{\text{dev}\,j} + c_2 s_{j,1} + \dots + c_{k+1} s_{j,k}$ is assumed, with coefficients $c_0 \dots c_{k+1}$ that need to be fitted by the learning algorithm. Note, that the $\approx$-sign means in this context 'equal except for an error $\varepsilon_j$', where the $\varepsilon_j$ are the differences between the measured response and the value predicted by the model, also called *residuals*.

Another way would be to take the difference between the two heading angles of the phone and skeleton, from now on called 'heading delta' $\Delta h_j$, and let it depend on the sensor features. This models the correlation between the deviation of the two heading values from each other and the pose directly. In fact, those two approaches are equivalent, as it corresponds to this rearrangement:

$$\begin{aligned} h_{\text{skel}\,j} &\approx c_0 + c_1 h_{\text{dev}\,j} + c_2 s_{j,1} + \dots + c_{k+1} s_{j,k} \\ h_{\text{skel}\,j} - c_1 h_{\text{dev}\,j} &\approx c_0 + c_2 s_{j,1} + \dots + c_{k+1} s_{j,k} \\ \Delta h_j &\approx c_0 + c_2 s_{j,1} + \dots + c_{k+1} s_{j,k} \text{ iff. } c_1 = 1 \end{aligned} \tag{5.1}$$

Note, that the last step is only possible if $c_1 = 1$. This means, that independently from the direction, where the person is facing at, the mobile device is always some angle $\alpha$ displaced from that. Different postures may result in different displacement angles, but if the pose is held constant, $\alpha$ is constant as well. In other words again, if the device is attached to the person and is rotating together with the person, this exact requirement $c_1 = 1$ is met. As this is the principle assumption, this work builds upon, this can be taken as given.

The second method turned out to be the better choice. Not only does it reduce the dimensionality of the model by one, it also solves a problem with the heading angles: as the values are $\in (-\pi, \pi]$, a complete tour around the circle results in a discontinuous graph. Due to the two angles being offset from each other to some degree, one of the two angles

reaches this discontinuity earlier as the other, resulting in a linear relationship only if it is specially accounted for this cyclic nature. The second method was finally chosen, because the introduction of a special treatment for cyclic values into the linear regression model does not yield any additional advantage, quite the contrary only making the model more complex.

After having fitted the model, and therefore the coefficients $c_0 \dots c_{k+1}$ are determined, $\Delta h_j$ can be predicted by the model (equation 5.1) using the sensor values $s_{j,1} \dots s_{j,k}$. As $h_{\text{dev } j}$ is inferred from the sensor values as well, the heading of the person is then finally given by adding these together:

$$h_{\text{skel } j} = \left(h_{\text{skel } j} - h_{\text{dev } j}\right) + h_{\text{dev } j} = \Delta h_j + h_{\text{dev } j}$$

The next features that should be trained correspond to the hip joints. It seems reasonable to do this with at least the one leg, where the device is placed inside the pocket, but the other one should also be examined. The results of this will be shown later on in the evaluation part of this thesis. For this purpose, the angles between the two thighs and the torso are calculated.

The axis of rotation is set as the line through both hip joints. The geometric planes for the thighs and the torso are described by the knee and the shoulder centre joint, respectively, as well as the rotation axis. The angle $\alpha_{\text{leg}}$ between the two planes, which will be the resulting angle for the leg, is found using the normal vectors of both planes: $n_1$ and $n_2$. As the direction of those normals depends on the order of the points from which they are calculated, the order is chosen in such a way, that the normals face to the front of the skeleton.

In 3D space, angles greater than $\pi$ are only meaningful, if a reference direction to look at the angle is given. This will be defined to be the direction of an imaginary observer, who is farther away from the right hip than from the left. The information, whether the angle is $> \pi$ can now be determined by deciding whether the cross product of both plane normals $n_1 \times n_2$ is faced in the same direction as the vector which is pointing from the left to the farther located right hip. This decision is always possible: The vector difference between the two hips represents the rotation axis, which is never of zero length for a valid skeleton. The cross product is by definition orthogonal to both plane normals and therefore describes the intersection line between those planes, which in turn is also the rotation axis. The information gained this way will be defined as $\alpha_{\text{sgn}}$, which is $-1$ if $n_1 \times n_2$ points towards the observer and $1$ if pointing away. If both normal vectors are equal, $\alpha_{\text{sgn}}$ is defined to be $0$. The angle between two planes is given by the inverse cosine of the dot product of the two normals. The final angle in the interval of $[0, 2\pi]$ is therefore:

$$\alpha_{\text{leg}} = \pi + \alpha_{\text{sgn}} \arccos(n_1 \cdot n_2)$$

Using this method, all relevant angle features of the revolute joint model can be calculated analogously. These features are also well suited for linear regression, as they are angular values that map well to the also circular attitude features of the sensor input. Also, for

these features, it does not have to be accounted for any circular discontinuities, as the angles are restricted to certain intervals given by the anatomy of the human skeleton. Therefore, it is guaranteed that the angles are at most $\in [0, \pi]$, or most likely even in a smaller subset of this interval and can not suddenly wrap around these boundaries. This will be further discussed later in the evaluation part.

### 5.4.3. Skeleton estimation from features

The transformation of the presented features back into a skeleton will not be exact, as some information was omitted, that would be required for a precise reconstruction. This is inevitable anyhow, because some components which are essential for the skeleton can impossibly be inferred from the mobile phone sensors. Given these limitations, it still can be estimated. The process to achieve this is laid out in the follows steps, which is basically the inverse of the feature extraction:

1. Generate a normalized skeleton based on the stature
2. Apply all known features of the kinematic chain
3. Rotate according to the heading information
4. Translate coordinates to the persons location

A normalized skeleton is taken as a basis in the first step. For this purpose, a skeleton of an average sized person, standing upright, was chosen. The pose is natural, having the arms relaxedly hanging besides and a upright straight torso and head. The skeleton is refined to be exactly numerical symmetrical with respect to the left and right body parts. It is sitting right in the origin of the coordinate system, facing towards the $z$-axis, which can be regarded as north in the local NED coordinate system. This defines the zero-state for the kinematic chains, the orientation and the location. It would also be possible to feed this initial state with the static information of stature and adjust bone lengths on a per person basis.

Given this basic skeleton, all features that influence the pose, are applied. In this step it is important to respect the order of the kinematic chains: Each rotation of a revolute joint has to move not only the adjust bone, but the whole structure that is connected. A rotation of the shoulder joint, for example, has to turn the whole arm with its three subsequent bones and joints. The rule is to always turn those sub-trees that are further away from the root joint. After having applied all known features, the skeleton is estimated as best as possible from the information that was available. The base skeleton from the first step ensures that even joints that were left untouched sit in a reasonable position. All joints can now be transformed back into vectors in a cartesian coordinate system for further processing.

Heading information can now be utilized to rotate the intermediate skeleton to the estimated orientation. Therefore, not solely the phone heading information is taken. Instead it is enriched by the heading delta, the result from the machine learning algorithm. Those two features are combined to result in the estimated direction, the person is facing. As

the skeleton is still located at the origin of the coordinate system, it can be turned using a rotation matrix that is applied to all joint vectors.

As a last step, the location can be applied by translating all vectors with a position offset. As already mentioned, this will not be discussed further as it is not part of this work. In the $y$-axis that is the up- and downward direction, however, the skeleton is translated in such a way, that the bottommost joint is touching the ground plane. The assumption therefore is, that the person is standing on the ground at any time. This way it can not be accounted for jumping, but these are of very short duration anyhow.

As a result, the skeleton is reconstructed as far as possible using only a sensor data from the mobile device and a trained linear regression model as input data. The stream of such skeletons can now be visualized or even used like the Kinect skeleton stream to build further applications on top of it.

# 6. Wearing position

To recapitulate, the work started with the assumption, that the mobile device is carried inside a person's trouser pocket. The method for reasoning about body features was then outlined, beginning with the sensors, leading to the extraction of input features, up to the point of fitting a linear regression model. Until now, the method is completely described, but only valid for the case of having a single fixed and above all *known* position of the device inside the pocket. As one would assume, people have different preferences and mobile devices are worn in very different manners, which is elaborated in detail in the following.

## 6.1. Wearing preferences

To study the preferences of people wearing their devices in the trouser pocket, a survey was carried out. Persons at the university campus, as well as volunteers via an online survey form, were asked which pocket they use for carrying their phones and how exactly the device is positioned in there. Another important question to be answered was, if they alter these positions from time to time or if one specific orientation is chosen predominantly over the others. Of all asked people, 65 of them had a smartphone which they also confirmed to carry in their pockets regularly. The results are shown in table 6.1. As no significant correlation between display orientation and standing direction could be deduced from the data, the table compares each of those independently with the pocket side.

Interestingly, no single person responded to change the pocket side. Several interviewees even gave reasoning for this unaskedly: while they have other objects in one pocket, like keys or similar, they only have a single free pocket to put the phone into. Another noteworthy outcome of this poll was, that not a single respondent stated to wear his cellphone

| | **Display orientation** | | | | | | **Standing direction** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Side** | **inwards** | | **outwards** | | **varying** | | **upright** | | **turned** | | **varying** | |
| Right pocket | 22% | (14) | 22% | (14) | 5% | (3) | 29% | (19) | 9% | (6) | 9% | (6) |
| Left pocket | 32% | (21) | 11% | (7) | 8% | (5) | 23% | (15) | 12% | (8) | 15% | (10) |
| Varying | 0% | (0) | 0% | (0) | 2% | (1) | 0% | (0) | 0% | (0) | 2% | (1) |
| Total | 54% | (35) | 32% | (21) | 14% | (9) | 52% | (34) | 22% | (14) | 26% | (17) |

Table 6.1.: The results of the poll about wearing preferences. Back pockets are omitted, due to no interviewed person using them.

in the back pocket of the trousers. In the 2005 study of locations of wearable or handheld devices in public spaces, still 6% of the interviewed users named to utilise their back pockets [ICG05]. The difference here is, that in the aforementioned survey not only smartphones were taken into consideration, but all kinds of handheld devices or cell phones. With smartphones, which feature a large display, the risk of breaking it is real, which may be the reason why people nowadays avoid putting it there.

Positioning the device's display to either face towards the body or the other way around was decided deliberately by the majority of users. It is less likely that a user changes this orientation frequently. Facing the display inside was justified by several people with being afraid of accidentally breaking the glass cover of the display, for example by hitting against the edge of a table. One person mentioned that this had already happened to him, which caused him to switch it around and always wear it with the display facing inwards since that time. Even so, it does not seem to be the case that one possibility is significantly favoured over the other in general.

A similar pattern is apparent with the up- and downwards orientation of the device. While there is a non-negligible fraction that does not pay attention on how they position their phone, the larger part of people rather selected an orientation which is maintained. This could be explained by habits that people became accustomed to when putting the device into the pocket, as confirmed by several individuals.

In conclusion, while not based on a comparatively large data set in this survey, it appears sufficient to allow the deduction of the following two key statements:

- **Regarding all interviewed persons:** the overwhelming majority uses their front pockets, but no specific side or orientation emerges over the others.
- **On a per person basis:** the side of the pocket, where the phone is carried, appears to be constant for nearly all users. Moreover, the orientation inside that particular pocket is more likely to stay fixed as well, rather than to vary from time to time.

These results are very important, because they lead to the following conclusions: if the preferences for a specific person are known a quite reliable basis is established to build upon. This preference can be gained from the user by either by having the user set them or by inferring them from available sensors. Further enhancement can be achieved by regularly validating those preferences with the current state in such a way that the model can be switched if a change in the wearing position was noticed. Candidates of such automatic classification methods are discussed next.

## 6.2. Classification

As seen, the side of the used pocket most probably does not change for a person. The user needs to set his preferred side once and is done. What is left is to determine the orientation of the display as well as the up and down direction. These are two independent problems, which have two possible values each. The problem therefore reduces to two binary classifiers. To find suitable input values for such classifiers, the following approaches have been

evaluated:

- Using both the primary and the font-facing cameras of the phone, try to infer the orientation from different average luminance values of both video streams.
- Using the speaker and the microphone of the device, try to infer the orientation based on reflection and damping effects of a short reference sound.
- Using the attitude information, based on the inertial sensors, repeatedly try to exclude impossible or at least improbable states until only a single valid state is left.

All three of those approaches were evaluated and are discussed in the following, starting with the luminance classification method.

### 6.2.1. Camera luminance

The basic idea behind this method is to leverage the fact that many trousers are made of fabrics that are light-transmissive to a certain degree. Even thick jeans cloths are not completely opaque. This can be exploited to measure the difference between the outwards facing side, looking through the cloth and the inwards oriented side of the device that is facing towards the body, where it is therefore darker. Many of the current phone devices have two built-in cameras, one primary at the back cover and a front-facing one that can be used for video calls, for instance.

To test if this idea could lead to a sufficiently accurate classification method, an application was implemented which activates both cameras and reads their preview video stream. The raw streams were used, where a single frame is encoded in the $YC_BC_R$ colour space, with the Y component containing the brightness information, $C_B$ representing the blue-difference and $C_R$ the red-difference chroma values [ITU94]. The luminance of a pixel could be directly read from the Y component. The average over every luma value of a frame was then used to determine the overall luminance of the picture.

Several measurements were made with the mobile phone in the pocket. Trousers made of different types of cloth, were used to determine the influence on the results. Each experiment was repeated several times, changing the lighting conditions with testing it indoors with artificial light and outdoors with daylight. Each combination of those parameters was measured for both cases of the display orientation. The results were clear and without ambiguity: this method is not suitable for any sort of binary classification. This is rooted in several reasons, starting with those that are intrinsic to the very basic idea:

- Different lighting conditions have very different luminance thresholds, i.e. the point off of which it can be classified as inwards or outwards facing.
- Different cloths are leading to different thresholds as well.
- At a very dark environment, the method has no chance at all to reason anything about the orientation.

While these restrictions imply enough issues already, there are even more complications. Both cameras are different in lens sizes and sensor surface area. This is explained as the

primary camera should be able to take higher quality photos, while the front-facing one is intended for lower quality video calls. This leads to different behaviour of both video streams: targeted at the same reference scene, both cameras produce different average luminance values. While this fact alone could be compensated by a calibration factor, it emerged that the luminance resolution is higher with the primary camera. Also, the time to initialize the stream and finish the focusing and lighting adjustments behaves different with both cameras. Even worse, cameras and all those discussed properties also vary with different phone models, making them even less comparable and the method therefore harder to generalise.

Ultimately, one final reason renders the method useless: the cameras have an adaptive mechanism to adjust the luminance amplification on the amount of light that is exposed to it. This amplification is the CMOS counterpart to the iris of a classical camera, which is also adapted to the light quantity, like described in [Nis96]. While this mechanism is desirable for taking photos or recording videos, it is counterproductive in this context. When the amount of light increases, this adaptive technique reduces the amplification slightly and is therefore darkening the image, while lightening up the image when it gets darker. This effectively narrows down the luminance resolution further. This behaviour can not be turned off, as it is either implemented in a core driver component or even integrated into hardware components.

### 6.2.2. Acoustic patterns

Another possible approach for classification of the devices orientation inside the pocket could be the exploitation of acoustic effects. The idea behind this is that microphone and speaker of the phone are placed in different spots when the orientation is changed. In addition, the pocket is vertically asymmetric, as it is open to the upside. When a particular sound is emitted by the speaker, it will be reflected and damped differently depending on the orientation, which in turn could be measured by capturing the sound again, using of the microphone.

To validate the suitability of this method, another application was created that plays a short 100 ms long beep sound, while starting a 300 ms long recording with the microphone, about 100 ms before the sound starts. As a reference sound, a triangle wave was used, with a base frequency of about 1,7 kHz and 4 odd harmonics to form the triangle shape. This way, the effect could be analysed for multiple frequencies. Again, tests were made with varying the orientation inside the pocket while playing a series of these sounds and recording them.

The experiments show that microphones of current mobile phones are just not accurate enough to capture the sound in the quality needed to allow for analyses of this kind. After all, those microphones are only optimized for capturing speech and being as compact as possible as well as being inexpensive. The frequency spectra of the recordings are strongly deformed, even when holding the device freely in mid-air for reference. Figure 6.1 shows the spectra of the reference sound and the recording, with the device lying on the ground.

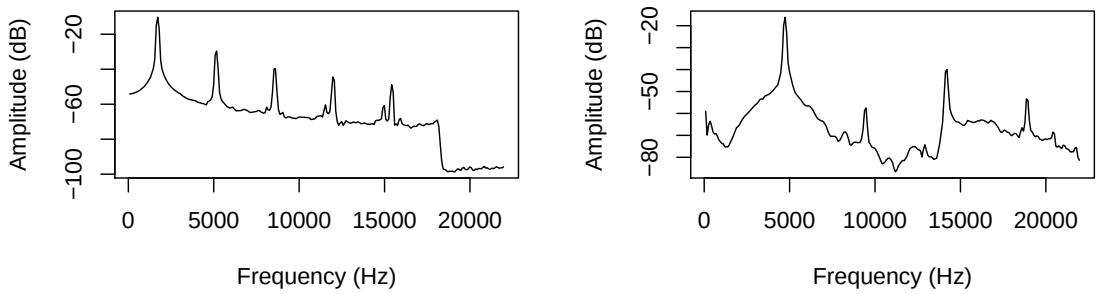



Figure 6.1.: Frequency spectra of the original sound and its recording.

To study the similarity of different spectra, pairs of all possible 2-combinations were built from the set of test recordings. For each pair, the sum of mean squared differences of the discrete spectral components was calculated. The spectra turned out to often times be more similar between two different orientations, as they were between two measurements in the same orientation. This can be seen in the detailed comparison listing in appendix 8. With such absence of any correlation between the orientation and possible patterns in the spectra, no meaningful classification can be found. Apart from that, even if such a classification was found, the method also had the following intrinsic drawbacks:

- Loud environment noises disturb the sound, making it impossible to apply this approach under such conditions.
- If the device is set to silent mode, the operating system will refuse to play the sound, which also renders this method useless in these cases.
- Using different trousers with wider or tighter pockets as well as thinner or thicker fabrics, the recorded sound has different spectra as well. Trying to account for all these diversities makes it even harder to find a common denominator for each orientation class.

### 6.2.3. Attitude heuristics

After having seen that other sensors of the mobile device are not well suited to reason about the phones orientation inside the pocket, the set of inertial sensors was left to be examined. The idea is to find heuristics about phone attitudes that are unlikely to happen, if the phone is being carried in certain positions. The primary disadvantage with this approach is that there is no guarantee to detect the right pocket orientation state immediately. The reason for this is, that different states lie in the same equivalence class of attitude measurements for certain body postures.

This means, that if for example the person was standing, there is no way to determine

whether the phones display is faced inwards or outwards just from the attitude of the phone. In the first case, if it was oriented away from the body, it looks approximately in the same direction as the user. In the second case, with the display faced inwards and the user standing in the opposite direction, the phones attitude is very similar. This makes it improbable to find a good classifier to separate those two cases. When sitting, however, one of those possibilities can be excluded, as it is impossible to have the phone in the front pocket facing outwards, while at the same time the sensor attitude describing the display is facing towards the ground plane. The same is true for the other pocket orientation parameter: with the person standing, it can be determined whether the phone is placed upright or upside down, as one of both would be only possible if the person was doing a handstand, which is improbable, or the phone was currently not inside the pocket. Using these domain specific contradictions, it leads to the heuristic approach described in the following.

For this method, the attitude of the phone is constantly observed. Recalling the characteristic vectors of the heading calculation in section 5.2.1, the projected vectors $\vec{v}'$ and $\vec{w}'$ are now utilised to detect those improbable positions. If one of those is nearing zero length, the direction the original vector $\vec{v}$ or $\vec{w}$ is given penalty points. The penalty points are weighted with how close the projected vector is to zero, i.e. the inverse of the interval $[0, 1]$ the projected vector length lies in. All those points are summed up, i.e. integrated over time. The direction with the least of such penalty points is then taken as the more probable one. $\vec{w}$ is thus used for assessing the state of the phone being carried upright or upside down, $\vec{v}$ is used to determine the display facing direction, respectively.

One problem remains, however: the method is very slow to react to position changes, if it was running for a relatively long time already. In this case, the carrying position has been well established and the sum of penalty points grew large. If the user is now taking the phone out of the pocket and after a short while puts it another way back in, it would take a very long time for the method to actually react to this change. The sum of penalty points of the new orientation has to catch up and overtake the established one first. To account for this, older penalty points are devalued when penalty points for the opposing side are generated.

Finding the right parameters for weighting the penalty points as well as the exact function for devaluation of opposing points are not easy to find and depend on the users activity properties. Such properties include the likelihood of the user changing positions regularly as well as the typical duration of carrying the phone in relation to the time it is outside the pocket. Nonetheless, this method proved to be the most reliable of all examined approaches in this work to determine the orientation of the device inside the users pocket. It takes an initialisation phase, but after this has finished, it finally stabilises and provides sufficiently correct results for practical purposes.

As this orientation state is likely to stay the same per user (see survey results above), it is fair to assume that on a subsequent start of the mobile application the orientation will not have changed. Therefore it is stored and taken as the initial state for future executions of this method. This does not mean that the orientation could not be changed then. The method is just initialised with a reasonable guess of the most probable state for the user, but moves on like previously described.

While this method is probably not the most advanced one, the idea could be improved with applying techniques of reinforcement learning, which will not be discussed in this work anymore, however. Furthermore, other approaches were shown in related work, which could also be used in conjunction with this. Kunze et al. [KLPB09], for example, have shown that the pocket wearing position can be deduced with the help of both accelerometer and location data when the person is walking.

## 6.3. Model selection

After having seen different approaches to classify those different wearing positions of the device, some of which are not very promising and one that is better suited, the results somehow have to be incorporated into the overall method. There are basically two methods to use this information.

As there are two possible disjunct states for each the pocket side, display orientation and standing direction, there are $8$ distinct states overall. The first possibility would thus be to train eight models for each state respectively. Each time the state is determined, it would be switched to the corresponding regression model. This approach has the drawback of implying the need for training of each of those different models. The quality and accuracy of each such model could be varying, which makes it complex to evaluate the overall method. On the other hand, those independent models could account for small peculiarities between the different states. The possible benefit of these detailed effects would then be improvements of the overall accuracy.

A second solution would be to train and use just one single regression model that is valid for all states. Therefore, a default state is defined, e.g. the phone being located in the right pocket, standing upright, with the display facing outwards. Any other state is mapped onto this default one by mirroring the sides or rotating the phone by $180°$ around the respective axes, if required.

Measurements have shown, that different models for each state are strongly similar, except for all critical dimensions being shifted or inverted. Models for the right and left side, for example, are nearly identical and yield valid results, when the left pocket model is supplied with data of right pocket measurements that were mirrored and vice versa. The same could be discovered with the other two state parameters. The models were interchangeable, when applying the corresponding rotation to the phone attitude. This can be explained by the symmetry of the contour of current smartphone devices: when turned around in the pocket, the devices attitude behaves identically for all body postures, except for being rotated by exactly a half turn around a known axis.

It was therefore decided for the second approach, using only a single model. This also has the advantage of concentrating all training data sets on this model. This, in turn, assures that there are enough example values to get a well fitted model from the machine learning process. The same model can also generalise better over different persons then.

# 7. Evaluation

Several intermediate evaluation results have been shown in the previous chapter, as they were necessary to develop the method in the first place. Therefore, the results of the overall method are left to be shown in the following. First, the implementation that was developed according to the presented approach is described shortly. Next, the statistical significance of the features for the various response variables is discussed, before presenting the quality of the overall results.

## 7.1. Implementation

The software implementation of the presented approach, called 'BodyOrientation', was written with a clean architecture, dividing the computing algorithms from the visualizing user interface modules. It is more than a mere prototype or proof-of-concept. The code, or parts of it, can be used for further research projects, as it is modularly structured and well commented. It was developed in the c# programming language and is using the .NET framework as a runtime environment. The source code was released and is available on the GitHub platform [Dau12a].

The software is able to receive mobile phone sensor values over a TCP connection with the use of a specially developed protocol and has an interface to the Kinect SDK, which enables it to gather skeleton data from a Kinect depth sensor. A recorder functionality was implemented that allows for storing the incoming data with the use of an own serialisation method. This played an important role in the development of the method, as it allowed to collect raw test data which could be saved and replayed over and over again to test and compare different refinement approaches for the method. The file format was designed to allow both real-time replay with the possibility to rewind and jump forward as well as the immediate processing. In the latter case, visualisations are disabled and the recording is not slowed down to match the speed with which it was recorded. This allows for faster processing of the recording into a stream of features for further evaluation, which can then happen as fast as the computational power allows.

Exporting functions to the *Weka* data mining software as well as to the *R* statistical language framework allowed for early tests. An interface to the *R* framework was implemented later on, to enable the application to send the preprocessed training data set to the model fitting algorithm and directly estimate response features with the fitted coefficients in the program.

User interface components were developed specially for this application to allow the visualisation of all raw data and processed features in diagrams. Furthermore, there are view-

ports where the mobile device's current attitude is rendered as well as both the measured and the estimated skeletons are shown. A screenshot of the application's user interface is shown in appendix 8.

It is accompanied by a mobile application, also written in c# and running on the Windows Phone mobile operating system. Its purpose is to collect all sensor values and the sensor fusion results and send them over an established network connection to the PC application. It was published to the Windows Phone Marketplace and can be downloaded for free [Dau12b]. Appendix 8 shows two screenshots of the application.

Such programs can also be written for any other mobile operating system. For this purpose, the protocol was held simple: a set of predefined sensor values has to be retrieved from the system and sent over a TCP socket, which is established with the PC by connecting to the correct port and receiving a constant magic number as a confirmation. After the connection is open, a packet has to be sent about every 50 ms, which holds the sensor values. Such a packet is a byte array containing the number of values as a 4 byte integer, followed by the sensor values as 4 byte floating point numbers, each in network byte order (big-endian). A description of the expected sensor values and their order as well as code examples can be found on [Dau12c].

## 7.2. Model analysis

The presented method could be evaluated with the help of this implementation. Having collected a training set, which was assembled from several recordings of 6 different test persons under changing circumstances, a basis for verifying different models was laid out. An analysis of different models shall be given in the following, where metrics for determining the quality of models are discussed as well as methods for selecting a suitable hypothesis space are shown. The reasons for the selection of the final models are shown, before discussing techniques for validation and their results. It all starts with having a look at the assumptions about linear regression models, however, which must be fulfilled first of all.

### 7.2.1. Residual metrics

In a fitted model, suitable values were already chosen for the coefficients $w_i$. As described previously, the $j$th item $\left(y_j, (x_{j,1} \cdots x_{j,i})^\top\right)$ of the training data set and its corresponding residual $\varepsilon_j$ have the following relation:

$$y_j = \sum_{i=1}^{n} w_i x_{j,i} + \varepsilon_j$$

where $y_j$ is the response value, also called fitted value in the following, and $x_{j,i}$ are the $i$ regressors, i.e. the mobile device features. The residuals are therefore given by the difference between the measured $y_j$ and the estimated response value $\hat{y}_j$:

$$\varepsilon_j = y_j - \sum_{i=1}^{n} w_i x_{j,i} = y_j - \hat{y}_j$$

The following four assumptions on the residuals have to be verified to assure that the hypothesis space for the linear regression was chosen correctly [Kle07]:

(I) The residuals **are independent.** If the still residuals contain a systematic trend, it would indicate that the model was not adequate.

(II) All residuals are **normally distributed.** This assumption was already made when the $L_2$ loss function was chosen to fit the model.

(III) All residuals have the **same variance.** The set of residuals is also called *homoscedastic* if this assumption is true.

(IV) All residuals have **zero mean.** This is equivalent to the statement that no outlier in the training data may significantly influence the fitted coefficients.

The first assumption is equal to the requirement of the response variables being independent. This in turn is subject to the design of the collection process of the training data. This assumption can be seen as confirmed in the context of this work, as the measurements were and will not be biased on the basis of previous measurements.

Furthermore, note that there is an important difference in the following two error types: while the *statistical error* of a measurement is the difference between the measured value from a true and unobservable value, the *residual* of the sample is its deviation from the estimated response of the fitted model. Regardless of the variances of the statistical errors, which may very well be equal over several measurements, as for example the properties of the presented inertial sensors showed, the variances of the residuals may differ for different items in the training data set.

Therefore, the residuals are adjusted by estimating their variances and normalising them. It can be shown [Lar08] that the these so called *standardised residuals* $s_j$ are then given by multiplying the corresponding residuals $\varepsilon_j$ with the following factor:

$$s_j = \varepsilon_j \frac{1}{\sqrt{1 - h_{jj}}}$$

where the *leverages* $h_{jj}$ are the diagonal elements of the so called hat matrix $H$. It is called that way because it maps the observed values to the fitted value, i.e. $\hat{y}_j = H y_j$ which 'puts a hat on $y_j$'. It is given by:

$$\begin{aligned} H &= \begin{pmatrix} h_{00} & h_{01} & \cdots & h_{0N} \\ h_{10} & h_{11} & \cdots & h_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N0} & h_{N1} & \cdots & h_{NN} \end{pmatrix} \\ &= X \left( X^\top \Sigma^{-1} X \right)^{-1} X^\top \Sigma^{-1} \end{aligned}$$

with $\Sigma$ being the covariance matrix of errors, which can be the identity matrix in case of uncorrelated errors. $X$ denotes the so called design matrix, which consists of all regressor values. Recalling $x_{j,0} := 1 \ \ \forall j \in \{1, 2, ..., N\}$ it is given by:

$$X = \begin{pmatrix} 1 & x_{0,1} & x_{0,2} & \cdots & x_{0,n} \\ 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & \cdots & x_{N,n} \end{pmatrix}$$

Given these metrics, the concrete models can be verified in the following.

### 7.2.2. Residual analysis

The final model for the heading delta, i.e. the difference between the shoulder orientation and the phone heading as described in section 5.4.2, will be taken as an exemplary model to analyse in the following. Which features are involved in this model and why they were selected will be shown in a later section.

Assumption (I) has already been discussed above. To verify assumptions (II) to (IV), several special scatterplots are typically used, which are shown for this model in figure 7.1. From the large training dataset that was collected for this work, 3500 items were randomly selected for these plots. The shown fitted values represent the response values, which are the heading delta values.

In the top left corner of the figure, the residuals are plotted against their fitted values. If assumptions (III) and (IV) are satisfied, the points should be evenly distributed around zero, with the spread being about the same throughout the plot. The red line should ideally lie straight on the dotted line, which it approximately does. It can be seen that there are two 'clusters' to the right and left, which result from the samples where the persons were sitting or standing respectively, as the heading delta values largely differ between these two states. Values in the middle of these two clusters are caused by the transitions between these two states. They are more sparse because there are simply more measured samples from persons staying in those states rather than transitioning between them, which has a much shorter duration. However, there is no global upward or downward trend visible, as well as no other kind of systematic trend, such as a curvilinear bend of the whole point cloud.

To examine the distribution of the residuals, which should be normal according to assumption (II), a normal Q-Q plot is used. In statistics, this is a graphical method for comparing probability distributions and examine them with respect to equality. Therefore, the quantiles of a theoretical ideal normal distribution is plotted against the quantiles of the set of residuals. The more similar these two distributions are, the nearer the points will lie on the identity line. In Q-Q plot in figure 7.1 shows a very linear relationship in the middle of the graph which is slightly skewed to both ends. This is because the outliers which are more sparsely available are not perfectly describing a normal distribution. Overall however, the

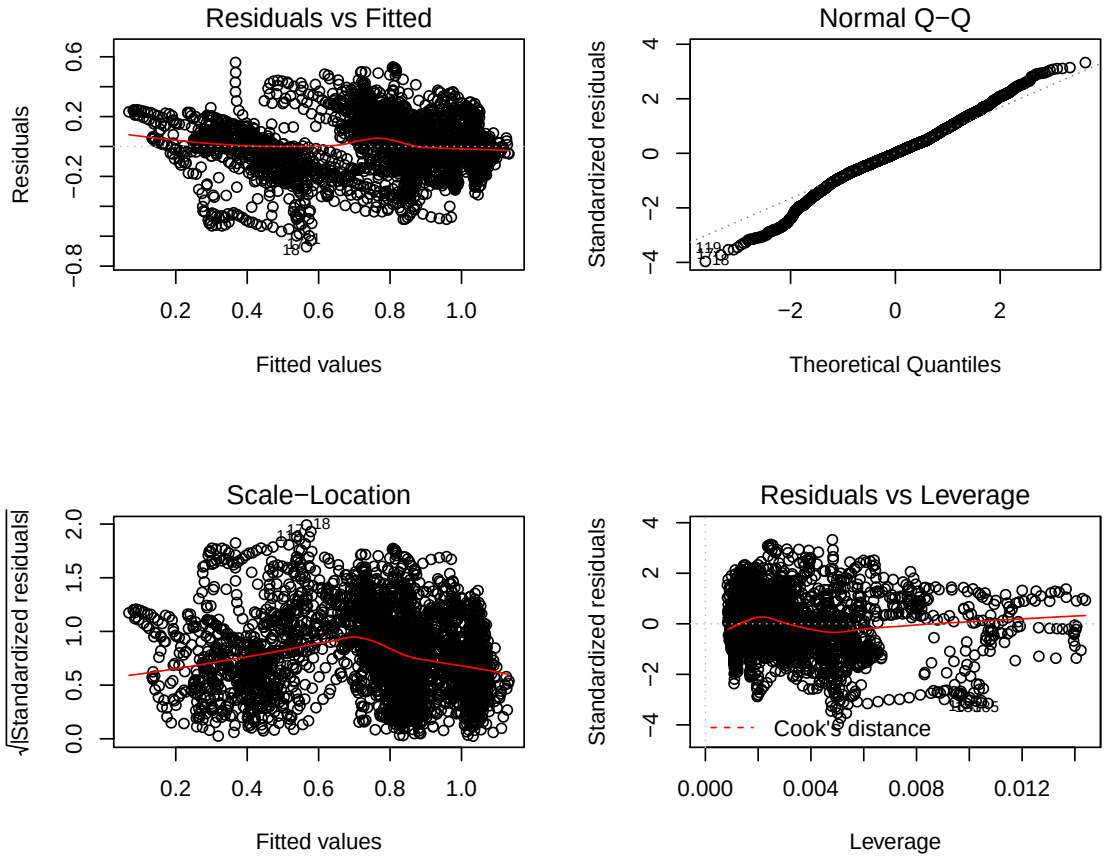



Figure 7.1.: Scatterplots for residual analysis of the heading delta fitted model. Fitted values and non-standardised residuals are given in radians.

deviation from the line is not significant and the normality assumption (II) can well be satisfied.

The third plot in the bottom left corner shows the square roots of the standardised residuals' absolute values against their corresponding fitted values. This is used for verifying assumption (III) in detail, i.e. the homoscedasticity of the residuals. If a systematic trend becomes apparent in the red average line, this assumption may be violated, e.g. if the line is steadily increasing over the whole length. It can be seen that there is a bend in the middle of the line, but a significant trend can not be noticed as the variance is rather homogeneous between values of $0.6$ and $0.9$.

As the four assumptions have been satisfied for this model, one last property can be examined: the existence and influence of outliers on the model, i.e. single points that are very far away from others. They can be studied with the help of the fourth and last plot in figure 7.1, which shows the leverage values $h_{jj}$ against the standardised residuals. A metric that is useful in this context is *Cook's distance* $D_j$ [Coo77] of each data point, which is given by

$$D_j = \left( \frac{\varepsilon_j^2}{p \sum_{k=0}^{N} \varepsilon_k^2} \right) \left( \frac{h_{jj}}{(1 - h_{jj})^2} \right)$$

where $p$ is the number of parameters or dimensions in the model. It is argued in the literature that a distance of $D_j > 1$ can be taken as a cut-off value, sometimes already $D_j > \frac{1}{2}$ for determining outliers. Both those lines would be shown in the plot, if they were not to far outside. This means that no point comes even near those distances, which in turn indicates that the model is not influenced negatively by such values.

Note, that these evaluations were also made with the models of the other body features, which are discussed later, with a very similar outcome. After having verified these models, their origin shall be outlined in the following.

### 7.2.3. Feature selection

First, the a very simple hypothesis space, in which the heading delta is described by a linear combination of only the current three attitude Euler angles features is regarded:

$$\Delta h \sim \phi + \theta + \psi$$

To get a glimpse on how good this describes the real relationship and how accurately future values will be predicted, the *coefficient of determination* [ST60], also called *R-squared* or $R^2$, can be used. It ranges from 0 to 1, and can be seen as the percentage with which the fitted model matches the data better than a simple average would. It is therefore defined with the help of a quotient of the sum of squared residuals and the sum of squared differences between the response values and their average:

$$R^2 = 1 - \frac{\sum_{j=0}^{N} \varepsilon_j^2}{\sum_{j=0}^{N} (y_j - \bar{y})^2} \quad \text{with } \bar{y} = \frac{1}{N} \sum_{j=0}^{N} y_j$$

A useless model would always predict the average value of the responses $y_j$ in the training data, therefore the all residuals were $\varepsilon_j = y_j - \bar{y}$, the quotient would be 1, finally yielding a $R^2$ value of 0%. If all training data points lie very close to the fitted model, however, the residuals get very small until reaching a $R^2$ value of 100% if all residuals would be exactly 0.

A fitted model in the aforementioned model of solely attitude angles has an $R^2$ value of 45% for the given total training data set. To compare this with more complex models, including more dimensions, the $R^2$ is not adequate. For this purpose, an adjusted version was presented in [The61] which accounts for the number of dimensions $n$ in the model and their relation to the number of samples $N$:

$$\bar{R}^2 = 1 - (1 - R^2) \frac{N - 1}{N - n - 1}$$

While $R^2$ can increase with the number of dimensions, even if no actual improvement was made in the model, the adjusted $R^2$ only grows if a new dimension improves the model more than would be expected by chance.

Due to the high number of samples and low number of dimensions that was used for the fitting of the simple attitude angle model, the adjusted $R^2$ is nearly the same as the non-adjusted version: also about 45%. To test if any relations between the angles explain the response more than the angles alone, the following model was fitted:

$$\Delta h \sim (\phi + \theta + \psi)^3$$

It uses all combinations of the angles to reason about the response value. The adjusted $R^2$ value indeed increased to a value of 52% in this case. This still does not contain any information about the past time window, which could hold valuable information. For this reason, the statistical features were presented in section 5.3.2. Using the standard deviations of the angles and the correlation coefficients between them over a certain time window to extend the hypothesis space yields the following ten-dimensional model space:

$$\Delta h \sim \phi + \theta + \psi \ + \ \sigma_\phi + \sigma_\theta + \sigma_\psi \ + \ \rho_{\phi,\theta} + \rho_{\phi,\psi} + \rho_{\theta,\psi}$$

These additions increased the $\bar{R}^2$ value to 66%, without even including any combination between the features. Only adding the standard deviations yielded 58%, only extending it with correlation coefficients still made a 52% adjusted $R^2$ value, i.e. in both cases it improved the model, both additions together improved it even more. This is a strong indication for those features to contribute positively to the explanation of the response. Note, that this model was also taken for the scatterplots in the previous section.

To analyse this further, t-tests can be made on each of the coefficients $w_i$. Therefore, the quotient of the fitted coefficient and its standard error $SE_{w_i}$ is used, i.e.

$$t = \frac{w_i}{SE_{w_i}}$$

will provide a t-statistic. The null hypothesis $H_0$ is in this case that the coefficient is zero and therefore the dimension only has a constant part which can also be added to the constant coefficient $w_0$, i.e the dimension does not contribute to the model. The results for these tests were in the above model with nine features, that for every feature, this hypothesis could be rejected with a confidence level of 99% or even more. This means these features all have a significant influence on the result and contribute to the quality of the model.

An additional test can be made on all coefficients at one. The idea behind this is to test if the model becomes significantly worse when removing certain dimensions. For this purpose, the residual sum of squares is calculated for the original ($RSS_1$) and the reduced model ($RSS_2$). The F-test is taken for this purpose, with the following F-statistic:

$$F = \frac{(RSS_2 - RSS_1)/q}{RSS_1/(N-n)}$$

where $q$ is the number of reduced parameters. For example, a model can be compared with the very simplest case of removing all dimensions except the constant coefficient. Again, for the currently regarded model with its nine features, the null hypothesis for this comparison could be rejected with a confidence of far over 99,99%. This means the model is a significant improvement compared to the constant one, which is not too surprising. With this test, however, two arbitrary models can be compared, if one uses a subset of features of the other.

### 7.2.4. Final models

After validating a lot of t-tests for single dimensions and F-tests for checking what happens if certain dimensions are omitted or added, the hypothesis spaces that proved to be the most suitable for the purpose of this work was selected. It was already anticipated: the heading delta is best represented by a model with all three attitude angles, standard deviations and correlation coefficients, i.e.

$$\Delta h \sim \phi + \theta + \psi \ + \ \sigma_\phi + \sigma_\theta + \sigma_\psi \ + \ \rho_{\phi,\theta} + \rho_{\phi,\psi} + \rho_{\theta,\psi}$$

No additional cross correlations were introduced between them, as the improvement was negligible and quite contrary the risk of overfitting became real.

It was concealed until now, however, that also several other features were tested, which showed to be insignificant and not improve the model, such as the periodicity features and the mean over the time window. Also, features were tested that were not specially presented in this work, like the linear acceleration itself as well as features which were deduced from it, which also proved to be unsuitable to explain the response variables.

Interestingly, the primary leg angle, i.e. the side where the mobile device is located, showed also to be best predicted by the very same features. Its hypothesis space is given by:

$$\alpha_{\text{leg}} \sim \phi + \theta + \psi \ + \ \sigma_\phi + \sigma_\theta + \sigma_\psi \ + \ \rho_{\phi,\theta} + \rho_{\phi,\psi} + \rho_{\theta,\psi}$$

The results for the opposite leg, however, show a comparatively large deviation from the normal distribution for the residuals. This became apparent in the normal Q-Q plot, where the values are bent away from the line in the lower and upper third. It also showed relatively large residuals. Those were clear indications that the model was not suitable and the given features could not sufficiently explain the response value.

Intuitively, this can be explained by imagining that the leg angles can be in two modes: when standing and sitting as well as transitioning between those states, both leg angles are nearly equal or at least rotating in the same direction. When walking, however, the

leg angles behave inversely, as in each step one angle is rotating in one direction while the other is rotating in the opposite way.

To model this, a reliable explanatory feature is needed for distinguishing those different movement patterns. As the results show, the periodicity feature presented in section 5.3.3 does not adequately capture this. Statistical tests showed that this features does not significantly contribute to the model. Maybe those more advanced periodicity detection mechanisms, mentioned in the same section (5.3.3), provide better suited regressor variables for this purpose. However, it was not possible to test this in this work anymore.

### 7.2.5. Cross-validation

The models for the heading delta and the primary leg angle were furthermore tested via *$k$-fold cross-validations* [RN10, p. 708]. This technique allows to estimate if enough training data was collected and if the model is likely to predict reasonable values for unknown future inputs. The test set is split into $k$ disjunct partitions for this purpose. Due to the independence of the samples, as discussed in section 5.1.3, these can be randomly chosen. These partitions serve a double duty then: in each of the $k$ rounds one is used to fit a model in the given hypothesis space, which is then tested against data from the respective other partitions.

To evaluate the results, the average $R^2$ value of all rounds is compared with the $R^2$ value of the model over the total training data set. If the test data is not sufficient, the average $R^2$ will be much lower than the total one, as the models perform poorer and are not able to predict data they were not trained on.

The results of a 10-fold cross-validations, which is a common value of $k$ for such purposes [RN10], were very clear: for the heading delta model, the $R^2$ average value was only less than 0,5% smaller than the total $R^2$. The model for the primary leg had even only 0,15% decrease. These are again clear indicators for the robustness of both models and are a sign that a sufficient amount of training data was taken into account.

## 7.3. Results

Having detailed the process of dimension selection for an adequate model and having tested those models against various assumptions, the results will now be presented.

### 7.3.1. Prediction precision

Aside from all these decisions specific to regression, other parameters had to be balanced as well to achieve a good quality of prediction. One of the most important ones in this context is the length of the sliding window. As presented, values in this window are taken to calculate time-dependent features of which two are especially relevant in the regression

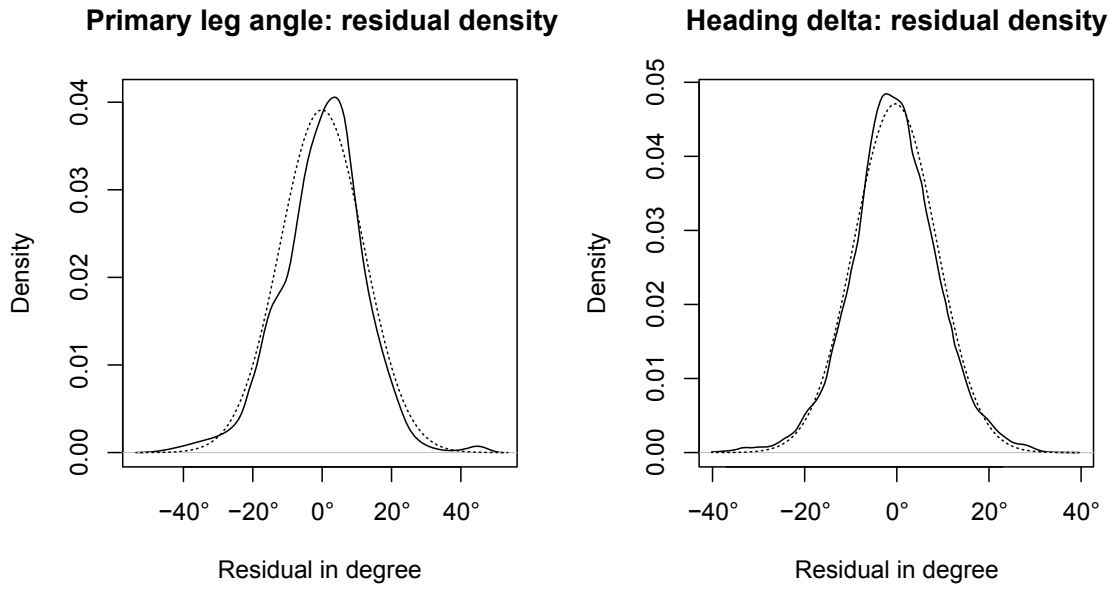



Figure 7.2.: The densities of the residuals are shown from the fitted models of both the primary leg angle and heading delta response variables. The courses of the ideal normal density graphs are shown with dotted lines.

models: the standard deviation and the correlation coefficient, each one needing to look back a certain number of time frames.

If this number is chosen too small, the calculated values lose their meaning, as the values of the sensors do not change drastically from one frame to another most of the time. A standard deviation over a very short period of time, which contains a set of nearly constant values will not carry much information, for instance. If the time window is chosen too large, the time-dependent features will no longer be responsive enough to changes, as they will be 'blurred' by values which lie too far in the past. After many different test, a time window of 32 samples has emerged to be the best choice in terms of precision of the results. A power of two was chosen as it simplified the FFT calculation for the periodicity features. With a frequency of about 31 frames per second (see section 5.1.2), this equals a time window of about one second.

The final precision of the predictions can be seen in the two density diagrams in figure 7.2 for the two discussed response variables. The evaluation of samples quantiles shows that with a probability of 50% the heading delta value differs no more than $5.5°$ from the true value and is in 75% of all cases still inside a range of $-9.8°$ to $10.7°$. Only in 5% of all cases, the model predicts an angle with a deviation of $\pm 20°$ or more.

For the leg angle, results are similarly promising, even though of slightly less precision. Half of all values are predicted with a deviation of about $\pm 7.8°$, 75% lie inside a $\pm 12.3°$ range, while angular differences of more than $26°$ are only predicted with a 5% chance.

These are the deviations from the predicted from the measured values, however. Those measured values from the Kinect are in turn again different from the true real-world values. As can be seen in [Kho11], the accuracy of the depth map is depending on the distance of the subject to the depth sensor. At a distance of the test persons of about $3.5$m, the standard deviation of depth measurements is still lower than $2$cm. With a typical shoulder joint distance of about $40$cm, this yields a standard deviation of the error for the measured shoulder angle as small as about $5.5°$ in the worst case. Both normally distributed errors sum up to an again normally distributed total error with the standard deviation also being the sum of the single ones, which is also shown in the next section.

### 7.3.2. Probability density of $\delta\theta$

As already mentioned in the first chapter of this work, the difference of shoulder heading angles of different persons $\delta\theta$ is an essential part for the method presented in [GLR+10], which uses this variable together with other parameters to reason about social interactions. As the method of this thesis yields just such single headings, which are absolutely oriented in a local reference frame, it can easily be extended to a system of two persons with mobile devices running this method, which both exchange their heading values to calculate the $\delta\theta$ value. Although such a system was not yet implemented, the expected precision shall be discussed in theory briefly.

Let $X$ and $Y$ be the random variables of the two errors, which are both normally distributed with a mean of $\mu_X = \mu_Y = 0$ and an equal standard deviation $\sigma_X = \sigma_Y$. The difference between those two variables will be defined as $Z = X - Y$. Due to the normal distribution with a zero mean, $Y = -Y$ follows, which can be seen by imagining the density graph, which is symmetric around $0$. Therefore, the difference is equivalent to the sum of both random variables, i.e. $Z = X + (-Y) = X + Y$. As $X$ and $Y$ are independent from each other, the resulting variable $Z$ is distributed normally again, with parameters $(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$. This can also be seen by imagining that the density function of $Z$ is the result of the convolution of the two original densities [HMC05], i.e.

$$f_Z = f_X * f_Y$$

Using the moment generating functions, like explained in [Ros07], yields:

$$\begin{aligned}
\phi_{X+Y}(t) &= \phi_X(t)\,\phi_Y(t) \\
&= \exp\left[\frac{\sigma_X^2 t^2}{2} + \mu_X t\right]\ \exp\left[\frac{\sigma_Y^2 t^2}{2} + \mu_Y t\right] \\
&= \exp\left[\frac{(\sigma_X^2 + \sigma_Y^2) t^2}{2} + (\mu_X + \mu_Y) t\right]
\end{aligned}$$

The resulting moment generating function uniquely describes the distribution and with the parameters of $X$ and $Y$, it follows: $Z \sim N(0, 2\sigma_X^2)$. Therefore, the standard deviation

of $Z$ is $\sqrt{2}$ times the one of $X$, which makes the total error of $\delta\theta$ only about 41% larger than one of the heading delta, shown above.

This means, $\delta\theta$ would result from two different predictions on two separate devices and could be precise to a range of $\pm 14^\circ$ with a 75% certainty. However, this requires also the chronological synchronisation to be accurate enough and can only be achieved if other disturbances are minimised as well.

### 7.3.3. Generalisation

The previously discussed results were based on a combined training set of different test subjects. Like already argued in the beginning of this work, in section 1.3.3, it is desirable to have such a generalised model that works well for different people. This has raised the question, however, how much accuracy is lost with such an approach in comparison to a model that is fitted individually for each user.

The results have answered this question pretty clearly: The standard deviation of the residuals of individually fitted models was decreased by about 5-10% at most. One test subject's model predicted even slightly worse values than the combined model in a later test. This was presumably caused by having had an insufficient amount of training data for the users own model. As often the case with machine learning techniques, every iteration of improvement is getting harder than the previous ones and yields less gain in the results.

In conclusion, the trade-off has to be made specifically with the application atop of this method in mind. It has to be carefully balanced whether or not this small increase in accuracy is worth the effort of training a specific model for each and every user. Furthermore, this gain can only be achieved by intensive training, again for each user. The training data set has to be carefully selected, only collecting data if the person is fully in sight of the depth sensor, and if the mobile device is actually inside the pocket. In many cases, this effort may not be in reasonable proportion to its comparatively small benefit.

It should be also noted that different trousers with differently cut pocket shapes or even other parameters can vary from time to time for the same user. These variations are fairly similar to those originating from varying the user in a generic model. With the inevitable presence of these uncertainties, the construction of a generalised model is even more favourable.

# Conclusion

In this thesis, a concept was proposed to use inertial sensors of smartphone devices to infer certain body geometry features of the user that is carrying it. The assumption was made, that the user carries his device in one of his trousers' pockets to make the scope of this work feasible. After having outlined the frame conditions that are given, namely the phone sensors and their properties, a method was presented that is able to infer those target values via a machine learning algorithm, after having been trained adequately. The collection of training data was achieved by using a depth image sensor and a skeleton estimation framework that provided with actual values of the user's body geometry, simultaneously to the measured phone sensor values.

The presented method uses a number of transformations before and after applying the actual learned model, which exploit domain knowledge to improve the overall precision of the result. A software implementation of the presented approach provided for data collection to asses different models and evaluate the overall results. Not only this, a complete modular framework was implemented that can be used to develop improved methods based on inertial sensors and machine learning techniques, as it offers all fundamental functionalities starting from recording and replaying sequential measurements over to signal processing and visualising intermediate and final results.

As conclusion about the specific results of the fitted body geometry features, it can be stated that especially two regarded features proved to be very reliably detectable: the users heading and the angle of the leg where the phone is located. Though other features about the body posture did not prove to yield promising results, the ability of determining these two is a huge achievement of this work. The final precision of both those values was better than the author expected. Furthermore, it is just those two features that are very valuable indeed for many applications.

It was already mentioned that this could be applied to social situation detection methods, where the relative heading of two persons is of special interest. As the yielded heading, like presented, is relative to a common local reference system, such a relative heading can easily be deduced.

As only one additional example, these two angles could also find application in pedestrian or indoor navigation, where the leg angles can be used to detect both the number of steps as well as their size, as a larger step must yield a larger deviation in the leg angle. At the same time, the users heading angle indicates the direction in which the user is walking, which is something that could not be accounted for with the phone heading alone.

Just like in this example, there are certainly many areas in which these body-related values are more useful than the raw phone-related ones. This might as well answer the question

that was left open in the first chapter: whether or not the applied domain knowledge allows to receive more expressive data than contained in the raw sensor readings alone. It is left to the reader to judge if this goal was achieved.

Transforming sensor values for feature extraction and applying them to the fitted regression model is expected, but not yet tested, to run in real-time even on smartphones themselves. This way, the software would act as a high-level sensor, introducing an abstracting layer above the sensor-fusion layer that is in turn using the raw sensor data. The direct output of body geometry parameters could then be used for enhanced functionalities in applications, as the phone is aware of the user's facing direction alongside other parameters about the users pose.

However, there is still room for improvements. Beginning with the detection of the wearing position, the periodicity extraction and not even ending at the very fundamental machine learning approach, there is a lot that can possibly be improved in future works. Maybe even other body geometry features can be estimated with further enhancements to the presented technique.

# Appendix

# A. Microphone recording correlations

| | I D #1 | I D #2 | I U #1 | I U #2 | I U #3 | I U #4 | L #1 | L #2 | O D #1 | O D #2 | O U #1 | O U #2 | O U #3 | O U #4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I D #1 | 100,0 % | | | | | | | | | | | | | |
| I D #3 | 89,8 % | 100,0 % | | | | | | | | | | | | |
| I U #1 | 71,1 % | 79,5 % | 100,0 % | | | | | | | | | | | |
| I U #2 | 54,3 % | 65,4 % | 85,0 % | 100,0 % | | | | | | | | | | |
| I U #3 | 82,7 % | 75,7 % | 79,2 % | 61,5 % | 100,0 % | | | | | | | | | |
| I U #4 | 89,7 % | 83,4 % | 76,0 % | 55,4 % | 90,1 % | 100,0 % | | | | | | | | |
| L #1 | 88,2 % | 85,0 % | 61,5 % | 45,9 % | 70,1 % | 77,7 % | 100,0 % | | | | | | | |
| L #2 | 87,5 % | 85,1 % | 60,4 % | 43,7 % | 69,0 % | 77,0 % | 99,7 % | 100,0 % | | | | | | |
| O D #1 | 85,6 % | 87,8 % | 87,8 % | 69,6 % | 82,0 % | 90,5 % | 80,0 % | 79,6 % | 100,0 % | | | | | |
| O D #2 | 96,5 % | 90,3 % | 76,4 % | 58,6 % | 82,0 % | 88,2 % | 83,7 % | 83,5 % | 85,4 % | 100,0 % | | | | |
| O U #1 | 62,4 % | 78,0 % | 90,3 % | 82,5 % | 67,6 % | 69,6 % | 57,1 % | 56,0 % | 87,6 % | 65,6 % | 100,0 % | | | |
| O U #2 | 60,2 % | 74,4 % | 94,7 % | 85,2 % | 70,3 % | 67,5 % | 47,8 % | 46,4 % | 82,8 % | 64,4 % | 95,9 % | 100,0 % | | |
| O U #3 | 84,9 % | 77,7 % | 35,4 % | 6,8 % | 60,3 % | 77,4 % | 83,2 % | 83,5 % | 66,4 % | 76,5 % | 30,9 % | 22,9 % | 100,0 % | |
| O U #4 | 83,0 % | 70,3 % | 33,1 % | 0,0 % | 63,2 % | 76,9 % | 76,6 % | 77,2 % | 64,0 % | 76,1 % | 26,1 % | 18,4 % | 93,6 % | 100,0 % |

Figure A.1.: A selected subset of measured sound spectra, with the device having been in various orientations. The table shows the similarity of each spectrum to all others. The similarity was determined by calculating the sum of squared differences of all frequency components of the respective spectra. The percentage values were deduced by linearly normalizing the results, such that 0% means the least of all similarities and 100% denoting identity. The abbreviations for the different states are as follows:

**L** — device lying on the ground
**I** — display facing inwards
**O** — display facing outwards
**U** — upright standing
**D** — upside down

# B. Implementation screenshots

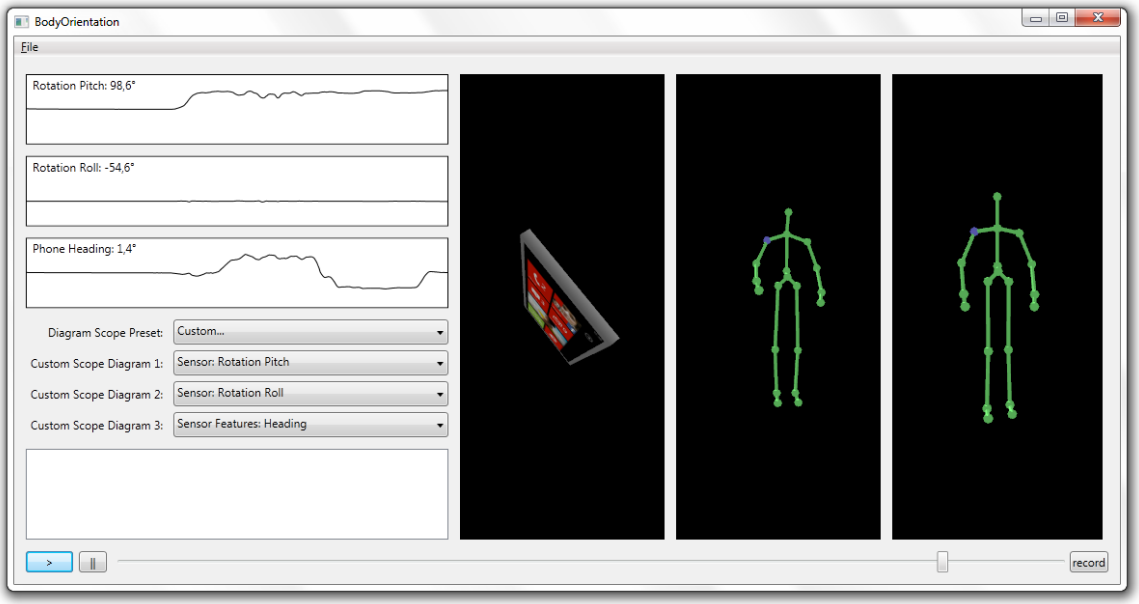



Figure B.1.: Screenshot of the application with the diagrams for selectable features and renderings of the device attitude, measured skeleton and estimated pose (from left to right) as well as the recorder controls to the bottom.

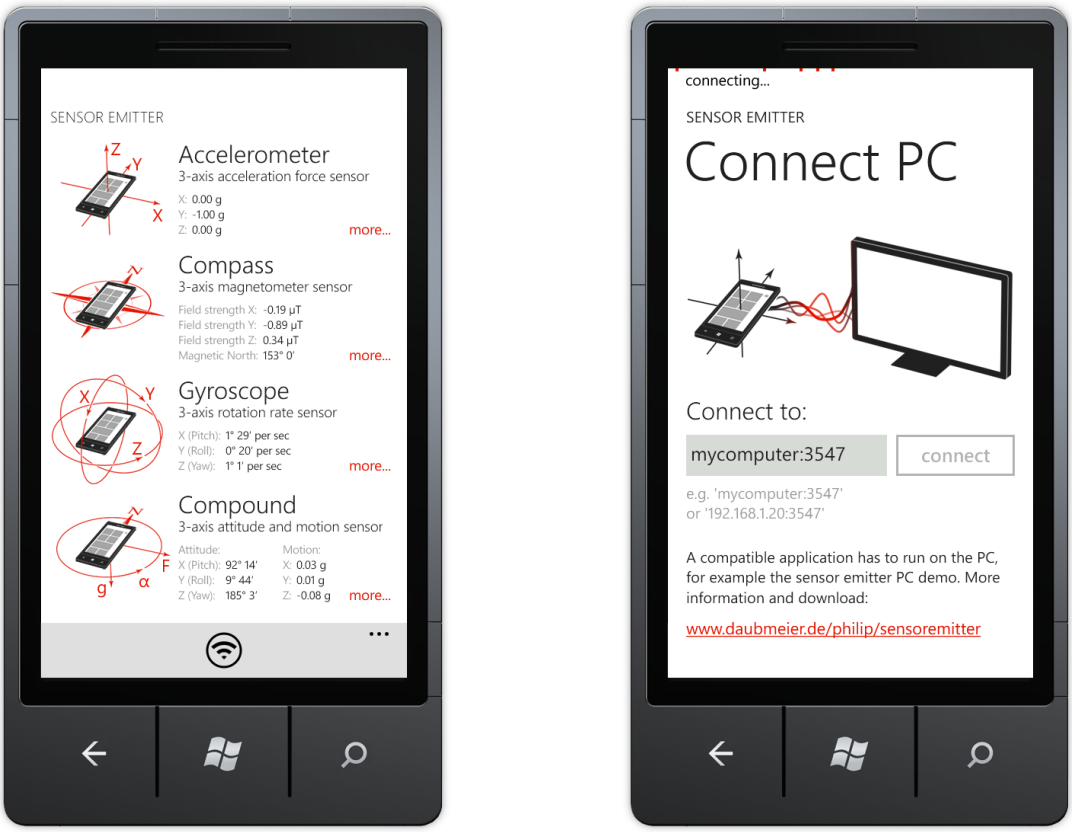



Figure B.2.: Screenshots of the mobile phone application. The left picture shows the start screen of the application, with real-time updated measurements of all sensors and the sensor fusion process. The user interface to connect to the PC software for transferring the sensor readings is shown to the right.

# Bibliography


[AAA07] R. Abdolvand, B.V. Amini, and F. Ayazi. Sub-Micro-Gravity In-Plane Accelerometers With Reduced Capacitive Gaps and Extra Seismic Mass. *Journal of Microelectromechanical Systems*, 16(5):1036–1043, October 2007. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4337776, doi:10.1109/JMEMS.2007.900879.

[ABMp+] Akin Avci, Stephan Bosch, Mihai Marin-perianu, Raluca Marin-perianu, and Paul Havinga. Activity Recognition Using Inertial Sensing for Healthcare , Wellbeing and Sports Applications : A Survey. *Methods*.

[App12] Apple Inc. CMMotionManager — 'startDeviceMotionUpdatesUsingReferenceFrame' Function, 2012. URL: http://developer.apple.com/library/ios/documentation/CoreMotion/Reference/CMMotionManager_Class/CMMotionManager_Class.pdf.

[Asa10] Asahi Kasei Microdevices. Technical Specifications: 3-axis Electronic Compass AK8975/B, 2010.

[BD10] Andrew Bookholt and Miroslav Djuric. iPhone 4 Gyroscope Teardown, 2010. URL: http://www.ifixit.com/teardown/t/3156.

[BI04] Ling Bao and Stephen S Intille. Activity Recognition from User-Annotated Acceleration Data. *Most*, pages 1–17, 2004.

[Bis06] C.M. Bishop. *Pattern Recognition And Machine Learning*. Springer, 2006.

[Bri88] E. Oran Brigham. *The fast Fourier transform and its applications*. Prentice Hall, Englewood Cliffs, N.J., 1988.

[CF86] G. A. Cavagn and P. Franzetti. The determinants of the step frequency in walking in humans. *The Journal of Physiology*, (373):235–242, 1986.

[CH06] S. Chatterjee and A.S. Hadi. *Regression Analysis by Example*. Wiley-Interscience, 2006.

[Col07] Shane Colton. The Balance Filter — A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform, 2007. URL: http://web.mit.edu/scolton/www/filter.pdf.

[Col11] R. Colin Johnson. MEMS gyroscopes become ubiquitous in smartphones. *MEMS Investor Journal*, 2011. URL: http://www.memsinvestorjournal.com/2010/05/mems-gyroscopes-become-ubiquitous-in-smartphones.html.

[Coo77] R. Dennis Cook. Detection of Influential Observations in Linear Regression. *Technometrics (American Statistical Association)*, 19(1):15–18, 1977. doi:10.2307/1268249.

[Dau12a] Philip Daubmeier. BodyOrientation — Source code, 2012. URL: https://github.com/philipdaubmeier/BodyOrientation.

[Dau12b] Philip Daubmeier. Sensor Emitter — Windows Phone Marketplace, 2012. URL: http://windowsphone.com/s?appId=08c94bea-924b-44b9-b4e3-03e571ea8ceb.

[Dau12c] Philip Daubmeier. Sensor Emitter web page, 2012. URL: http://daubmeier.de/philip/sensoremitter/.

[Die06] James Diebel. Representing Attitude : Euler Angles , Unit Quaternions , and Rotation Vectors. 2006.

[DKL98] Erik B Dam, Martin Koch, and Martin Lillholm. Quaternions , Interpolation and Animation. *Spring*, 1998. URL: http://www.itu.dk/people/erikdam/DOWNLOAD/98-5.pdf.

[DW10] St. J. Dixon-Warren. Motion sensing in the iPhone 4: MEMS accelerometer. 2010. URL: http://www.memsinvestorjournal.com/2010/12/motion-sensing-in-the-iphone-4-mems-accelerometer.html.

[DW11a] St. J. Dixon-Warren. Motion sensing in the iPhone 4: electronic compass. *MEMS Investor Journal*, 2011. URL: http://www.memsinvestorjournal.com/2011/02/motion-sensing-in-the-iphone-4-electronic-compass.html.

[DW11b] St. J. Dixon-Warren. Motion sensing in the iPhone 4: MEMS gyroscope. *MEMS Investor Journal*, 2011. URL: http://www.memsinvestorjournal.com/2011/01/motion-sensing-in-the-iphone-4-mems-gyroscope.html.

[Fac12] Facebook. Facebook Newsroom — Statistics, 2012. URL: http://newsroom.fb.com/content/default.aspx?NewsAreaId=22.

[Gar84] W. A. Gardner. Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique. *Signal Processing*, 6(2):113–133, 1984. doi:10.1016/0165-1684(84)90013-6.

[GD10] Georg Groh and Philip Daubmeier. State of the Art in Mobile Social Networking on the Web — Tech.-Report TUM-I1014. Technical report, Institut für Informatik, TU-München, 2010.

[GLR+10] Georg Groh, Alexander Lehmann, Jonas Reimers, Rene Friess, and Loren Schwarz. Detecting Social Situations from Interaction Geometry. *Proc. IEEE SocialCom 2010, Minneapolis USA*, 2010. doi:10.1109/SocialCom.2010.11.

[Goo11] Google Inc. Android API Guides — Motion Sensors, 2011. URL: http://developer.android.com/guide/topics/sensors/sensors_motion.html.

[GR11] Duncan Graham-Rowe. A New Direction for Digital Compasses. *Technology Review*, 2011.

[Hal79] E. H. Hall. On a New Action of the Magnet on Electric Currents. *American Journal of Mathematics*, 2(3):287, September 1879. doi:10.2307/2369245.

[HMC05] R. V. Hogg, J. W. McKean, and A. T. Craig. *Introduction to mathematical statistics*. Pearson Education, 2005.

[ICG05] F. Ichikawa, J. Chipchase, and R. Grignani. Where's the phone? A study of mobile phone location in public spaces. *IEE Mobility Conference 2005. The Second International Conference on Mobile Technology, Applications and Systems*, 2005:142–142, 2005. doi:10.1049/cp:20051557.

[IDC11] IDC — Press Release. Smartphones Outstrip Feature Phones for First Time in Western Europe as Android sees Strong Growth in 2Q11, 2011. URL: http://www.idc.com/getdoc.jsp?containerId=prUK23024911.

[Inv12] InvenSense Inc. MPU-9150 Nine Degrees of freedom inertial measurement unit, 2012.

[ITU94] ITU Radiocommunication Assembly. Recommendation ITU-R BT.601-4 — Encoding parameters of digital television for studios, 1994.

[JL03] Eric Jacobsen and Richard Lyons. The Sliding DFT. *IEEE Signal Processing Magazine*, (March), 2003.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983. doi:10.1126/science.220.4598.671.

[Kho11] K Khoshelham. Accuracy Analysis of Kinect depth data. *ISPRS Workshop Laser Scanning*, 2011.

[KJvdK98] B Kemp, a J Janssen, and B van der Kamp. Body position can be monitored in 3D using miniature accelerometers and earth-magnetic field sensors. *Electroencephalography and clinical neurophysiology*, 109(6):484–8, December 1998. URL: http://www.ncbi.nlm.nih.gov/pubmed/10030679.

[KL07] Kai Kunze and Paul Lukowicz. Using acceleration signatures from everyday activities for on-body device location. *2007 11th IEEE International Symposium on Wearable Computers*, pages 1–2, October 2007. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4373794, doi:10.1109/ISWC.2007.4373794.

[Kle07] D. G. Kleinbaum. *Applied Regression Analysis and Other Multivariable Methods*. Brooks/Cole, 2007.

[KLPB09] Kai Kunze, Paul Lukowicz, Kurt Partridge, and Bo Begole. Which Way Am I Facing: Inferring Horizontal Device Orientation from an Accelerometer Signal. *2009 International Symposium on Wearable Computers*, pages 149–150, September

2009. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5254664, doi:10.1109/ISWC.2009.33.

[Lar08] Pia Veldt Larsen. Regression and analysis of variance — Residual analysis, 2008.

[Lee12] Peter M. Lee. Theory of the correlation coefficient. In *Bayesian Statistics: An Introduction*, pages 182–183. John Wiley and Sons, 4th edition, 2012.

[Leh09] Alexander Lehmann. *Towards Mobile Location- and Orientation-Based Detection of Social Situations*. Diploma thesis, 2009.

[MG11] J. J. Macionis and L.M. Gerber. *Sociology 7th Edition*. Toronto: Pearson Prentice Hall, 2011.

[Mic10] Microsoft Corp. PrimeSense Supplies 3-D-Sensing Technology to 'Project Natal' for Xbox 360, 2010. URL: http://www.microsoft.com/en-us/news/press/2010/mar10/03-31PrimeSensePR.aspx.

[Mic11a] Microsoft Corp. Microsoft Releases Kinect for Windows SDK Beta for Academics and Enthusiasts, 2011. URL: http://www.microsoft.com/en-us/news/press/2011/jun11/06-16MSKinectSDKPR.aspx.

[Mic11b] Microsoft Corp. Natural User Interface — Skeletal Tracking, 2011. URL: http://msdn.microsoft.com/en-us/library/hh973074.

[Mic11c] Microsoft Corp. Using the Combined Motion API for Windows Phone, 2011. URL: http://msdn.microsoft.com/library/hh202984.aspx.

[MR09] Gerard Pons Moll and Bodo Rosenhahn. Ball joints for Marker-less human Motion Capture. *2009 Workshop on Applications of Computer Vision (WACV)*, pages 1–8, December 2009. doi:10.1109/WACV.2009.5403056.

[MSSD06] U. Maurer, A. Smailagic, D.P. Siewiorek, and M. Deisher. Activity Recognition and Monitoring Using Multiple Sensors on Different Body Positions. *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, pages 113–116, 2006. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1612909, doi:10.1109/BSN.2006.6.

[Nas05] Steven Nasiri. A Critical Review of MEMS Gyroscopes Technology and Commercialization Status. 2005.

[NBW06] M. Newman, A.-L. Barabasi, and D. J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006.

[Nis96] Kazuhito Nishida. United States Patent 5,497,196 — Video camera having an adaptive automatic iris control circuit, 1996. URL: http://patft1.uspto.gov/netacgi/nph-Parser?patentnumber=5497196.

[NSY] Ben Nham, Kanya Siangliulue, and Serena Yeung. Predicting Mode of Transport from iPhone Accelerometer Data.

[Nyq28] Harry Nyquist. Certain topics in telegraph transmission theory. *Trans. AIEE*, 47:617–644, 1928. URL: http://replay.web.archive.org/20060706192816/http://www.loe.ee.upatras.gr/Comes/Notes/Nyquist.pdf.

[OSB99] Alan V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-time signal processing*. Prentice Hall, Upper Saddle River, N.J., 1999.

[Pen09] Alex Pentland. Reality Mining of Mobile Communications: Toward a New Deal on Data. In *Technology Report 2008 — 2009 Mobility in a Networked World*, pages 75–80. 2009.

[Pla50] R. L. Plackett. Some Theorems in Least Squares. In *Biometrika*, pages 149–157. 1950.

[PLGR12] Gerard PonsMoll, Laura LealTaix, Juergen Gall, and Bodo Rosenhahn. Data-driven Manifolds for Outdoor Motion Capture. *Theoretic Foundations of Computer Vision: Outdoor and Large-Scale Real-World Scene Analysis*, pages 1–25, 2012.

[Por11] Mike Porteous. Introduction to Digital Resampling. 2011.

[RDML] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity Recognition from Accelerometer Data. *Energy*, pages 1541–1546.

[RM00] P. Read and M.P. Meyer. Film Projection — Frame rates. In *Restoration of Motion Picture Film*, pages 24–26. Butterworth-Heinemann, 2000.

[RN10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence — A Modern Approach*, volume 23. Alan Apt, 3rd edition, June 2010.

[Ros07] S. M. Ross. Section 2.6. — Moment Generating Functions. In *Introduction to Probability Models*, page 70. Academic Press, 2007. doi:9780125980623.

[RWY+09] Dahai Ren, Lingqi Wu, Meizhi Yan, Mingyang Cui, Zheng You, and Muzhi Hu. Design and Analyses of a MEMS Based Resonant Magnetometer. *Sensors*, 9(9):6951–6966, September 2009. URL: http://www.mdpi.com/1424-8220/9/9/6951/, doi:10.3390/s90906951.

[SCCD] J. Solà i Carós, O. Chételat, P. Celka, and S. Dasen. Very Low Complexity Algorithm for Ambulatory Activity Classification. *Statistics*.

[SFC+] Jamie Shotton, Andrew Fritzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images.

[Sha49] Claude E. Shannon. Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 37(1):10–21, 1949. URL: http://www.stanford.edu/class/ee104/shannonpaper.pdf.

[SL11] A. A. Salah and B. Lepri. Human Behavior Understanding for Inducing Behavioral Change: Application Perspectives. In *Proceedings of the second International Workshop for Human Behavior Understanding, Amsterdam*, pages 5–9, 2011.

[SMN11] Loren Schwarz, Diana Mateus, and Nassir Navab. Recognizing Multiple Human Activities and Tracking Full-Body Pose in Unconstrained Environments. *Pattern Recognition*, 2011.

[Sny05] Jan A. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer, 2005.

[Spr88] T. Springer. Sliding FFT computes frequency spectra in real time. *EDN Magazine*, pages 161–170, 1988.

[ST60] R. G. D. Steel and J. H. Torrie. *Principles and Procedures of Statistics*. McGraw-Hill, New York, 1960.

[STM08] STMicroelectronics. MEMS motion sensor: 3-axis smart digital output 'nano' accelerometer LIS331DL — Datasheet. (April), 2008.

[STM11] STMicroelectronics. First generation digital compass — LSM303DLH. 2011.

[STM12] STMicroelectronics. MEMS motion sensor: 3 axis analog output gyroscope A3G4250D — Datasheet. (February), 2012.

[The61] Henri Theil. *Economic Forecasts and Policy*. North, Amsterdam, Holland, 1961.

[TLH11] M J Thompson, M Li, and D A Horsley. Low power 3-axis lorentz force navigation magnetometer. *Response*, (3):593–596, 2011.

[VCY05] Michail Vlachos, Vittorio Castelli, and Philip Yu. On Periodicity Detection and Structural Periodic Similarity. In *Proceedings of the Fifth SIAM International Conference on Data Mining*, pages 449–460, Hawthorne, NY, 2005. IBM T.J. Watson Research Center.

[VM04] Michail Vlachos and Chris Meek. Identifying Similarities , Periodicities and Bursts for Online Search Queries. *Power*, 2004.

[VSP09] A Vinciarelli, H Salamin, and M Pantic. Social Signal Processing: Understanding Social Interactions through Nonverbal Behavior Analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami, FL, 2009. doi:10.1109/CVPRW.2009.5204290.

[WF02] Greg Welch and Eric Foxlin. Motion Tracking : No Silver Bullet, but a Respectable Arsenal. *Ieee Computer Graphics And Applications*, (December), 2002.

[ZA07] Wiebren Zijlstra and Kamiar Aminian. Mobility assessment in older people: new possibilities and challenges. *European Journal of Ageing*, 4(1):3–12, February 2007. URL: http://www.springerlink.com/index/10.1007/s10433-007-0041-9, doi:10.1007/s10433-007-0041-9.