HAW HAMBURG

BACHELORTHESIS
Phillip Schackier

Algorithmic activity detection based on motion and position sensors

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Phillip Schackier

Algorithmic activity detection based on motion and position sensors

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai v. Luck
Zweitgutachter: Jan Schwarzer, Ph.D.

Eingereicht am: 19. Februar 2021

Phillip Schackier

Thema der Arbeit

Algorithmische Aktivitätserkennung auf Basis von Bewegungs- und Lagesensoren

Stichworte

Data Science, Segmentierung

Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit der Segmentierung von repetitiven Sportübungen. Die Segmentierung solcher Zeitreihendaten ist ein wissenschaftlich interessantes Problem, da diese die Analyse einzelner Wiederholungen mit Methoden wie dem maschinellen Lernen ermöglicht. Um zu ermitteln, ob ein Segmentierungsalgorithmus implementierbar ist, der auf einem Mikroprozessor ausführbar ist, wird in dieser Arbeit ein prototypischer Entwurf entwickelt. In einem Experiment führten Probanden Übungen durch, während sie Sensoren am Körper trugen. Ein Segmentierungsalgorithmus wurde entwickelt und anschließend genutzt, um die gesammelten Daten zu verarbeiten. Die Segmentierung wurde evaluiert, indem die algorithmisch bestimmten Segmentierungspunkte mit einer Ground Truth verglichen wurden, welche aus Videoaufnahmen basierte. Der Algorithmus wies lineare Laufzeitkomplexität und eine geringe Fehlerkennung auf. Bei einer Fenstergröße von 501 Datenpunkten ergaben sich 0,02% falsch negative Segmentierungspunkte und 0,04% falsch positive Segmentierungspunkte. Die Studie zeigt, dass ein geeigneter Algorithmus implementierbar ist.

Phillip Schackier

Title of Thesis

Algorithmic activity detection based on motion and position sensors

Keywords

Data Science, Segmentation

Abstract

This thesis deals with the segmentation of repetitive sports exercises. The segmentation of such time series data constitutes a scientifically interesting problem, as it enables analysis of individual repetitions using techniques such as machine learning. To determine whether it is possible to implement a segmentation algorithm that can run on a microprocessor unit, this thesis proposes a prototypical design. An experiment was conducted in which test persons performed exercises while equipped with sensors. A segmentation algorithm was devised and subsequently used to process the signal data collected. The segmentation was evaluated by comparing the algorithmically determined segmentation points with a ground truth based on video recordings. The algorithm exhibited linear time complexity and a low failure rate. A window size of 501 data points resulted in 0.02% false negative segmentation points and 0.04% false positive segmentation points. The study shows that a suitable algorithm can be implemented.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ARC activity recognition chain.

CSV comma-separated-values.

fps frames per second.

IMU inertial measurement unit.

MLA magnitude of linear acceleration.

MPU microprocessor unit.

1 Introduction

Starting in the 1990s, many designers began to experiment with wearable computers as a result of the developments in small-scale computing [24]. Various devices had self-tracking capabilities, allowing users to track their bodies [24]. These developments have simplified self-tracking [38]. Probably the most well-known website of the self-tracking community is the Quantified Self website [24].[1] Many of the early adopters of self-tracking technology are athletes [38]. Self-tracking smartphone applications for fitness tracking are available, which can detect the gym exercise performed [20].

According to Guo et al. [14], an increasing number of people exercise at home without an instructor. They explain that doing so increases the likelihood of injury caused by incorrect execution of the exercises. Furthermore, they note that some of these people wish to document their exercise for self-tracking purposes. Lupton [24] identifies the Quantified-Selfers as one such group. To evaluate each repetition of an exercise, one must first be able to detect these repetitions. The repetitions are distinguished in a continuous stream of exercise data by a segmentation algorithm [2]. Lin et al. [23] describe the segmentation of movement data as a process of determining segmentation points within a data stream and dividing it into segments. They also state that, in exercise and physical rehabilitation, the segments are used to assess the quality of the repetitions that constitute the exercise.

Several studies have employed computerized systems using inertial measurement unit (IMU) sensors to evaluate exercises, such as Baumbach et al. [1], Bevilacqua et al. [2], and Huang et al. [16]. Huang et al. [16] identified several reasons why the classification of human movements is difficult, including the synchronicity of movements, the degrees of freedom for each movement, the variability of repetitions of the same exercise by the same person, and the variability between different persons. This thesis deals with the problem of segmenting the repetitive time series data gathered in an experiment for part

[1]https://quantifiedself.com (Accessed 2021-01-19)

of the MoGaSens project.[2] The MoGaSens project aims to develop a kind of smart shirt that measures several parameters that are relevant to medical or sports scientists using sensors such as accelerometers and gyroscopes. Specifically, this thesis involves data analysis of human exercise movement. Several authors including Ishii et al. [18] and Kowsar et al. [21] use peak finding algorithms to segment accelerometer signals.

Similarly, in this work, IMU sensors were used and data were collected in an experimental setup. The findings from this study may be used, for instance, to determine the quality of exercise execution by using machine learning classification algorithms. Moreover, the topic of this thesis is related to the field of metrology, specifically waveform analysis, which includes smoothing of a waveform and subsequent segmentation. Prazdny [28], for instance, contributed such an algorithm in 1983. The segmentation step of an existing analysis pipeline for IMU sensors – called the activity recognition chain (ARC) [5] – served as a methodological blueprint for the work carried out for this thesis.

Within the context of computer science, this thesis deals with part of the groundwork to be completed before the application of artificial intelligence, such as machine learning classification models, to real-world data. The topic of motion segmentation is relevant because it is used in many applications, such as imitation learning, exercise, and physical rehabilitation [23]. This thesis provides insight into the segmentation of repetitive time series data by means of an experimental design. The purpose is to lay a theoretical foundation for the smart shirt developed by Hamburg Applications MES UG[3] as part of the MoGaSens project.

1.1 Research Aim and Objectives

The aim of this thesis was to devise a segmentation algorithm for IMU data. The algorithm must be able to segment an accelerometer signal accurately enough for the use of the segments in feature extraction. Additionally, the algorithm must be efficient enough in terms of space and time to be able to run on a microprocessor unit (MPU) and allow for timely evaluation of each repetition's quality.

This research aim was achieved using the following objectives:

1. Investigate motion segmentation approaches in the literature.

[2]https://csti.haw-hamburg.de/project/mogasens/ (Accessed 2021-01-19)
[3]https://hamburg-applications.com (Accessed 2021-01-19)

2. Propose an experimental setup to collect the required data.
3. Describe and implement an algorithm to segment the accelerometer signal from sets of repetitions of push-ups and squats.
4. Evaluate the devised algorithm.

The research questions are as follows: Can such an algorithm be implemented? How suitable is it for the task? What limitations exist?

1.2 Thesis Structure

This thesis is divided into five chapters. Following the introduction, Chapter 2 reviews the literature and introduces the problem statement. Chapter 3 presents a solution to the problem. Chapter 4 evaluates the solution and discusses its limitations. Finally, Chapter 5 concludes the thesis and suggests directions for future research.

2 Analysis

This chapter reviews the literature on the Quantified Self movement, sets out the research context of this thesis, and reviews related work on preprocessing and segmentation of time series data.

2.1 Quantified Self

People have long engaged in self-tracking for self-improvement and self-reflection [24]. The Quantified Self movement and the innovations in computerized self-tracking technology, however, are a recent phenomenon [24]. Swan [34] describes the Quantified Self movement as a trend of individuals or groups tracking biological, physical, behavioral, or environmental information. "Quantified Self" refers to both the name of the Quantified Self community and the process of self-tracking [6]. The digital devices used gather detailed, continuous data on everyday practices, social interactions, and bodily functions [24]. The data collected may be quantitative or qualitative (e.g., step counts or moods) [22].

Wolf [38] describes in his article "Know Thyself: Tracking Every Facet of Life, from Sleep to Mood to Pain, 24/7/365," how he, along with his fellow Wired writer Kevin Kelly, decided to create a website, called Quantified Self, to track the self-tracking systems available. He explains that methods of quantitative assessment used to be laborious, but now much of the data-gathering can be automated and the record-keeping and analysis can be delegated to web applications. Furthermore, he describes how open-source software for random experience sampling avoids introducing biases that manual journal-keeping would introduce.

The Quantified Self community describes itself as an international community of users and makers of self-tracking tools who share an interest in self-knowledge through numbers [32]. The Quantified Self website provides discussion forums, supports regional meetings, hosts two annual international conferences, and publishes a blog [24]. Quantified Self has become a community engaged in self-monitoring and the development of self-monitoring technology [6]. Hanze University of Applied Sciences established an academic research institute named the Quantified Self Institute [24].[1] The Quantified Self website details over 500 self-tracking tools, enabling tracking of geolocation, health, fitness, weight, sleep, diet, mood, and emotion [24]. Quantified-Selfers use a variety of tools for self-tracking, including data collection tools, such as commercial hardware, spreadsheet software, productivity tracking software, and custom-built software [6]. Data exploration tools include spreadsheet software for running simple statistics and creating graphs, custom-built software, commercial websites, commercial software, and statistical software such as R [6].

Lupton [24] describes the use of sensors as a pivotal feature in self-tracking. She also states that biosensors include reactive agents that can respond to changes in bodily functions and indicators (e.g., blood glucose, hormones, enzymes, or oxygen levels). Additionally, she notes that sensors such as GPS, digital compasses, gyroscopes, and accelerometers are typically included in smartphones and that they can be employed to track users' geolocation and movements.

Some commercial self-tracking products such as the Nike+ FuelBand or the family of Fitbit products enable the average consumer to track their exercise [37]. The Fitbit Sense [10] smartwatch includes a heart rate sensor, a gyroscope, an altimeter, a triaxial accelerometer, and a skin temperature sensor. Fitbit developed an activity-recognition solution called SmartTrack for their Fitbit Charge HR and Fitbit Surge products. SmartTrack can automatically detect certain exercises, such as walking, running, outdoor cycling, workouts on an elliptical trainer, and high-movement sports (tennis, basketball, soccer, etc.) [9]. Fitbit states that activities that last at least 10 minutes can be captured using their technology. Additionally, self-tracking smartphone applications for fitness tracking, such as Google Fit [13], also support automatic detection of activities such as walking, running, and cycling.

2.2 Research Context

This section gives a short overview of the MoGaSens project, describes the IMU sensor hardware used in this study, and introduces the problem statement.

[1] http://qsinstitute.com (Accessed 2021-01-19)

2.2.1 MoGaSens Project

This thesis builds on research conducted as part of the MoGaSens project,[2] in which a smart shirt is being developed that collects accelerometer and gyroscope data. The research was carried out by a research team at the Creative Space for Technical Innovations at the Hamburg University of Applied Sciences.[3] Additional project partners are the University of Hamburg,[4] whose sports scientists labeled every push-up as correct or incorrect, and Hamburg Applications MES UG,[5] who are developing the smart shirt.

The project's overall aim is to develop a sensor system (i.e., the smart shirt) that can detect the kind of exercise performed and assess the quality of each repetition. To this end, sensor data was processed and subsequently analyzed so that each exercise repetition could be evaluated.

2.2.2 Hardware

This study used four Bosch BMI160 [3] sensors. These sensors contain both an accelerometer and a gyroscope [3]. The accelerometer measures acceleration of gravitational force (g-force), and the gyroscope measures rotation in degrees per second. Both the accelerometer and the gyroscope are triaxial (x-axis, y-axis, and z-axis) [4]. Appendix A.1 includes a detailed description of the sensors' characteristics.

For the task of segmenting push-up and squat exercises, a sensor measuring the height of the test person would have been ideal, as a local maximum would indicate the start of an exercise repetition and the end of the preceding repetition. Such a sensor was not available for the project.

The four sensors were used to measure the acceleration (g-force) at different points on the test person's body as repetitions of an exercise (push-ups or squats) were performed. The acceleration does not directly translate to the respective height, however, and reconstructing the movement from the sensor data is challenging.

The sensors' raw data channels contained a degree of white noise, which is typical for IMUs such as the Bosch BMI160 [21]. The Bosch BMI160 supports three filtering modes

[2]https://csti.haw-hamburg.de/project/mogasens/ (Accessed 2021-01-19)
[3]https://csti.haw-hamburg.de (Accessed 2021-01-19)
[4]https://uni-hamburg.de (Accessed 2021-01-19)
[5]https://hamburg-applications.com (Accessed 2021-01-19)

[4], but the effect of the built-in filtering was inadequate for the task, and so additional software-based filtering (a moving average filter or a Butterworth filter) was used.

Bosch BMI160 sensors were used, because the same sensors are used by Hamburg Applications MES UG for the smart shirt. Reasons for their use included their relatively small size (2.50 x 3.00 mm^2 footprint and 0.83 mm in height), low power consumption (typically 925 µA), and low cost [4].

Several steps had to be taken before the sensors could be used. The hardware timestamps of the Bosch BMI160 sensors overflow after a while, and the timestamp values must be restored. Furthermore, the accelerometer and gyroscope readings are converted to floating point values by dividing them by the appropriate divisors. Additionally, the Euclidean norm is calculated for the three axes of the accelerometer and the gyroscope. Subsequently, a low-pass filter is applied to the raw signal data to account for the high degree of white noise. The preprocessing steps that the use of the sensors necessitates are described in detail in section 3.3.2.

2.2.3 Problem Statement

This thesis proposes an algorithm to segment time series data.[6] It is assumed that the exercise to be performed is known in advance. Additionally, the start and end time points of the set of exercise repetitions are also assumed to be known.

Segmentation of the preprocessed data is performed to determine in which segments of the data streams the relevant information is most often to be found [5]. Segmenting a stream of sensor data is challenging, however, as the fluid manner in which humans move means that successive activities blur into one another [5]. All motion sensors exhibit some degree of white noise, which must be removed before analysis [21]. This thesis focuses on the problem of segmenting such sensor data.

A particular challenge was that the segmentation solution developed had to be executable online (i.e., in near real-time) on a small, embedded device that is integrated into the smart shirt. This requirement for the solution to be able to run in near real-time derived from the MoGaSens project, as the smart shirt must allow for rapid feedback to the

[6] Another approach to activity recognition, in contrast to the algorithmic approach taken in this thesis, would be to use neural networks to assess a stream of labeled raw data or to use neural networks for segmentation [26]. This thesis favors the algorithmic approach, because the amount of data available was limited to 18 labeled data sets of push-up exercises and 3 unlabeled data sets of squat exercises.

wearer regarding the quality of each exercise repetition performed. Feedback for the exercise repetition performed should be delivered quickly to the user, with a delay of approximately one repetition. Such short feedback cycles should allow the wearer of the smart shirt to quickly improve their performance (e.g., by adjusting their posture). This requirement necessitates that the runtime complexity and the memory usage of the algorithm are appropriately low. Another challenge is that data loss due to hardware limitations necessitates interpolation of the missing data.

2.3 Related Work

Many techniques to recognize and classify exercises can be found in the literature. This section reviews the literature and presents various preprocessing techniques to enable segmentation. Additionally, it presents various techniques for segmenting the data so that features can be extracted from the resulting segments.

The preprocessing and segmentation steps are derived from the ARC, which was introduced by Bulling et al. [5] as a model for activity recognition. It is a sequence of signal processing, pattern recognition, and machine learning. The ARC is similar to the knowledge-discovery-in-databases process proposed by Fayyad et al. [8]. The ARC is used as a model for a data analysis pipeline consisting of preprocessing, segmentation, feature extraction, and classification. This thesis focuses on the preprocessing and segmentation steps.

2.3.1 Preprocessing

According to Kowsar et al. [21] raw signal data are filtered using a filter such as a low-pass filter (e.g., a Butterworth filter [2]) before the signal is segmented and features are subsequently extracted. Segmentation of unfiltered data is difficult, because IMU signals typically contain white noise [21].

Related studies used filters in varying configurations. For example, Baumbach et al. [1] used a 3$^{\text{rd}}$ order median filter to deal with noisy values in the accelerometer data resulting from the sensors' inaccuracy, noise in the sensors' signals, and unexpected behavior of the test person. Bevilacqua et al. [2] employed a 4$^{\text{th}}$ order Butterworth filter to remove noise introduced by the elastic vibration of the strap securing the IMU device to the test person's body. They subsequently applied a min-max normalization to all signals [2]. Similarly, Giggins et al. [12] used a 4th order Butterworth filter with a cutoff frequency of 20 Hz. Likewise, Huang et al. [16] used 5th order Butterworth filters with -1 dB ripple for accelerometer signals and 4th order Butterworth filters for the gyroscope signals. They used a cutoff frequency of 50 Hz for the thigh and shin/foot mounted sensors and 25 Hz for the other sensors [16]. Seiffert et al. [31] and Zhang et al. [39] used 2nd order Butterworth low-pass filters with a cutoff frequency of 3 Hz.

2.3.2 Time Series Data Segmentation

Several methods for segmentation have been used in the literature. Both Lin et al. [23] and Dreher et al. [7] have proposed frameworks to compare and evaluate movement-segmentation algorithms. Lin et al. [23] give an overview of the segmentation approaches, as shown in figure 2.1.

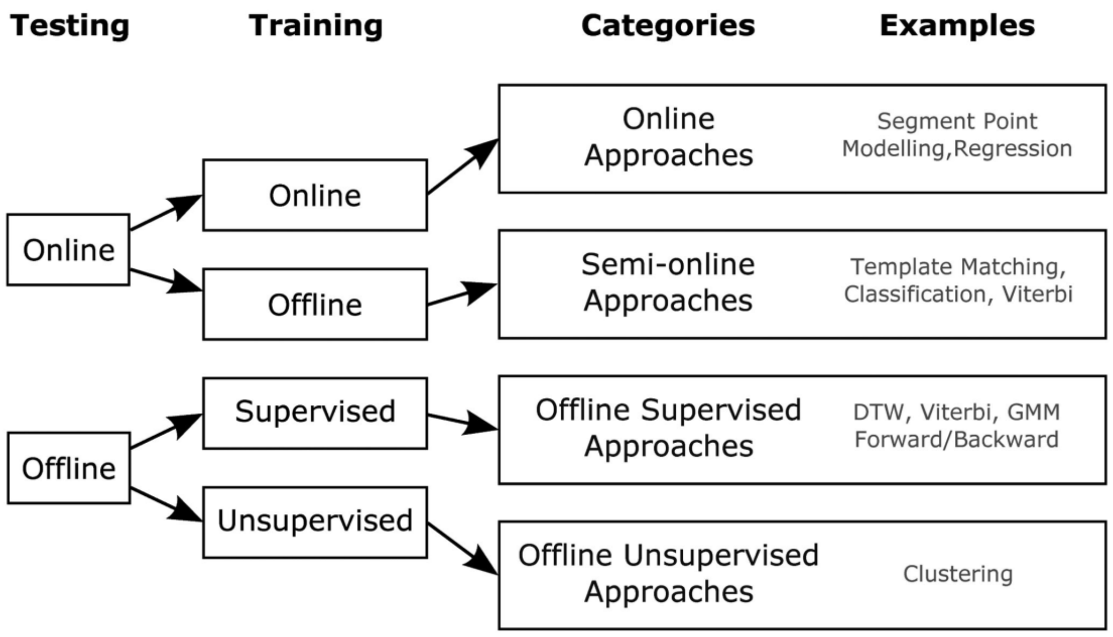

Testing
Training
Categories
Examples
Online
Online
Offline
Online Approaches
Segment Point Modelling,Regression
Semi-online Approaches
Template Matching, Classification, Viterbi
Offline
Supervised
Unsupervised
Offline Supervised Approaches
DTW, Viterbi, GMM Forward/Backward
Offline Unsupervised Approaches
Clustering

Figure 2.1: An overview of common segmentation approaches.

Source: Lin et al. [23, p. 329]

© 2016 IEEE

One approach is to segment the filtered input signal by local extrema [18, 14, 21] or by zero crossings [2, 16]. Other segmentation approaches include the Viterbi algorithm, and clustering approaches [23].

Ishii et al. [18] implemented a real-time segmentation solution consisting of a series of steps. First, the norm of the sensor's axes is calculated, which serves as a synthetic acceleration signal. They state, however, that segmenting through this synthetic signal has the disadvantage of reducing the differences between movements that are similar but occur along a different axis. Second, unspecified smoothing is applied to the signal. Third, segmentation is performed by local maxima, which they call "peaks." They used a segmentation algorithm with a sliding window of 0.25 s to find the local maxima.

Another approach was proposed by Guo et al. [14], who introduced the idea of the magnitude of linear acceleration (MLA). First, the MLA, which is a combination of the x-, y-, and z-axes, is calculated as follows:

$$MLA(i) = \sqrt{(a(i)_x)^2 + (a(i)_y)^2 + (a(i)_z)^2} - g$$

where a represents the acceleration and g the acceleration of gravity. The short-time energy of the MLA is used as the input signal of the segmentation algorithm. The segmentation is performed by local minima using sliding windows of short lengths.

A different approach was presented by Kowsar et al. [21], who implemented real-time segmentation for weight training exercises. In contrast to other studies, the segmentation is performed through a single sensor axis, called the axis of effect. They explain how segmenting through a single axis is suitable for weight training exercises. A Kalman filter is applied to the input signal, and the segmentation is performed by local minima.

The approaches proposed by Bevilacqua et al. [2] and Huang et al. [16] involve segmentation by zero crossings of filtered accelerometer data. Both approaches use Butterworth filters to filter the raw accelerometer signal. Bevilacqua et al.'s approach uses thresholds to discard some segmentation points and uses k-means clustering. The centroid of each cluster is used as the representative point for a signal window. These serve as candidates for segmentation points. Huang et al.'s approach employs a fast template matching algorithm for segmentation after having reduced the signal noise using a Butterworth filter.

Manual segmentation of accelerometer data was employed by Taylor et al. [35]. Baumbach et al. [1] segmented filtered data with a window size of 5 s. They used a 3rd order median filter to filter the data. Another approach, based on several features, including zero-velocity crossings, was proposed by Sarsfield et al. [30].

Rodrigues et al. [29] developed a framework called Symbolic Search in Time Series, which is a syntactic tool for pattern searching in time series data. The framework was used by Pereira et al. [27], who filtered their data using a low-pass filter. The accelerometer and gyroscope signals were fused together using a 2$^{\text{nd}}$ order complementary filter.

Gharghabi et al. [11] proposed an algorithmic approach, called the fast low-cost unipotent semantic segmentation algorithm. Their algorithm was domain agnostic and they developed a variant suitable for online segmentation.

In summary, segmentation based on accelerometer data seems to be the most common approach. Huang et al. [16] concluded that acceleration data are more effective than gyroscope data for most types of movement analysis.

2.4 Summary

The goal of the MoGaSens project is to develop a smart wearable (specifically, a smart shirt) that can evaluate the quality of each repetition of an exercise the wearer performs. To achieve this goal, Bosch BMI160 sensors are integrated into the smart shirt. A machine learning model that is run on an MPU will classify each exercise repetition as correct or incorrect. This thesis contributes to the project by devising an online (i.e., near real-time) segmentation algorithm that can segment a filtered accelerometer signal so that features can be extracted from the segments.

The accelerometer data will be preprocessed before segmentation. The preprocessing will primarily consist of applying a low-pass filter to the raw data signal. Then the data will be segmented in a uniform manner so that features can be extracted from the segments in a way that is suitable for use with a machine learning classifier. For this purpose, a segmentation algorithm must be devised that has non-exponential time complexity and non-exponential memory usage. Furthermore, the segmentation algorithm should, in principle, be usable in a (near) real-time environment and be suitable for segmenting push-ups and squats with high accuracy.

3 Design

First, this chapter describes the experimental setup that was used to gather the required sensor data. Second, it describes the design of the software to analyze the quality of push-ups and squats. Section 3.2 describes the visionary design (i.e., the system as envisioned), and section 3.3 describes the prototypical design that algorithmically segments the time series data by batch-mode processing.

3.1 Experimental Setup

To gather sensor data for analysis, an experiment was conducted in which test persons performed sets of an exercise while equipped with IMUs. Four Bosch BMI160 sensors were attached to the test person's body while they performed push-ups or squats. Sensors were attached to the side of each arm, to the abdomen, and to the chest. Each sensor contained a triaxial accelerometer and a triaxial gyroscope. The abdomen and chest sensors were oriented so that the positive z-axis was facing toward the ground. The sensors attached to the sides of the arms had their positive x-axis facing toward the ground when the test person was standing upright. Figure 3.1 gives a visualization of the sensor orientation.

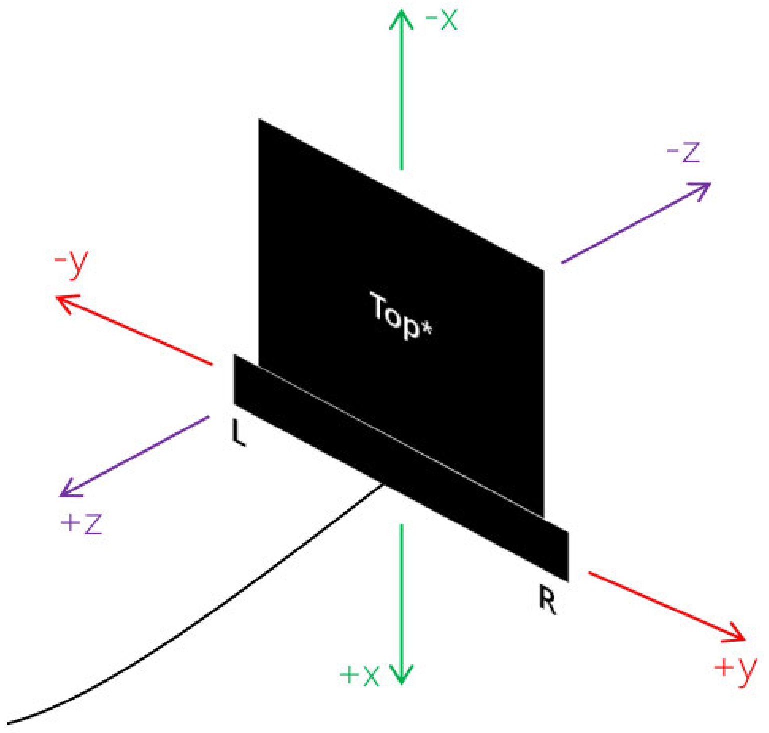


Figure 3.1: A visualization of the orientation of the sensor axes.

The test persons were video recorded while they performed the exercises. The videos served as ground truth data to evaluate the accuracy of the proposed segmentation algorithm. The videos were recorded with a resolution of 1,280 × 720 pixels and 30 frames per second (fps). Two cameras were used to record the videos: one recorded the test person performing the exercise from the front; the other recorded the test person from the side. The two camera signals were merged into single MP4 [19] video files for each recording.

Six test persons performed the exercises. Recordings were made of 18 sets of push-ups (three per person) and two sets of squats. For the first nine sets of push-ups, a sampling rate of 250 Hz was used; for the squats and the other nine sets of push-ups, the sampling rate was 200 Hz. The data were collected so that each repetition could be classified into one of two distinct categories: correct and incorrect. For both the accelerometer and the gyroscope, the normal filter mode was configured in the Bosch BMI160 sensors. In the repetitive time series data, activities were recognized by means of segmentation. It was assumed that a push-up takes between 1.40 s and 2.50 s. These numbers are derived from

Hsu et al. [15], who considered seven push-ups in 10 s to be fast, five push-ups in 10 s to be regular, and four push-ups in 10 s to be slow.

3.2 Visionary Design

In the visionary design, the user wears a smart shirt that has IMUs and an MPU integrated into its fabric. The user defines a workout plan using a smartphone application and then performs the exercises as instructed.

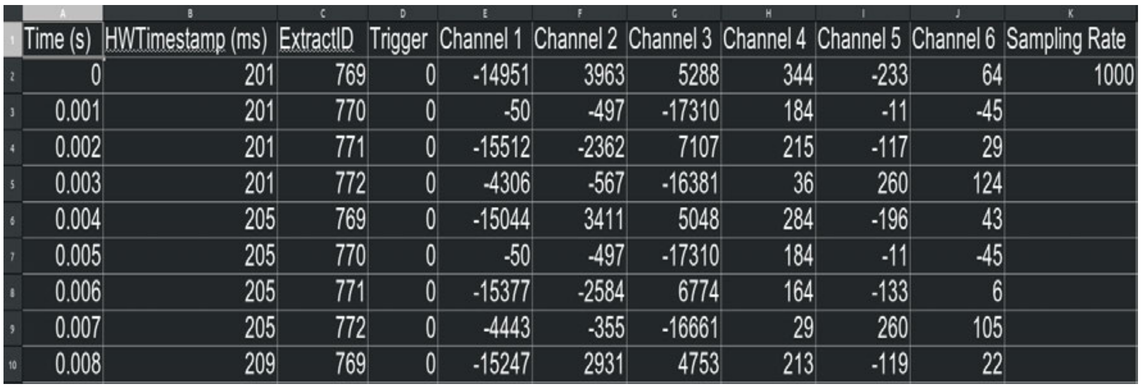
The IMUs collect accelerometer and gyroscope data, which are transmitted to the MPU via radio. A pre-trained machine learning model to classify a given segment runs on the MPU. The MPU buffers the data until sufficient data are gathered according to the window size of the segmentation algorithm. The gathered data are then repeatedly processed as follows:

1. The Euclidean norm of the accelerometer axes and of the gyroscope axes is calculated.
2. The four sensors are normed together.
3. A 3^{rd} order Butterworth filter is applied to the normed signal.
4. Local extrema are found in the buffered data.
5. For each segment (i.e., the data points between two local extrema) a feature vector is calculated.
6. Each feature vector is fed into the pre-trained machine learning model.
7. The machine learning model classifies each exercise repetition as correct or incorrect.
8. The classification of the most recently detected repetition is sent to the smartphone application using a network connection, and the application then communicates the result to the user.

3.3 Prototypical Design

As the smart shirt was not available when this study was conducted, a prototype was developed to investigate the suitability of a segmentation algorithm that segments by local extrema for solving the problem of segmenting sets of exercise repetitions. The prototype employed batch-mode processing as opposed to live (i.e., online) segmentation. As described in section 3.1, the test persons wore four IMUs. While the test person performed push-ups or squats, the signals of the accelerometer and gyroscope sensors were transmitted to another computer via radio, where they were recorded and saved as comma-separated-values (CSV) [33] files.

The CSV files contain 11 columns. See figure 3.2 for details.



Time (s)	HWTimestamp (ms)	ExtractID	Trigger	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Sampling Rate
0	201	769	0	-14951	3963	5288	344	-233	64	1000
0.001	201	770	0	-50	-497	-17310	184	-11	-45	
0.002	201	771	0	-15512	-2362	7107	215	-117	29	
0.003	201	772	0	-4306	-567	-16381	36	260	124	
0.004	205	769	0	-15044	3411	5048	284	-196	43	
0.005	205	770	0	-50	-497	-17310	184	-11	-45	
0.006	205	771	0	-15377	-2584	6774	164	-133	6	
0.007	205	772	0	-4443	-355	-16661	29	260	105	
0.008	209	769	0	-15247	2931	4753	213	-119	22	

Figure 3.2: The columns of the CSV file.

The Time column contains the time of arrival of the sensor data in seconds. The HW-Timestamp column contains the hardware timestamps in milliseconds as recorded by the IMU. The ExtractID column lists unique identifiers: 769 represents the left arm sensor, 770 the abdomen sensor, 771 the right arm sensor, and 772 the chest sensor. (Other ExtractID values, including one for the battery, are not relevant to this thesis.) The Trigger column is currently unused and is always 0. The columns Channel 1 to Channel 6 contain the raw sensor readings. Channel 1 is the x-axis of the accelerometer, channel 2 the y-axis, and channel 3 the z-axis. Likewise, channel 4 is the x-axis of the gyroscope, channel 5 the y-axis, and channel 6 the z-axis. The Sampling Rate column is currently unused.

3.3.1 Activity Recognition Chain

Figure 3.3 shows the ARC proposed by Bulling et al. [5], on which the proposed data analysis pipeline is based. The implementation consists of the segmentation and the required preprocessing.

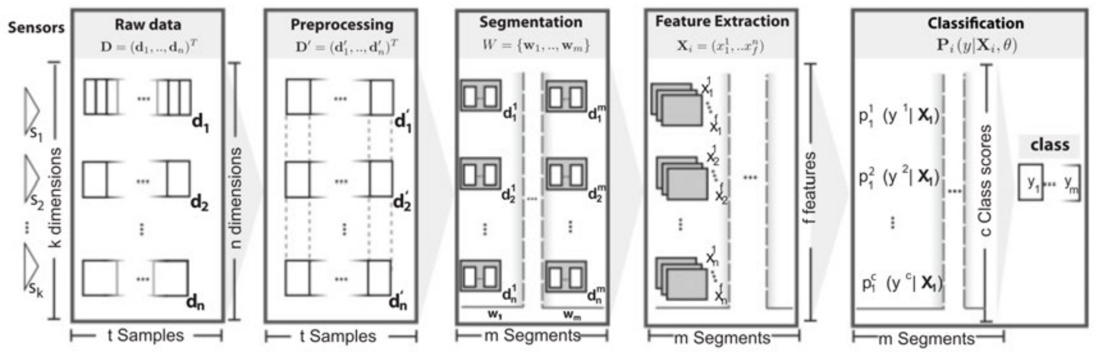


Figure 3.3: Visualization of the activity recognition chain, illustrating the steps of the analysis pipeline.

Source: Bulling et al. [5, p. 8]

Copyright 2014 ACM

In the first stage (referred to as the raw data stage), raw data are acquired using sensors attached to the body. The sensors can use different sampling frequencies. The raw sensor data may be corrupted by artifacts [5].

The purpose of the second stage (the preprocessing stage) is to remove such artifacts, reduce noise, and prepare the signals for feature extraction. They describe how the preprocessing of accelerometer and gyroscope signals may involve calibration, unit conversion, normalization, resampling, synchronization, and signal-level fusion [5].

In the third stage (segmentation), the segments of preprocessed data that are likely to contain information about activities are identified. Each data segment is defined by its start and end times. Bulling et al. state that one approach for time series data segmentation is the sliding window approach. In this approach, a window is moved over the time series data to extract segments. The window size directly influences the delay of the recognition system. The larger the window size, the longer the feature extraction stage must wait for a new segment to become available [5].

The fourth stage (feature extraction) reduces the signals into features that can be used to discriminate between the activities [5].

What happens in the fifth stage (classification) depends on whether the training or classification mode is used. In the training mode, the extracted features and the ground truth labels are used as input to train the classifier model. In the classification mode, the trained model and the extracted features are used to calculate a score for each class [5].

3.3.2 Implementation

For this thesis, the first three steps of the ARC (raw data, preprocessing, and segmentation) have been implemented.[1] The implementation consists of the following components:

- Sensor hardware consisting of four Bosch BMI160 sensors [3].
- Software to receive the sensor data.
- Preprocessing component written in MATLAB [25].
- Segmentation component written in Python [36].

Data Acquisition

Each test person performed a predefined exercise (push-ups or squats) until exhaustion with sensors attached to their body. While the test person performed the exercise, a video was recorded, which served as the ground truth. Additionally, the repetitions performed were counted for each data set by manually inspecting the videos recorded. Four sensors were used: one attached to the left arm, one to the right arm, one to the abdomen, and one to the chest. Each sensor consisted of a triaxial accelerometer and gyroscope. The sensor data were sent to a computer that aggregated the data into CSV files. Data sets for both push-ups and squats were recorded as CSV files.

[1] For the source code developed for this thesis by the thesis author see `https://github.com/CppPhil/mogasens_csv` (Accessed 2021-02-16).

Preprocessing

The preprocessing stage was implemented by André Jeworutzki.[2] The data sets were preprocessed as follows:

1. The hardware timestamp values were restored from their overflowed values. The timestamps were stored as 16-bit unsigned integers and overflowed given sufficient time.[3]

2. Accelerometer readings greater than or equal to 1.99 g and less than or equal to -1.99 g were deleted, because they are beyond the bounds of the possible sensor values given the sensor sensitivity.

3. Gyroscope readings greater than or equal to 1,999.99 °/s and less than or equal to $-1{,}999.99$ °/s were deleted, because they are beyond the bounds of the possible sensor values given the sensor sensitivity.[4]

4. The missing data were interpolated.

5. The 16-bit signed integer accelerometer values were divided by 16,384 so that they were within -1.99 to 1.99.

6. The 16-bit signed integer gyroscope values were divided by 16.40 so that they were within $-1{,}999.99$ to 1,999.99.[5]

7. The Euclidean norm was calculated for the accelerometer signals and the gyroscope signals. Equation 3.1 defines the Euclidean norm, where I is the IMU (i.e., accelerometer or gyroscope); x, y, and z are the axes of the IMU; and t is the hardware timestamp.[6] The Euclidean norm was calculated for every hardware timestamp for which a data point existed, norming the entirety of each input channel

[2] https://csti.haw-hamburg.de/andre-jeworutzki (Accessed 2021-01-19)

[3] An implementation correcting the hardware timestamps is available at https://github.com/CppPhil/mogasens_csv/blob/master/fix_csv/src/adjust_hardware_timestamp.cpp (Accessed 2021-02-16).

[4] An implementation handling both the accelerometer and gyroscope readings is available at https://github.com/CppPhil/mogasens_csv/blob/master/fix_csv/src/delete_out_of_bounds_values.cpp (Accessed 2021-02-16).

[5] An implementation is available at https://github.com/CppPhil/mogasens_csv/blob/master/fix_csv/src/main.cpp (Accessed 2021-02-16) with the accelerometer readings being adjusted in lines 199 to 200 and the gyroscope readings being adjusted in lines 227 to 228.

[6] An implementation of the Euclidean norm is available at https://github.com/CppPhil/mogasens_csv/blob/master/python/modules/euclidean_norm.py (Accessed 2021-02-16).

into one normed, artificial channel.

$$Norm(t) = \sqrt[2]{I_x(t)^2 + I_y(t)^2 + I_z(t)^2} \tag{3.1}$$

8. A moving average filter with a window size of 125 values was used to filter the three channels of the accelerometer, the gyroscope, and the normed signals for both.[7]

9. A 3rd order low-pass Butterworth filter with a cutoff frequency of 0.80 Hz was used to filter the three channels of the accelerometer, gyroscope, and the normed signals for both.

Figure 3.4 shows raw accelerometer data, including all three axes and their Euclidean norm. The x-axis shows the hardware timestamps in milliseconds, and the y-axis shows the gravitational force. Motion sensors such as accelerometers exhibit some degree of white noise [21], and filtering is often required to remove such sensor noise from the raw data [23].

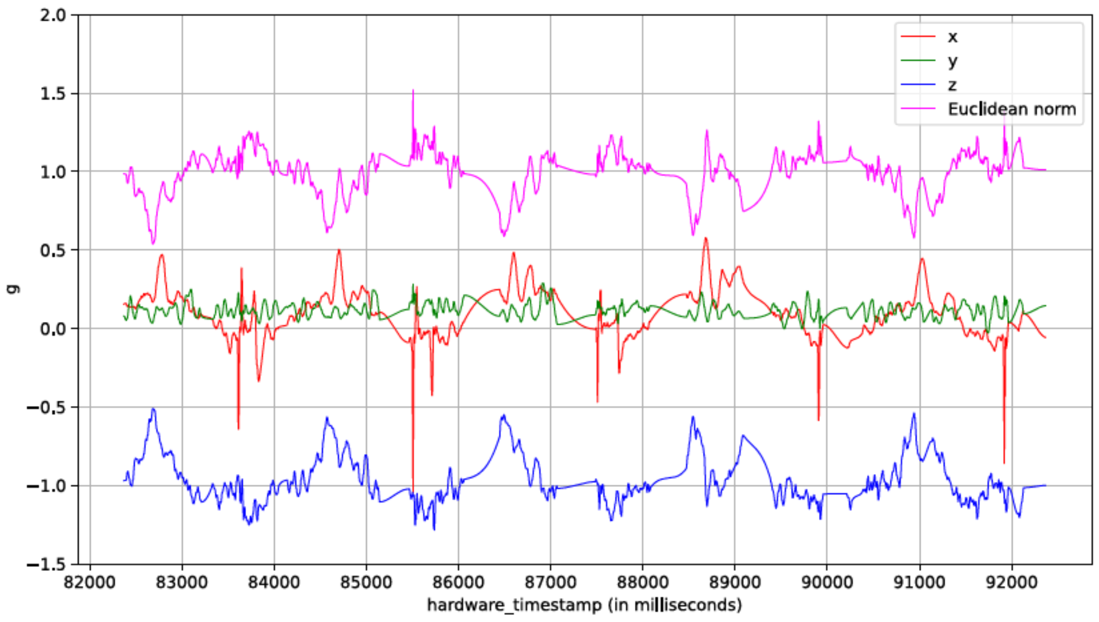


Figure 3.4: Test person 1, set 1 abdomen sensor unfiltered accelerometer signal.

[7] An implementation of the moving average filter is available at https://github.com/CppPhil/mogasens_csv/blob/master/python/modules/moving_average_filter.py (Accessed 2021-02-16).

Two filter approaches were chosen to smooth the incoming raw signal: a moving average filter of 125 samples and a 3rd order Butterworth filter. A push-up was expected to take between 1.40 s and 2.50 s [15] and should contain one local minimum and one local maximum. If the sampling rate is 250 Hz, 125 samples amount to 500 ms; and if the sampling rate is 200 Hz, 125 samples amount to 625 ms. In both cases, the time over which the data points are averaged is below 1.40 s, which accounts for fast push-ups.

Figure 3.5 shows the Euclidean norm of an accelerometer signal of a test person performing push-ups filtered with the moving average filter and the Butterworth filter.

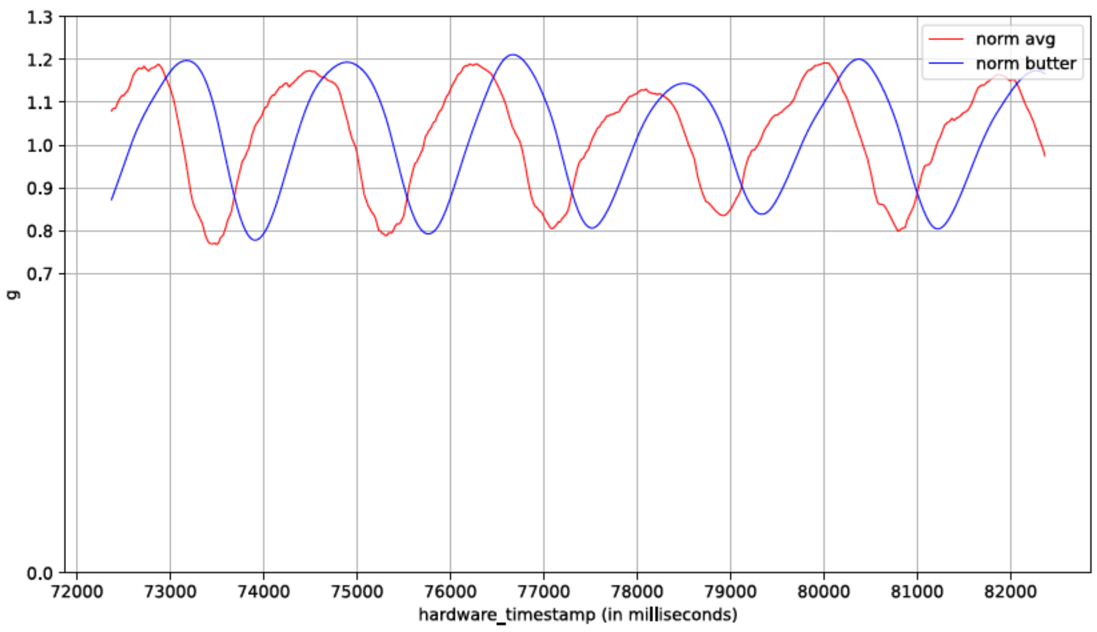


Figure 3.5: Test person 1, set 1 chest sensor filtered accelerometer signal.

After preprocessing, the data sets were exported as CSV files. These CSV files contain raw data, the normed data, and their moving-average and Butterworth-filtered counterparts.

Segmentation

The preprocessed data sets are segmented using an offline (batch processing) fixed-size sliding window algorithm that segments the data by local extrema (local minima, local maxima, or both). As the Euclidean norm is used, sliding windows must also be used, according to Imani et al. [17].

The data to be segmented are read from the preprocessed CSV files.

The proposed algorithm has the following parameters:

1. *skipWindow*: If set, the algorithm will ignore the remaining data points in the sliding window if the current data point has been determined to be a local extremum rather than continuing with the following data point.
2. *deleteTooClose*: If set, any segmentation point x will be discarded if the distance between $x{-}1$ and x is less than 250 ms.[8]
3. *deleteLowVariance*: If set, any segmentation point x will be discarded if the variance of the data points in the segment spanning from $x{-}1$ to x is less than 0.002.[9]
4. *imu*: Determines which IMU (accelerometer or gyroscope) the data sets are to be segmented by.
5. *segmentationKind*: Determines whether to segment by local minima, local maxima, or both.
6. *windowSize*: Determines the size of the fixed-size sliding window (in data points).
7. *filter*: Determines whether to segment based on the moving-average-filtered or Butterworth-filtered Euclidean norm of the IMU.

Algorithm 3.1 shows a pseudo code representation of the segmentation algorithm devised.[10]

The pseudo code algorithm has four parameters. The first parameter, *data*, is the normed and filtered data in which local extrema serving as segmentation points are found (i.e., the filtered and normed accelerometer or gyroscope signal). *data* is assumed to be a list with 0-based indexing. The second parameter, *windowSize*, is the size of the sliding

[8] An implementation deleting too close segmentation points is available at `https://github.com/CppPhil/mogasens_csv/blob/master/python/preprocessed_segment.py` (Accessed 2021-02-16) in lines 73 to 86.

[9] An implementation deleting segments with too little variance is available at `https://github.com/CppPhil/mogasens_csv/blob/master/python/preprocessed_segment.py` (Accessed 2021-02-16) in lines 89 to 98.

[10] An exemplary implementation written in Python [36] is available at `https://github.com/CppPhil/mogasens_csv/blob/master/python/modules/segmentation_points.py` (Accessed 2021-01-25)

window (in data points). The *windowSize* must always be an odd number greater than or equal to 3. The third parameter, *skipWindow*, is a Boolean flag that is true if the remainder of the current sliding window's elements should not be examined given that the current data point is a local extremum, and false if all data points should serve as potential segmentation points, irrespective of their proximity to another segmentation point. The fourth parameter, *segmentationKind*, indicates whether the algorithm should detect local minima, local maxima, or both. It can be thought of as a bitflag type. The *deleteTooClose* and *deleteLowVariance* Boolean parameters are not part of the core algorithm and are applied as filters to the result of the pseudo code algorithm.

The output of the pseudo code algorithm is a list containing the indices into *data* where the values signify local extrema.

In Line 1, the radius is calculated by subtracting 1 from the *windowSize* and then halving the result. The radius is the number of neighboring data points to consider (in either direction) when determining whether the current data point is a local extremum. If the *windowSize* is 51, for example, the radius is 25. Hence, to be considered a local maximum, x must be greater than all 25 data points preceding it and all 25 data points following it.

In Line 2, *extremumIndices* is initialized with the empty list. The indices of the local extrema are collected in this list.

In Line 3, *currentIndex* is initialized to 0. This variable is the index of the current data point, that is classified as a local minimum, local maximum, or neither.

The main loop of the algorithm starts in Line 4 and ends in Line 42. *dataPointCount* is the number of data points in *data* (i.e., the length of *data* in elements).

In Lines 5 and 6, *windowBegin* and *windowEnd* are set. *windowBegin* indicates the index with which the current sliding window begins, and *windowEnd* indicates the index with which it ends. These bounds are to be interpreted as a closed range (i.e., both the elements at *windowBegin* and *windowEnd* are part of the window).

The Boolean variable *wasAdded* is defined as false in Line 7. This variable indicates whether *currentIndex* was appended to *extremumIndices*.

In Lines 8 to 13, the *windowBegin* and *windowEnd* variables are clamped so that they remain in bounds. This results in a smaller sliding window at the beginning and the end of the data, as there are insufficient data points preceding or following the current data point to fill the window entirely.

If *segmentationKind* indicates that the algorithm should look for local minima, Lines 14 to 25 test whether the data point at *currentIndex* is a local minimum. By default, the current data point is assumed to be a local minimum (Line 15). If any of the other elements are smaller than the current data point (Lines 16 to 20), *isLocalMinimum* is set to false (Line 18). Note that the loop can be broken out of when *isLocalMinimum* is set to false. In Lines 21 to 25, *currentIndex* is appended to *extremumIndices* if it is a local minimum. Note that *wasAdded* is set to true in Line 23 when *currentIndex* is appended to *extremumIndices*.

If *segmentationKind* indicates that the algorithm should look for local maxima, Lines 26 to 36 check whether the current data point is a local maximum. Note that the algorithm only tests whether the data point at *currentIndex* is a local maximum if *wasAdded* is false (i.e., it was not already determined to be a local minimum). This avoids redundant comparison of data points. The current data point is considered not to be a local maximum if any of the other data points in the sliding window are greater than the current data point (Lines 28 to 32). Note that this loop can also be broken out of as soon as the *isLocalMaximum* variable is set to false.

Lines 37 to 41 either increment *currentIndex* by one or, if *skipWindow* is true and the *currentIndex* has been appended to *extremumIndices*, move *currentIndex* just beyond the current sliding window.

The algorithm terminates after having iterated through all the elements in *data*, returning the resulting list of indices of local extrema in Line 43.

Algorithm 3.1 The segmentation algorithm as pseudo code.

Input: data, windowSize, skipWindow, segmentationKind

Output: *extremumIndices* (list of indices into the input data that signify local extrema)

```
1: $radius \leftarrow (windowSize - 1)/2$
2: $extremumIndices \leftarrow emptyList$
3: $currentIndex \leftarrow 0$
4: while $currentIndex < dataPointCount$ do
5:     $windowBegin \leftarrow currentIndex - radius$
6:     $windowEnd \leftarrow currentIndex + radius$
```

```
 7:     $wasAdded \leftarrow false$
 8:     **if** $windowBegin < 0$ **then**
 9:         $windowBegin \leftarrow 0$
10:     **end if**
11:     **if** $windowEnd \geq dataPointCount$ **then**
12:         $windowEnd \leftarrow dataPointCount - 1$
13:     **end if**
14:     **if** segmentationKind includes LOCAL_MINIMA **then**
15:         $isLocalMinimum \leftarrow true$
16:         **for** $i \leftarrow windowBegin$ to $windowEnd$ **do**
17:             **if** $i \neq currentIndex$ AND $data[i] < data[currentIndex]$ **then**
18:                 $isLocalMinimum \leftarrow false$
19:             **end if**
20:         **end for**
21:         **if** $isLocalMinimum$ **then**
22:             append currentIndex to extremumIndices
23:             $wasAdded \leftarrow true$
24:         **end if**
25:     **end if**
26:     **if** segmentationKind includes LOCAL_MAXIMA AND NOT wasAdded **then**
27:         $isLocalMaximum \leftarrow true$
28:         **for** $i \leftarrow windowBegin$ to $windowEnd$ **do**
29:             **if** $i \neq currentIndex$ AND $data[i] > data[currentIndex]$ **then**
30:                 $isLocalMaximum \leftarrow false$
31:             **end if**
32:         **end for**
33:         **if** $isLocalMaximum$ **then**
34:             append currentIndex to extremumIndices
35:         **end if**
36:     **end if**
37:     **if** $skipWindow$ AND $wasAdded$ **then**
38:         $currentIndex \leftarrow windowEnd + 1$
39:     **else**
40:         $currentIndex \leftarrow currentIndex + 1$
41:     **end if**
42: **end while**
```

```
43: return extremumIndices
```

Before segmentation begins, the data points chronologically preceding and following the exercise are discarded.[11] The timestamps marking the start and the end of the exercise timeframe were determined by manually inspecting the recorded video data.

Figure 3.6 shows part of a segmentation of a set of squat exercises. The magenta-colored vertical lines indicate segmentation points. The moving-average-filtered Euclidean norm of the accelerometer signals is plotted in red, and the Butterworth-filtered one is plotted in blue.

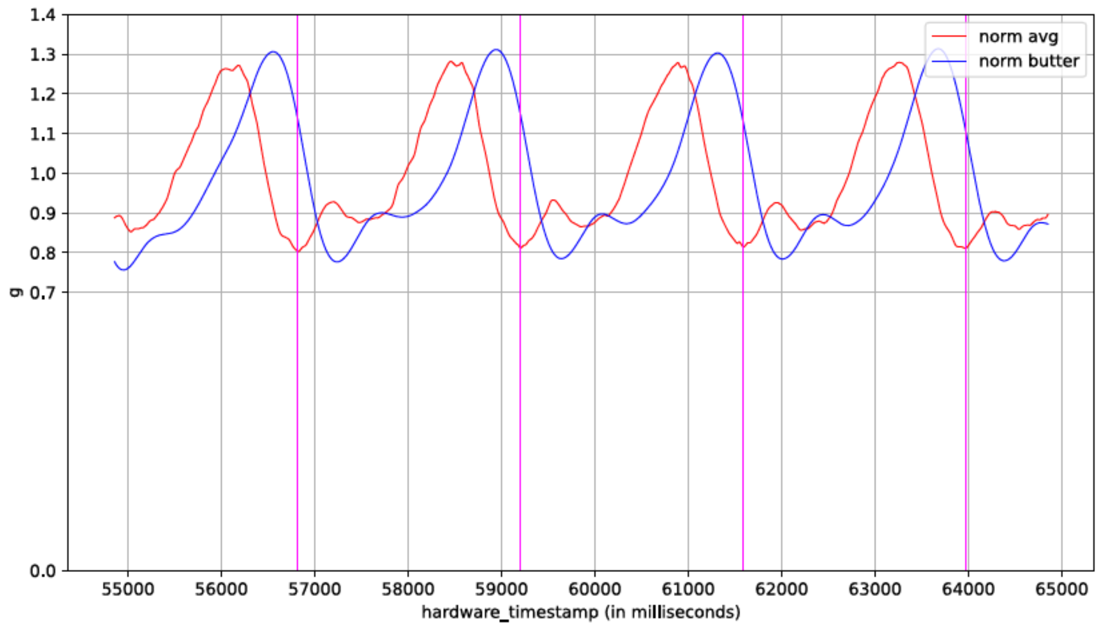


Figure 3.6: Test person 5, squats set 2 abdomen sensor segmentation.

The parameters of the algorithm were set as follows:

- *skipWindow*: false
- *deleteTooClose*: false
- *deleteLowVariance*: true

[11]These crop points are available at https://github.com/CppPhil/mogasens_csv/blob/master/python/preprocessed_segment.py (Accessed 2021-02-16) in lines 206 to 260.

- *segmentationKind*: local minima
- *windowSize*: 401
- *filter*: moving average

Figure 3.7 shows part of a segmentation of a set of push-up exercises.

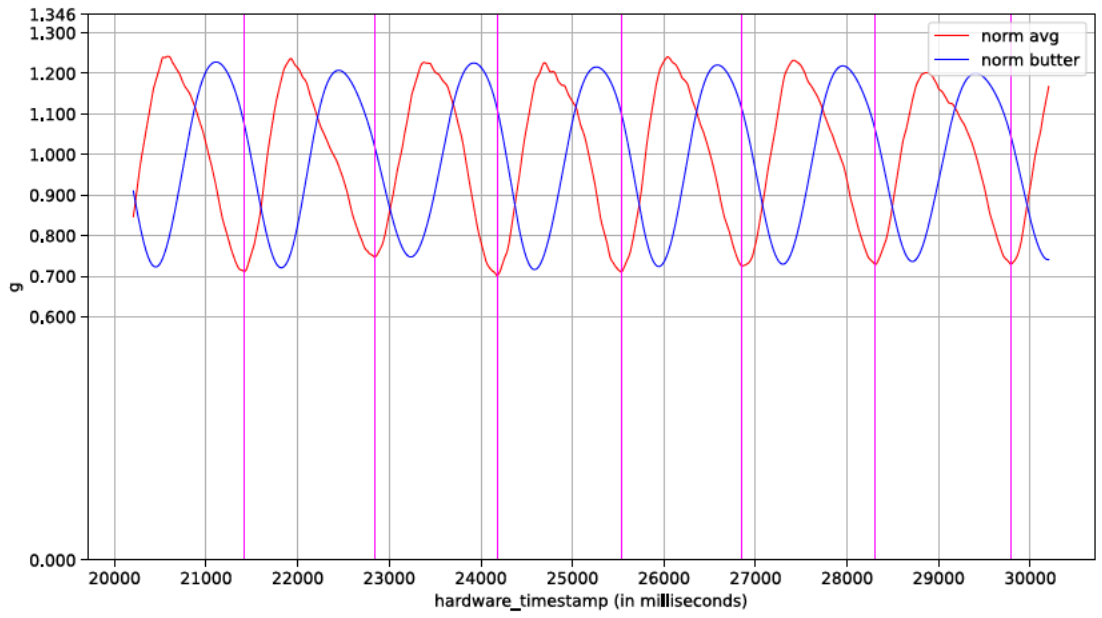


Figure 3.7: Test person 2, set 1 abdomen sensor push-ups segmentation.

The parameters of the algorithm were set as follows:

- *skipWindow*: false
- *deleteTooClose*: false
- *deleteLowVariance*: false
- *segmentationKind*: local minima
- *windowSize*: 451
- *filter*: moving average

The proposed system provides the preprocessing required for subsequent segmentation and implements a segmentation algorithm that has a time complexity of $n * m$ where n is the number of data points in which the algorithm should look for segmentation points and m is the size of the sliding window in data points, which is a runtime constant. Additionally, the memory usage grows linearly in n because an index is appended to the resulting list whenever a local extremum is detected.

4 Evaluation

This chapter evaluates the solution proposed in Chapter 3. First, it gives an overview of the number of exercise repetitions counted in each data set. Second, it presents and evaluates the results, focusing on the quality of segmentation, which is evaluated by both the count of segmentation points and a confusion matrix. The chapter ends with a discussion of the results.

The repetitions in each data set were counted using the video data (i.e., ground truth) to evaluate whether the implementation can algorithmically determine the correct number of segmentation points. Table 4.1 shows the number of repetitions of the push-up exercise performed by each test person, as counted by several researchers by inspecting the video recordings.

Data set	Repetitions (author)	Repetitions (University of Hamburg)	Repetitions (Jan Schwarzer)
Test person 1, set 1	24	24	24
Test person 1, set 2	20	20	20
Test person 1, set 3	15	14	14
Test person 2, set 1	26	25	25
Test person 2, set 2	22	21	21
Test person 2, set 3	18	17	17
Test person 3, set 1	10	10	10
Test person 3, set 2	16	16	16
Test person 3, set 3	18	17	17
Test person 4, set 1	25	25	25
Test person 4, set 2	19	19	19
Test person 4, set 3	13	13	13
Test person 5, set 1	27	27	27
Test person 5, set 2	20	20	20
Test person 5, set 3	17	17	17
Test person 6, set 1	24	24	24
Test person 6, set 2	19	18	18
Test person 6, set 3	11	11	10

Table 4.1: An overview of the repetitions as counted by several researchers.

The third column shows the repetitions counted by sports scientists from the University of Hamburg.[1] The fourth column shows the repetitions counted by Jan Schwarzer. There are small discrepancies of one repetition between the counts, which can be explained by the difficulty in determining whether the last repetition should be counted as such. The test persons were instructed to perform push-ups until exhaustion, causing the final repetition to be of questionable quality. The following evaluation was based on the repetitions counted by the author.

The algorithm used for segmentation is described in section 3.3.2. A moving average filter with a sample count of 125 or a 3rd order Butterworth filter was applied to the data before they were segmented. To find a suitable configuration (a set of parameter values) of the segmentation algorithm, all possible combinations of parameter values were tested with the data sets using window sizes of 51, 101, 151, 201, 251, 301, 351, 401, 451, 501, and 551. The counts of the algorithmically determined segmentation points in the data sets were written to separate log files.[2] These log files were subsequently analyzed to find the configuration, that produced the closest number of segmentation points to the expected number.[3]

4.1 Results

4.1.1 Euclidean Norm

Initially, only the z-axis was considered for the segmentation of the push-up exercise, because the movement is primarily downward and upward; only small sideward or forward and backward movements were expected. Use of the Euclidean norm was introduced to base the segmentation on the axis of effect [21] without having to find that axis. Figure 4.1 shows an example in which the x-axis, not the z-axis, is the most relevant axis.

[1] `https://uni-hamburg.de` (Accessed 2021-01-19)

[2] The implementation creating these log files is available at `https://github.com/CppPhil/mogasens_csv/blob/master/ruby/segment_all.rb` (Accessed 2021-02-16). The log files are available at `https://github.com/CppPhil/mogasens_csv/tree/master/segmentation_comparison/logs` (Accessed 2021-02-16).

[3] An implementation that analyzes these log files is available at `https://github.com/CppPhil/mogasens_csv/blob/master/compare_segmentation/src/main.cpp` (Accessed 2021-02-16).

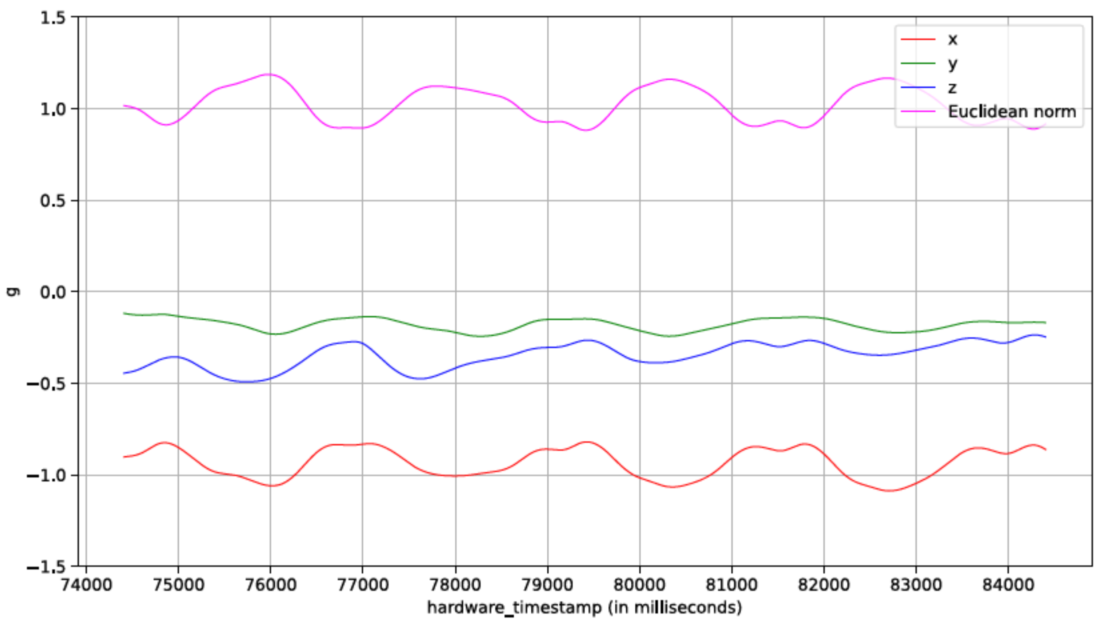


Figure 4.1: Test person 3, set 1 chest sensor filtered accelerometer signal: x-axis as the axis of effect.

If the segmentation had been based on the z-axis (instead of the Euclidean norm of all three axes), the performance of the segmentation would have been much poorer overall.

4.1.2 Sensor Fusion

The chest sensor and the abdomen sensor each proved sufficient for segmentation of both push-ups and squats. In the Appendix, table A.2 shows how many segmentation points were algorithmically determined using the algorithm configuration that yielded the optimal results. One can observe that the chest sensor and the abdomen sensor yielded the results closest to the expected number in general. To generalize to other forms of exercise, one could fuse the four sensors into one artificial signal, as the three axes of an IMU are normed together.

4.1.3 Segmentation

Table 4.2 shows the optimal configurations of the segmentation algorithm for the squats, and the push-ups sampled at 250 Hz and for all data sets combined.

Parameter	Squats (sampled with 200 Hz)	Push-ups (sampled at 250 Hz)	All data sets
skipWindow	false	false	false
deleteTooClose	false	false	false
deleteLowVariance	true	false	false
segmentationKind	local minima	local minima	local maxima
windowSize	401	451	451
filter	moving average	moving average	moving average
Distance score	11	42	108

Table 4.2: Configurations yielding the lowest distance scores.

The distance score is the sum of the distances from the manually counted repetitions to the algorithmically determined segmentation points in each data set. Lower distance scores are preferable because they indicate less deviation from the ground truth. If the segmentation quality remains the same over all the repetitions in all the data sets, then one would expect the distance score to grow linearly with the count of repetitions, because the likelihood of error remains constant with each repetition equally contributing to the distance score.

Table 4.3 shows the optimal configurations using the Butterworth filter, which is the second-most optimal filter setting for the data sets. By comparing the distance scores in table 4.3 with the ones in table 4.2 one can observe to what degree the segmentation is affected by the choice of filter.

Parameter	Squats (sampled at 200 Hz)	Push-ups (sampled at 250 Hz)	All data sets
skipWindow	false	false	false
deleteTooClose	false	false	false
deleteLowVariance	true	false	false
segmentationKind	local minima	local maxima	local maxima
windowSize	551	401	501
filter	Butterworth	Butterworth	Butterworth
Distance score	15	45	120

Table 4.3: Optimal configurations using the Butterworth filter.

Table 4.2 shows that the configurations using the Butterworth filter have slightly higher (less optimal) distance scores. This indicates that the suitability of both filtering approaches for subsequent segmentation is similar because the distance scores are marginally higher.

Table 4.4 compares the optimal moving average filter configuration for the squats with the optimal one using the Butterworth filter.

Data set	Sensor	Repetitions (ground truth)	Moving average filter segmentation points	Butterworth filter segmentation points
Squats 1	abdomen	30	29	32
Squats 1	chest	30	33	30
Squats 1	left arm	30	31	33
Squats 1	right arm	30	29	31
Squats 2	abdomen	49	49	48
Squats 2	chest	49	53	51
Squats 2	left arm	49	50	52
Squats 2	right arm	49	49	46

Table 4.4: A comparison of the optimal configurations for both filtering approaches for the squats exercise.

The repetitions column shows the manually counted repetitions, which serve as the expected number of segmentation points (i.e., the ground truth). The moving average filter segmentation points column shows the count of segmentation points that were algorithmically determined based on data filtered using a moving average filter. A filter sample count of 125 was used for the moving average filter to match the expected duration of an exercise (see section 3.3.2). The Butterworth filter segmentation points column shows the count of segmentation points that were algorithmically determined based on data filtered using a 3$^{\text{rd}}$ order Butterworth filter. The segmentation quality is quite similar. This is also the case for the push-ups sampled at 250 Hz (see table A.1) and for all data sets (see table A.2).

To evaluate the segmentation algorithm, the data were manually segmented by inspecting the video recordings (i.e., the ground truth). The manual segmentation points were created by inspecting the video recordings frame by frame.[4] A frame was deemed a segmentation point if it was within the upward or the downward movement of a push-up and the visual change observed since the previous frame was the greatest among the frames the movement consists of. During manual segmentation, no attempt was made to differentiate between local minima or local maxima. The videos had a frame rate of 30 fps.

Furthermore, the video recordings started earlier than the sensor recordings. This necessitated synchronization of the video recordings with the sensor recordings. The manual segmentation points derived from the video recordings were therefore first converted to

[4]See https://github.com/CppPhil/mogasens_csv/blob/master/confusion_matrix/data/manual_segmentation.csv (Accessed 2021-02-16) for the manual segmentation points.

hardware timestamps so that they could be compared to the segmentation points determined algorithmically. To convert the manual segmentation points to hardware timestamps, the first algorithmically determined segmentation point was assumed to be the same as the first manual segmentation point. The hardware timestamps of the manual segmentation points were calculated by adding the difference between a manual segmentation point and the first manual segmentation point to the hardware timestamp of the first algorithmically determined segmentation point.[5]

Confusion matrices for the configurations of the segmentation algorithm were created using a delta of ±450 ms over the four sensors in all the data sets. A delta (i.e., difference) of ±450 ms was used to account for the inaccuracy of the manual segmentation, because determining which frames in a video constitute segmentation points is challenging. From the video recordings, a push-up was estimated to take approximately 27 frames. As the videos were recorded with 30 fps, the duration of each frame was approximately 33.33 ms. Therefore, the duration of 27 frames is 900 ms. The 900 ms were halved, as the delta of 450 ms allows for deviation in either positive or negative directions.

Configurations with window sizes of 3, 51, 101, 151, 201, 251, 301, 351, 401, 451, 501, 551, 601, 651, 701, 751, 801, 851, 901 and 951 were compared with one another. All configurations used were segmented by both local minima and local maxima, as the ground truth includes both local minima and local maxima.[6]

For every hardware timestamp in each data set, the true positives, false positives, false negatives, and true negatives were counted as follows:[7]

1. The number of true positives is incremented by 1 for each timestamp that the algorithm considers to be a segmentation point and that exists within a delta of ±450 ms in the ground truth.

[5]An implementation converting the manual segmentation points to hardware timestamps is available at https://github.com/CppPhil/mogasens_csv/blob/master/confusion_matrix/src/manual_segmentation_point.cpp (Accessed 2021-02-16) in lines 360 to 414.

[6]See https://github.com/CppPhil/mogasens_csv/tree/master/confusion_matrix/data/segmentation_points_imported_from_python (Accessed 2021-02-17) for text files containing the segmentation points that were algorithmically determined. The text files are named according to the configuration used.

[7]See https://github.com/CppPhil/mogasens_csv/blob/master/confusion_matrix/src/confusion_matrix_best_configs.cpp (Accessed 2021-02-16) lines 101 to 193 for an implementation creating the confusion matrices.

2. The number of false positives is incremented by 1 for each timestamp that the algorithm considers to be a segmentation point but that does not exist within a delta of ±450 ms in the ground truth.
3. The number of false negatives is incremented by 1 for each timestamp that is a segmentation point according to the ground truth but does not exist within a delta of ±450 ms in the list of algorithmically determined segmentation points.
4. The number of true negatives is incremented by 1 for every other timestamp.

Table 4.5 shows the optimal configuration found by calculating the confusion matrices for the configurations used.[8]

Parameter	Setting
skipWindow	false
deleteTooClose	false
deleteLowVariance	true
segmentationKind	both
windowSize	501
filter	Butterworth
imu	accelerometer

Table 4.5: Configuration with the most true positives and true negatives and the fewest false positives and false negatives.

Analogously, figure 4.2 shows the confusion matrix of the optimal configuration.

[8]The configurations are available at `https://github.com/CppPhil/mogasens_csv/blob/master/output.txt` (Accessed 2021-02-16). The “addTrueSubtractFalse” comparison method was used for this thesis, with the optimal configuration in line 12815.

		prediction outcome		
		p	**n**	**total**
actual value	**p′**	0.19%	0.02%	P′
	n′	0.04%	99.75%	N′
	total	P	N	

Figure 4.2: Confusion matrix for the optimal overall configuration.

The *p* column shows the percentages of timestamps that the algorithm considered to be segmentation points, and the *n* column shows the percentages of timestamps that the algorithm considered not to be segmentation points. The *p’* row shows the percentages of timestamps that are segmentation points according to the ground truth, and the *n’* row shows the percentages of timestamps that are not segmentation points according to the ground truth. The upper left quadrant shows the percentage of timestamps that are true positives (0.18% i.e., they are segmentation points according to both the ground truth and the algorithm). The upper right quadrant shows the percentage of timestamps that are false negatives (0.02%; segmentation points according to the ground truth but not the algorithm). The lower left quadrant shows the percentage of timestamps that are false positives (0.04%; segmentation points according to the algorithm but not the ground truth). Finally, the lower right quadrant shows the percentage of timestamps that are true negatives (99.75%; not segmentation points according to both the ground truth and the algorithm). There are 0.06% errors in total (the sum of the false negatives and false positives) and 99.79% correctly classified timestamps (the sum of the true negatives and true positives). Most timestamps (99.75%) were true negatives as they were not segmentation points (i.e., local extrema).

The optimal configuration was determined by comparing the confusion matrices of the configurations as follows:[9]

[9]An implementation of the comparator used is available at https://github.com/CppPhil/mogasens_csv/blob/master/confusion_matrix/include/confusion_matrix_best_configs.hpp (Accessed 2021-02-16) in lines 125 to 148.

1. The number of positive points was calculated by adding the number of true positives and true negatives.
2. The number of negative points was calculated by adding the number of false negatives and false positives.
3. If the number of negative points was greater than the number of positive points the configuration was awarded 0 points.
4. Otherwise, the configuration was awarded the number of positive points minus the number of negative points.
5. The configuration awarded the most points was deemed optimal.

4.2 Discussion

This section begins by discussing the parameters of the configuration. The *windowSize* parameter has the largest impact on the distance score. The *skipWindow* and *deleteTooClose* parameters have no impact for larger window sizes; they only make a difference for smaller window sizes such as 51.[10] Notably, both the moving average filter and the Butterworth filter allowed for satisfactory segmentation of the input signal, given a suitable window size for the segmentation algorithm. Initially, it was expected that the use of the Butterworth filter would lead to significantly more accurate segmentation, because the smoothing effect observed is more pronounced, but the moving average filter proved to be similarly suitable for the task.

A larger sliding window results in less over-segmentation. However, the sliding window should not be too large; otherwise, multiple repetitions would be considered part of the same segment. Hsu et al. [15] considered seven push-ups in 10 s to be fast and four push-ups in 10 s to be slow. Hence, a push-up should take about 1.40 s to 2.50 s.

If a sampling rate of 250 Hz is used, a new data point for each sensor channel arrives every 4 ms. Using a sliding window size of 501 elements therefore means that the distance between two segmentation points is at least 1 s. If the sampling rate is 200 Hz,

[10]See https://github.com/CppPhil/mogasens_csv/blob/master/segmentation_comparison/out.txt (Accessed 2021-02-16) starting in line 534. One can observe that most configurations yield the same distance score when only the *skipWindow*, or *deleteTooClose* settings differ between them.

the minimum distance between segmentation points will be 1.25 s for the same window size. With a window size of 501, no other segmentation point may exist in the interval $[x - 250, x + 250]$ for a given segmentation point x. The window size must be sufficiently small to allow for accurate segmentation when the repetitions are fast.

A large window size increases the delay in online use, because more data must be buffered before they can be segmented. The increased delay may be undesirable, and hence a smaller window size should be used to minimize the delay at the expense of segmentation accuracy. For real world use, the window size should be smaller than 501 data points so that the delay is acceptably short.

The videos recorded to serve as the ground truth had frame rates of 30 fps. This resulted in low accuracy of the manual segmentation points, because 30 fps (i.e., 30 images every second) is significantly lower than the 250 Hz or 200 Hz (i.e., 250 or 200 data points per second) that were used to sample the signals. Additionally, precisely determining the point of maximum acceleration by visual inspection proved to be challenging.

Overall, the author expects that the solution presented here will be suitable for implementation in the smart shirt's microprocessor due to its low runtime complexity and memory consumption. The algorithm can be used for online segmentation purposes. Other approaches such as the Viterbi algorithm were not considered because of the runtime requirements [23]. Modified versions of the Viterbi algorithm suitable for online use exist, but they require the model to be pre-trained [23]. Such an approach was not implemented because writing a correct online Viterbi algorithm was more challenging than implementing the proposed algorithm. A dynamic time warping algorithm, such as the one used by Ishii et al. [18], is assumed to mitigate over-segmentation and under-segmentation using a fixed-size window. Such an approach was not used due to the runtime complexity.

The solution introduced here has been tested with the push-up and squat exercises. Whether it can be generalized to other exercises remains to be seen. Due to the nature of the algorithm (i.e., the detection of local extrema), the exercise must result in repeating sensor patterns that consistently contain such extrema that can be used for segmentation. Different exercises may require recalibration of the parameters for optimal segmentation quality, especially if the time taken for a single repetition is much different from that of other exercises, because the algorithm is highly sensitive to the *windowSize* parameter.

Additionally, for some exercises, it may be beneficial to regard a time span of more than the distance from one segmentation point to the next as a single segment. For instance, in the case of sit-ups, Ishii et al. [18] detected three peaks for each motion. In such a case, the sit-up exercise spans three peaks (i.e., segmentation points) rather than two. Notably, the authors do not consider the entire timespan of all three peaks, but rather take into account only the first segment of a sit-up motion.

Using the optimal configuration for a variety of exercises is assumed to yield less accurate results than using different (optimal) configurations for each kind of exercise. For instance, the optimal configuration for push-ups sampled at 200 Hz achieves a lower (more optimal) distance score for push-ups sampled at 200 Hz than the optimal overall (i.e., for all data sets) configuration does. Table 4.6 shows the optimal configuration for push-ups sampled at 200 Hz and the optimal overall configuration. Both configurations were applied to push-ups sampled at 200 Hz. The optimal overall configuration yields a higher (i.e., less optimal) distance score.

Parameter	Optimal configuration for push-ups sampled at 200 Hz	Optimal overall configuration
skipWindow	false	false
deleteTooClose	false	false
deleteLowVariance	true	false
segmentationKind	local maxima	local maxima
windowSize	451	451
filter	Butterworth	moving average
Distance score	19	23

Table 4.6: Comparison of the optimal configuration for the push-ups sampled at 200 Hz and the optimal overall configuration.

Likewise, table 4.7 shows that the optimal configuration for squats achieves a more accurate segmentation for squats than the optimal overall configuration does.

Parameter	Optimal configuration for squats	Optimal overall configuration
skipWindow	false	false
deleteTooClose	false	false
deleteLowVariance	true	false
segmentationKind	local minima	local maxima
windowSize	401	451
filter	moving average	moving average
Distance score	11	33

Table 4.7: Comparison of the optimal configuration for the squats and the optimal overall configuration.

If the set of possible exercises is finite and known in advance, these optimal configurations could, given enough data, be predetermined. Lastly, pre-processing the data with different filters may affect the segmentation (e.g., lead to over-segmentation).

5 Conclusion and Future Work

This work proposed an implementation of a segmentation algorithm for accelerometer signals and demonstrated its suitability through a series of tests. Most of the parameters of the algorithm have little impact on the segmentation quality. The quality is primarily influenced by the *windowSize* parameter. Additionally, both the moving average and the Butterworth filters are capable of suitable signal-smoothing for subsequent segmentation.

The algorithm has several limitations, however, including the issue of over-segmentation, which is common in segmentation algorithms based on local extrema or zero-crossings. A larger sliding window reduces the degree of over-segmentation but introduces longer delays, as more data points must be buffered before a segment is passed to the feature extraction step. This would cause the user to receive delayed feedback regarding their exercise performance. Rapid feedback is important not only so that the user experience is satisfactory, but also so that athletes can quickly correct their exercise form to achieve more effective training results or abort the exercise to avoid injury, especially when performing weight-training exercises.

Furthermore, it is unknown how well the algorithm can be generalized to other exercises or to segmentation of entirely different signals. Resolving these issues will require further research. The results may be useful to researchers facing similar problems with the segmentation of repetitive time series data. To use the proposed algorithm to solve another problem, that problem must exhibit some repetitive pattern (e.g., in sensor data) that can be segmented into individual components (segments) by detection of local extrema in the data. One such field for which the algorithm may be useful is anomaly detection. The algorithm may be used to segment any signal data that can be segmented by local extrema and for which the individual segments must match some pattern or exhibit some measurable quality. These patterns or qualities may be extracted from the segments determined by the algorithm.

This thesis concerns the segmentation for accelerometer signals of sports exercises, but the approach may be generalized to rehabilitation exercises or other medical uses, which may lead to a possibility of devising a system to aid medical diagnoses. Such avenues of research are left for future work.

This thesis has laid the foundation for a data analysis pipeline for push-up and squat exercises based on accelerometer signals, leading to the subsequent stages of the ARC: feature extraction and classification. These stages were prototypically implemented by a team of researchers at the Creative Space for Technical Innovations. Their implementation enables segments of the push-up data sets to be determined as correct or incorrect using a classifier that is trained using some of the data sets recorded and the labeling of the repetitions as provided by the University of Hamburg.

This thesis only considers push-ups and squats. New exercises might require adjustments to the segmentation and other parts of the pipeline. It is assumed that generalization to other problems is possible, given that the data produced consist of individual components (e.g., repetitions) and so segmentation by local extrema results in useful segments for the subsequent steps in the analysis pipeline. This thesis has shown the feasibility of segmentation of push-up exercises and squat exercises by detecting the local extrema in the respective accelerometer signals. If the proposed algorithm can be used for other problems, then it solves the problem of the segmentation of repetitive time series data holistically, rather than only for the specific exercises examined here.

Merging the four sensors into one artificial sensor may be necessary to generalize the algorithm to other exercises that require more than one sensor for subsequent analysis.

Determining the ground truth segmentation points from the video recordings and correlating them with the algorithmically determined ones was challenging, because the timestamps were not synchronized. Hence, improving the experimental setup may be a worthwhile pursuit in future work. Specifically, one could ensure that the sensor recording and video recording start at the same time (e.g., by sending a start signal to the sensor hardware, which then responds with the hardware timestamp). This way, the hardware timestamp that corresponds to the start of the video recording is the same as the one that corresponds to start of the sensor recording. Additionally, providing a more accurate (i.e., higher resolution) source for the ground truth may be considered.

The ability to easily switch between implementations of the steps of the ARC is desirable for scientific experimentation. For instance, one could change the filter used on the data, the segmentation algorithm, and the machine learning classifier to find a suitable solution to a given data science problem.

Bibliography

[1] Sebastian Baumbach and Andreas Dengel. Measuring the performance of push-ups - qualitative sport activity recognition. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART*, pages 374–381. INSTICC, SciTePress, February 2017. ISBN 978-989-758-220-2. doi:10.5220/0006114503740381.

[2] Antonio Bevilacqua, Bingquan Huang, Rob Argent, Brian Caulfield, and Tahar Kechadi. Automatic Classification of Knee Rehabilitation Exercises Using a Single Inertial Sensor: a Case Study. In *Proceedings of the 15th International Conference on Wearable and Implantable Body Sensor Networks*, pages 21–24, Las Vegas, Nevada, USA, March 2018. IEEE. ISSN 2376-8894. ISBN 978-1-5386-1109-8. doi:10.1109/BSN.2018.8329649.

[3] Bosch Sensortec GmbH. BMI160, 2018. URL `https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi160.html`, (Accessed 2020-07-19).

[4] *BMI160 – Data sheet.* Bosch Sensortec GmbH, November 2020. URL `https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi160-ds000.pdf`, (Accessed 2020-12-28). Rev. 1.0.

[5] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.*, 46(3), January 2014, New York, NY, USA, Association for Computing Machinery. ISSN 0360-0300. doi:10.1145/2499621.

[6] Eun Kyoung Choe, Nicole B. Lee, Bongshin Lee, Wanda Pratt, and Julie A. Kientz. Understanding quantified-selfers' practices in collecting and exploring personal data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*,

CHI '14, page 1143–1152, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450324731. doi:10.1145/2556288.2557372.

[7] Christian R. G. Dreher, Nicklas Kulp, Christian Mandery, Mirko Wächter, and Tamim Asfour. A framework for evaluating motion segmentation algorithms. *CoRR*, abs/1810.00357, October 2018. doi:10.1109/HUMANOIDS.2017.8239541.

[8] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37, March 1996. doi:10.1609/aimag.v17i3.1230.

[9] Fitbit, Inc. 3 New Features Make Exercising with Fitbit Better than Ever, 2015. URL `https://blog.fitbit.com/3-new-features-make-exercising-with-fitbit-better-than-ever`, (Accessed 2020-10-20).

[10] Fitbit, Inc. Fitbit Sense™ Specs, 2020. URL `https://www.fitbit.com/global/us/products/smartwatches/sense`, (Accessed 2020-10-20).

[11] S. Gharghabi, Y. Ding, C. M. Yeh, K. Kamgar, L. Ulanova, and E. Keogh. Matrix profile viii: Domain agnostic online semantic segmentation at superhuman performance levels. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 117–126, New Orleans, LA, USA, November 2017. Institute of Electrical and Electronics Engineers. ISSN 2374-8486. ISBN 978-1-5386-3835-4. doi:10.1109/ICDM.2017.21.

[12] Oonagh M Giggins, Kevin T Sweeney, and Brian Caulfield. Rehabilitation exercise assessment using inertial sensors: a cross-sectional analytical study. *Journal of NeuroEngineering and Rehabilitation*, 11(158), November 2014. doi:10.1186/1743-0003-11-158.

[13] Google, LLC. Google Fit: Health and Activity Tracking, 2020. URL `https://play.google.com/store/apps/details?id=com.google.android.apps.fitness&hl=en&gl=US`, (Accessed 2020-12-09).

[14] Xiaonan Guo, Jian Liu, and Yingying Chen. When your wearables become your fitness mate. *Smart Health*, 16(100114), May 2020. ISSN 2352-6483. doi:10.1016/j.smhl.2020.100114.

[15] Hsiu-Hao Hsu, You-Li Chou, Yen-Po Huang, Ming-Jer Huang, Shu-Zon Lou, and Paul Pei-Hsi Chou. Effect of push-up speed on upper extremity training until fatigue. *Journal of Medical and Biological Engineering*, 31(4):289–293, 2011.

[16] Bingquan Huang, Oonagh Giggins, Tahar Kechadi, and Brian Caulfield. The limb movement analysis of rehabilitation exercises using wearable inertial sensors. In *Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4686–4689, Orlando, Florida, USA, August 2016. IEEE. doi:10.1109/EMBC.2016.7591773.

[17] Shima Imani, Frank Madrid, Wei Ding, S. Crouter, and Eamonn J. Keogh. Introducing time series snippets: a new primitive for summarizing long time series. *Data Mining and Knowledge Discovery*, pages 1–31, November 2020. doi:10.1007/s10618-020-00702-y.

[18] Shun Ishii, Kizito Nkurikiyeyezu, Anna Yokokubo, and Guillaume Lopez. ExerSense: Real-Tme Physical Exercise Segmentation, Classification, and Counting Algorithm Using an IMU Sensor. *Sensors*, April 2020, MDPI AG. doi:10.3390/s21010091.

[19] ISO/IEC 14496-14:2020. Information technology – Coding of audio-visual objects – Part 14: MP4 file format. Standard, International Organization for Standardization, Geneva, CH, January 2020.

[20] Usman Ali Khan, Iftikhar Ahmed Khan, Ahmad Din, Waqas Jadoon, Rab Nawaz Jadoon, Muhammad Amir Khan, Fiaz Gul Khan, and Abdul Nasir Khan. Towards a complete set of gym exercises detection using smartphone sensors. *Scientific Programming*, 2020, July 2020, Hindawi. doi:10.1155/2020/6471438.

[21] Yousef Kowsar, Masud Moshtaghi, Eduardo Velloso, Lars Kulik, and Christopher Leckie. Detecting unseen anomalies in weight training exercises. In *Proceedings of the 28th Australian Conference on Computer-Human Interaction*, OzCHI '16, page 517–526, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450346184. doi:10.1145/3010915.3010941.

[22] Ian Li, Anind K. Dey, and Jodi Forlizzi. Understanding my data, myself: Supporting self-reflection with ubicomp technologies. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, page 405–414, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306300. doi:10.1145/2030112.2030166.

[23] Jonathan Lin, Michelle Karg, and Dana Kulic. Movement primitive segmentation for human motion modeling: A framework for analysis. *IEEE Transactions on Human-Machine Systems*, 46:1–15, January 2016. doi:10.1109/THMS.2015.2493536.

[24] Deborah Lupton. *The Quantified Self*. Polity Press, 1st edition, 2016. ISBN 1509500596.

[25] MATLAB. *9.9.0.1462360 (R2020b)*. The MathWorks Inc., Natick, Massachusetts, USA, 2020.

[26] Robin Müller. Erkennung menschlicher Aktivitäten auf Basis von multivariaten Zeitreihendaten mit Hilfe von Deep Learning Verfahren. Bachelor's thesis, Hochschule für Angewandte Wissenschaften Hamburg, Hamburg, Germany, June 2019. URL `https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/r_mueller.pdf`, (Accessed 2021-01-21).

[27] Ana Pereira., Duarte Folgado., Ricardo Cotrim., and Inês Sousa. Physiotherapy Exercises Evaluation using a Combined Approach based on sEMG and Wearable Inertial Sensors. In *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 4: BIOSIGNALS*, pages 73–82, Prague, Czech Republic, 2019. INSTICC, SciTePress. ISBN 978-989-758-353-7. doi:10.5220/0007391300730082.

[28] K. Prazdny. Waveform segmentation and description using edge preserving smoothing. *Computer Vision, Graphics, and Image Processing*, 23(3):327–333, September 1983. ISSN 0734-189X. doi:10.1016/0734-189X(83)90030-0.

[29] João Rodrigues, Duarte Folgado, David Belo, and Hugo Gamboa. SSTS: A syntactic tool for pattern search on time series. *Information Processing & Management*, 56 (1):61–76, January 2019. ISSN 0306-4573. doi:10.1016/j.ipm.2018.09.001.

[30] J. Sarsfield, D. Brown, N. Sherkat, C. Langensiepen, J. Lewis, M. Taheri, L. Selwood, P. Standen, and P. Logan. Segmentation of exercise repetitions enabling real-time patient analysis and feedback using a single exemplar. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(5):1004–1019, April 2019, Institute of Electrical and Electronics Engineers. ISSN 1558-0210. doi:10.1109/TNSRE.2019.2907483.

[31] M. Seiffert, F. Holstein, R. Schlosser, and J. Schiller. Next generation cooperative wearables: Generalized activity assessment computed fully distributed within a wireless body area network. *IEEE Access*, 5:16793–16807, September 2017, Institute of Electrical and Electronics Engineers. ISSN 2169-3536. doi:10.1109/ACCESS.2017.2749005.

[32] Quantified Self. Quantified Self: What is Quantified Self? URL https://quantifiedself.com/about/what-is-quantified-self, (Accessed 2020-10-19).

[33] Y. Shafranovich. Common format and mime type for comma-separated values (csv) files. RFC 4180, RFC Editor, October 2005. URL https://www.rfc-editor.org/rfc/rfc4180.txt, (Accessed 2021-01-13).

[34] Melanie Swan. The quantified self: Fundamental disruption in big data science and biological discovery. *Big Data*, 1:85–99, June 2013. doi:10.1089/big.2012.0002.

[35] P. E. Taylor, G. J. M. Almeida, T. Kanade, and J. K. Hodgins. Classifying human motion quality for knee osteoarthritis using accelerometers. In *Proceedings of the 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 339–343, Buenos Aires, Argentina, August 2010. IEEE. ISSN 1558-4615. doi:10.1109/IEMBS.2010.5627665.

[36] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, USA, 2009. ISBN 1441412697.

[37] Mark Whooley, Bernd Ploderer, and Kathleen Gray. On the integration of self-tracking data amongst quantified self members. In *Proceedings of the 28th International BCS Human Computer Interaction Conference on HCI 2014 - Sand, Sea and Sky - Holiday HCI*, BCS-HCI '14, page 151–160, Swindon, GBR, September 2014. BCS. doi:10.14236/ewic/hci2014.16.

[38] Gary Wolf. Know Thyself: Tracking Every Facet of Life, from Sleep to Mood to Pain, 24/7/365, 2009. URL https://www.wired.com/2009/06/lbnp-knowthyself, (Accessed 2020-10-19).

[39] Wei Zhang, G. Ruben, H. Regterschot, Hana Schaabova, Heribert Baldus, and Wiebren Zijlstra. Test-retest reliability of a pendant-worn sensor device in measuring chair rise performance in older persons. *Sensors*, 14(5):8705–8717, May 2014, MDPI AG. ISSN 1424-8220. doi:10.3390/s140508705.

A Appendix

A.1 Description of the Bosch BMI160 Sensor

Bosch BMI160 IMUs were used in the experiment. The Bosch BMI160 sensors contain an accelerometer and a gyroscope [3]. The accelerometer data from the sensors are sampled at a rate of 1,600 Hz, and the gyroscope data are sampled at a data rate of 6,400 Hz. The data are then filtered with a low pass filter so that the output data rate corresponds to the one configured.

The noise density is typically $180\,\mu\mathrm{g}/\sqrt{\mathrm{Hz}}$ for the accelerometer (at most $300\,\mu\mathrm{g}/\sqrt{\mathrm{Hz}}$ [4]) and $0.007\,^\circ/\mathrm{s}/\sqrt{\mathrm{Hz}}$ for the gyroscope [3]. The hardware timestamps of the Bosch BMI160 have a resolution of 39 µs [4]. Acceleration ranges of ±2 g, ±4 g, ±8 g, and ±16 g are selectable via a serial digital interface. An acceleration range of ±2 g was used for the MoGaSens project. The start-up time of the accelerometer is at most 3.80 ms. The resolution of the accelerometer signal is 16 bits. The sensitivity of the accelerometer signal is 15,729 LSB/g to 17,039 LSB/g for an acceleration range of ±2 g, 7,864 LSB/g to 8,520 LSB/g for an acceleration range of ±4 g, 3,932 LSB/g to 4,260 LSB/g for an acceleration range of ±8 g, and 1,966 LSB/g to 2,130 LSB/g for an acceleration range of ±16 g. The minimum and maximum values are $\pm 3\,\sigma$. The accelerometer output data rate is 12.5 Hz to 1,600 Hz and the output data rate accuracy is within ±1% [4].

Gyroscope ranges of $125\,^\circ\,\mathrm{s}^{-1}$, $250\,^\circ\,\mathrm{s}^{-1}$, $500\,^\circ\,\mathrm{s}^{-1}$, $1{,}000\,^\circ\,\mathrm{s}^{-1}$, and $2{,}000\,^\circ\,\mathrm{s}^{-1}$ are selectable via a serial digital interface. A range of $2{,}000\,^\circ\,\mathrm{s}^{-1}$ was used for the MoGaSens project. The start-up time of the gyroscope is typically 55 ms. The sensitivity of the gyroscope signal is 15.90 LSB/°/s to 16.90 LSB/°/s for a gyroscope range of $2{,}000\,^\circ\,\mathrm{s}^{-1}$, 31.80 LSB/°/s to 33.80 LSB/°/s for a gyroscope range of $1{,}000\,^\circ\,\mathrm{s}^{-1}$, 63.60 LSB/°/s to 67.60 LSB/°/s for a gyroscope range of $500\,^\circ\,\mathrm{s}^{-1}$, 127.20 LSB/°/s to 135.20 LSB/°/s for a gyroscope range of $250\,^\circ\,\mathrm{s}^{-1}$, and 254.50 LSB/°/s to 270.30 LSB/°/s for a gyroscope range of $125\,^\circ\,\mathrm{s}^{-1}$. The output data rate of the gyroscope is 25 Hz to 3,200 Hz and the output data rate accuracy is within ±2%. The cross-axis sensitivity (sensitivity to stimuli in non-sense-direction) is 2% [4].

The accelerometer supports three filter modes: a normal filter mode, an oversampling rate of 2, and an oversampling rate of 4 [4]. In normal mode the accelerometer data are sampled at equidistant points in time, defined by the accelerometer output data parameter. The filter bandwidth has a 3 dB cutoff frequency [4].

Likewise, the gyroscope digital filter also supports three modes: normal filter mode, an oversampling rate of 2, and an oversampling rate of 4 [4]. In normal filter mode, the gyroscope data are sampled at equidistant points in time, defined by the setting of the gyroscope output data rate parameter. The filter bandwidth, as configured by the gyroscope output data rate, has a 3 dB cutoff frequency [4].

An oversampling rate of 2 requires the output data rate to be twice that of the normal filter mode [4]. Additionally, with a given filter setting, the filter bandwidth will be half of the bandwidth in the normal filter mode (assuming the same output data rate) [4]. Likewise, an oversampling rate of 4 requires the output data rate to be four times as high as in the normal filter mode. The filter bandwidth will be a quarter of the bandwidth in the normal filter mode [4]. This holds true for both the accelerometer and the gyroscope [4].

A.2 Overview of Detected Segmentation Points and Repetition Counts for Push-up Data Sets with a Sampling Rate of 250 Hz

Data set	Sensor	Repetition count (ground truth)	Segmentation points (moving average filter)	Segmentation points (Butterworth filter)
Test person 1, set 1	abdomen	24	25	26
Test person 1, set 1	chest	24	26	26
Test person 1, set 1	left arm	24	25	26

Test person 1, set 1	right arm	24	25	25
Test person 1, set 2	abdomen	20	21	21
Test person 1, set 2	chest	20	21	21
Test person 1, set 2	left arm	20	21	21
Test person 1, set 2	right arm	20	21	21
Test person 1, set 3	abdomen	15	16	16
Test person 1, set 3	chest	15	16	17
Test person 1, set 3	left arm	15	17	18
Test person 1, set 3	right arm	15	16	17
Test person 2, set 1	abdomen	26	26	26
Test person 2, set 1	chest	26	26	26
Test person 2, set 1	left arm	26	27	26
Test person 2, set 1	right arm	26	26	26
Test person 2, set 2	abdomen	22	22	22
Test person 2, set 2	chest	22	22	22
Test person 2, set 2	left arm	22	23	22
Test person 2, set 2	right arm	22	23	22
Test person 2, set 3	abdomen	18	17	17
Test person 2, set 3	chest	18	17	17
Test person 2, set 3	left arm	18	15	16
Test person 2, set 3	right arm	18	13	17
Test person 3, set 1	abdomen	10	11	12
Test person 3, set 1	chest	10	11	12
Test person 3, set 1	left arm	10	11	12
Test person 3, set 1	right arm	10	11	12
Test person 3, set 2	abdomen	16	17	17
Test person 3, set 2	chest	16	18	18
Test person 3, set 2	left arm	16	18	18
Test person 3, set 2	right arm	16	19	18
Test person 3, set 3	abdomen	18	18	19
Test person 3, set 3	chest	18	19	20
Test person 3, set 3	left arm	18	19	20
Test person 3, set 3	right arm	18	17	17

Table A.1: Filter comparison for push-ups (sampled at 250 Hz).

A.3 Overview of Detected Segmentation Points and Repetition Counts for All Data Sets

Data set	Sensor	Repetition count (ground truth)	Segmentation points (moving average filter)	Segmentation points (Butterworth filter)
Test person 1, set 1	abdomen	24	26	26
Test person 1, set 1	chest	24	26	26
Test person 1, set 1	left arm	24	26	25
Test person 1, set 1	right arm	24	25	25
Test person 1, set 2	abdomen	20	21	21
Test person 1, set 2	chest	20	21	21
Test person 1, set 2	left arm	20	20	20
Test person 1, set 2	right arm	20	21	21
Test person 1, set 3	abdomen	15	17	16
Test person 1, set 3	chest	15	17	16
Test person 1, set 3	left arm	15	17	18
Test person 1, set 3	right arm	15	17	16
Test person 2, set 1	abdomen	26	26	26
Test person 2, set 1	chest	26	26	26
Test person 2, set 1	left arm	26	27	23
Test person 2, set 1	right arm	26	27	24
Test person 2, set 2	abdomen	22	23	22
Test person 2, set 2	chest	22	23	22
Test person 2, set 2	left arm	22	23	22
Test person 2, set 2	right arm	22	23	22
Test person 2, set 3	abdomen	18	16	16
Test person 2, set 3	chest	18	16	16
Test person 2, set 3	left arm	18	12	14
Test person 2, set 3	right arm	18	14	13
Test person 3, set 1	abdomen	10	11	12
Test person 3, set 1	chest	10	11	12

Test person 3, set 1	left arm	10	11	12
Test person 3, set 1	right arm	10	11	12
Test person 3, set 2	abdomen	16	19	17
Test person 3, set 2	chest	16	18	17
Test person 3, set 2	left arm	16	17	16
Test person 3, set 2	right arm	16	17	17
Test person 3, set 3	abdomen	18	18	17
Test person 3, set 3	chest	18	17	18
Test person 3, set 3	left arm	18	19	19
Test person 3, set 3	right arm	18	17	17
Test person 4, set 1	abdomen	25	26	18
Test person 4, set 1	chest	25	27	20
Test person 4, set 1	left arm	25	28	17
Test person 4, set 1	right arm	25	26	18
Test person 4, set 2	abdomen	19	21	19
Test person 4, set 2	chest	19	20	20
Test person 4, set 2	left arm	19	20	18
Test person 4, set 2	right arm	19	20	19
Test person 4, set 3	abdomen	13	13	13
Test person 4, set 3	chest	13	13	14
Test person 4, set 3	left arm	13	14	13
Test person 4, set 3	right arm	13	14	13
Test person 5, set 1	abdomen	27	27	27
Test person 5, set 1	chest	27	27	27
Test person 5, set 1	left arm	27	27	27
Test person 5, set 1	right arm	27	27	27
Test person 5, set 2	abdomen	20	20	20
Test person 5, set 2	chest	20	20	20
Test person 5, set 2	left arm	20	20	20
Test person 5, set 2	right arm	20	20	20
Test person 5, set 3	abdomen	17	17	18
Test person 5, set 3	chest	17	17	17
Test person 5, set 3	left arm	17	18	17
Test person 5, set 3	right arm	17	17	18

Test person 6, set 1	abdomen	24	26	25
Test person 6, set 1	chest	24	25	25
Test person 6, set 1	left arm	24	25	26
Test person 6, set 1	right arm	24	25	25
Test person 6, set 2	abdomen	19	19	19
Test person 6, set 2	chest	19	20	19
Test person 6, set 2	left arm	19	19	19
Test person 6, set 2	right arm	19	19	19
Test person 6, set 3	abdomen	11	10	8
Test person 6, set 3	chest	11	11	9
Test person 6, set 3	left arm	11	11	9
Test person 6, set 3	right arm	11	10	9
Squats 1	abdomen	30	33	33
Squats 1	chest	30	33	33
Squats 1	left arm	30	37	38
Squats 1	right arm	30	35	35
Squats 2	abdomen	49	52	52
Squats 2	chest	49	56	52
Squats 2	left arm	49	54	50
Squats 2	right arm	49	49	50

Table A.2: Filter comparison for all data sets.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen.“

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: ____________________

Vorname: ____________________

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Algorithmic activity detection based on motion and position sensors

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

		[redacted]
Ort	Datum	Unterschrift im Original