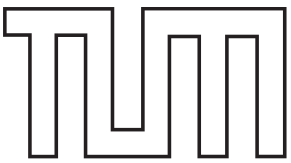


Technische Universität München

Fachgebiet
Distributed Multimodal Informationprocessing

Prof. Dr. Matthias Kranz

Studienarbeit

Sensor-Based Skill Assessment for Health and Fitness Applications

Author:	Miklós Kirilly
Matriculation Number:	[redacted]
Address:	[redacted]
Advisor:	Andreas Möller
Begin:	1. March 2011
End:	20. October 2011

Kurzfassung

Sensorgestützte Trainingsbewertung in dem Bereich Gesundheit und Fitness

Training ist gesundheitsfördernd, wenn die Übungen korrekt und regelmäßig durchgeführt werden. In dieser Arbeit untersuchte ich, welche Möglichkeiten die mit mehreren Sensoren ausgestatteten Smartphones bieten für die Verbesserung des Trainings ohne professionelle menschliche Betreuung. Ich erweiterte eine Android Anwendung um einen Algorithmus für die echtzeitfähige automatische Bewertung von Übungen auf einem Kippbrett. Die Bewegungsmuster wurden anhand des zeitlichen Verlaufs der Orientierung des Brettes untersucht. Um die Orientierung zu bestimmen, wurden Beschleunigungssensoren und magnetische Feldsensoren untersucht. Die Ergebnisse der automatischen Bewertung für die Übungen aus einer Datenbank wurden mit der Bewertung von einem Experten verglichen. Die Kreuzkorrelation in dem Fall beider Sensoren betrug über 0,51 für dynamische und 0,76 für statische Übungen. Auf einer Skala zwischen 0 und 100 war der Unterschied zwischen menschlicher und automatischer Bewertung weniger als 10 Punkte bei 77% der dynamischen und 89% der statischen Übungen.

Abstract

Exercising can contribute to a healthier and fuller life, however, correct and regular execution of the chosen exercises is of great importance. In the current project I investigated the possibilities offered by modern smartphones to enhance the effectiveness of training without a trained supervisor, through automated skill assessment. I implemented a real-time capable algorithm for the assessment of balancing board exercises, and integrated it into an Android application. I concentrated onto accelerometer and magnetometer sensors as means for measuring the orientation of the smartphone, which in turn was used for the analysis of motion patterns. Using a database of recorded exercises I compared the automated scores with scores given by a human expert. The computed score had a linear cross correlation coefficient of above 0.51 with the expert's scores for dynamic and more than 0.76 for static exercises. On a 0 to 100 scale, the absolute difference of human and automatic scores was smaller than 10 points in more than 77% of the dynamic exercises and 89% of the static exercises.

Contents

Chapter 1

Introduction

Physical and mental health is of huge importance for the quality of life, and people in the developed world increasingly pay attention to factors that influence their health. This includes diet and physical exercising as well as mental hygiene. Sports can improve fitness and lead to weight loss, and may also prevent lower back and other kinds of musculoskeletal pains [1]. Furthermore, it was shown in [2] that cognitive functions are also benefitting from exercise training. Due to these and other beneficial effects of sports the American College of Sports Medicine and the American Heart Association recommends [3] adults to regular sporting activity.

In the project exercises on balance boards are analyzed. Balance boards are boards to step on with unstable support so that the user needs to control his or her posture while standing on it. In this project the boards manufactured by Thera Band[1] were taken as reference models. They may be used in rehabilitation or post-rehabilitation conditioning after an injury, or as part of a general fitness training, and they are designed to improve balance and proprioception. They have been tested for injury prevention, however, results in football are inconclusive. [4] found no preventive effect, whereas the earlier [5] showed decreased anterior cruciate ligament injury in connection with regular balance board exercises. The review of van der Wees et. al [6] concluded that wobble board exercise therapy (along other types of exercise therapy) is likely to decrease recurring ankle sprains.

Exercises in general need to be performed correctly in order to avoid injury, in [7] correct technique was found to reduce injury rate in weight training. Checking the correctness of exercises is therefore beneficial for the progress, and it is crucial especially with beginners, who have not yet mastered the motion patterns required for the exercise. For this reason it is recommended (e.g. in [7]) to work out with an instructor or under the oversight of an expert, but this limits the flexibility of workout schedule, and some may choose to rather start on their own. These people would benefit from an automatic 'mobile instructor' that could supervise them while exercising. A smartphone with its integrated sensors offers

[1] `http://www.thera-band.com/`

sufficient capabilities to develop such an instructor application (app) for selected exercises.

As Panjan and Sarabon point out in [8], human evaluation is always subjective. An automated assessment of correctness could also avoid errors originating from exhaustion of the instructor or from his lack of knowledge or concentration.

Regularity of exercising also plays a very important role in training, regardless of whether it is part of recovery from injury or done purely to keep fit. If the exercises are not carried out according to the workout plan they might not show any effect. This requires motivation, which may be hard to maintain without training companions. A possibility to get real-time feedback from an app and to be able to examine the personal improvement may boost motivation. As Ahtinen et al. showed in [9] a well-designed mobile application can motivate the users through many factors, such as long term graphs, suggestions and coaching.

The purpose of this project was therefore to develop such a "mobile instructor" for balance board exercises. The application is designed to be used with a set of 20 exercises. Some of them include back-and-forth or side-to-side tilting of the balance board, in others the user needs to hold his or her balance. For the dynamic kind of exercises, the ideal execution means a smooth, regular sequence of ten back-and-forth or side-to side periods, at a pace of approximately one repetition per second. For the static balancing exercises, the user should stand firmly for ten seconds, while keeping the surface of the balancing board horizontal. The orientation of the balance board can be described by two rotational angles, one for the forward-backward (pitch) and one for the left-right (roll) deflection. The assessment was done by analyzing the trajectory of the orientation in the pitch-roll two dimensional space.

During my work I was able to use a database of previously recorded exercises, which included the time series of various sensor values. The sensors used were magnetometer, accelerometer, and an Android specific "virtual" sensor, the orientation sensor, the values of which were computed from the previous two sensors' samples. The execution quality of some of the exercises was also assessed by an expert based on video records. This assessment was used as a guideline for developing an adapted scoring scheme, and the results from the automated method were compared to these scores for verification. The proposed analysis method extracts various measures of the orientation angles of the board, such as the mean and the variance, or the number of times the board was tilted too far. For the dynamic exercises the individual repetitions were also recognized and individually analyzed.

I found encouraging level of correlation between the automatic and the human scores, however, there is still room for improvement.

The developed application included feedback during and immediately after the exercise and a history graph supported by a database showing how scores changed in the past. The application has been implemented for Android mobile operating system, and it was successfully tested on devices running version 2.2 and 2.3 of Android.

The rest of this report is organized into four chapters.

Chapter 2 gives an overview about current research topics and commercial applications related to my project.

In chapter 3 the algorithm for automatic assessment is described. I describe the important aspects for the choice of the orientation sensor, then discuss the importance of an appropriate sampling frequency and the reliability of the Android operating system. I then go on to describe the proposed scoring scheme and important parts of it.

Chapter 4 discusses the questions and solutions related to user interaction. After a brief overview of how the exercises are started and stopped the focus is on the various forms of feedback: instantaneous feedback during the exercise, after-session assessment and long time result history.

Chapter 5 shows the results of the proposed algorithm compared to the expert's assessment.

Chapter 2

Related Work

2.1 Stabilometry

Stabilometry, the science of measuring the balance of people, is closely related to application scenario, and it has been investigated for a long time. There are static tests, where subjects have to hold their balance in various positions and dynamic ones, where the subject walks, for instance. In Romberg's test, an often used static test, the subject stands upright with feet close together, arms by the side and eyes open; after this the subject closes. The supervising physician records signs of imbalance in both cases [10].

The tests are observed by an expert or performed on a force platform that measures the weight distribution, from which the center of pressure (COP) is determined using multiple pressure sensors. The displacement of the COP in one direction as a function of time is called the stabilogram. Various features of the stabilograms, derived either from the time function or its Fourier spectrum, can be used to assess the subject's stability.[11] Such features found in the literature are frequency, duration and mean and maximum amplitudes of oscillations [12], the parameters of the characteristic circle [11] and the path length of COP path [13]. The characteristic circle is defined in [11] as the appropriate shape of minimum area that holds 95% of the samples on the statokinesigram, the two-dimensional plot of samples of the COP. If the test provides such conditions, features describing the difference caused by opening or closing the eyes can be recorded.

Physicians diagnose patients for vestibular disorder or proprioceptive dysfunction, among others, based on insights from these features. Such assessments may also be used to avoid injuries among elderly people and athletes.[8]

Schouten et al. [14] proposed a novel platform for investigating the role of ankles in the balance of humans. While a subject stands on the platform it can be tilted under his or her feet, and an additional force platform can be used to find the center of gravity for each foot. This way the potentially different load on both feet can also be detected.

2.2 Computer Aided Skill and Health Assessment

In [15] Sabelman and his colleagues describe a system for remote mobility assessment through the application of a wearable sensor system. The main goal is to measure the "cummulative quality of motion", while real-time monitoring and fall detection was considered to be of secondary importance. They used four 3-axis MEMS acceleration sensors placed on the eyeglass frame and the hips of the patient. Two different steps are described in the paper to process the sensor data. The first step is to identify specific tasks such as stair climbing, straight walking and turning. In the second step these tasks are assessed by comparing the recordings with an age and gender dependent, experimentally determined standard.

Diemer and Chakraborty [16] also used a body area sensor network composed of triaxial accelerometer nodes, but they applied a mobile phone as central processing unit. However, since their algorithm, as published in the paper, makes use of only one sensor, it would be possible to use only the mobile phone for sensing if the user always keeps it in a pocket.[1] Their application recognized motion types such as walking, running, resting, sitting down or standing up. They used the temporal variance of the acceleration samples and examined the existence of characteristic peaks in the signals' Fourier transform.

[17] describes a system for automatic assessment of rehabilitation exercises typically prescribed for knee osteoarthritis. The movement is captured using body-worn accelerometers, then the most significant frequency components of the frequency data are fed to a classifier. A statistical classifier was used

There is also a sizeable literature describing research projects that use desktop or tablet computers to assess motoric or other skills. For instance, [18] shows a method to apply objective diagnosis of Parkinson's Disease (PD) by assessing mouse movements and clicking accuracy on a desktop computer. The paper describes a standalone testing program, in which users have to perform specific tasks.[2] Main criteria for the assessment are speed and precision of clicking the target.

Wang et al. described in [19] a method for diagnosis of PD based on free spiral drawing, whereas Hoque et al. [20] attempts to assess children's drawing skills and diagnose dyspraxia using a standard diagnosis procedure. Both approaches record the trajectory of the tip of the pen used for drawing, and analyze the movement path to make a diagnosis. Wang et al. used, among other features, the standard deviation of the drawing speed and the number of extrema in the (θ, r) space to quantify differences between PD patients and the control group. Hoque et al. used a 21-element feature vector to describe each drawing,

[1] A major objective of the study was to investigate possibilities to break down into smaller modules and distribute signal processing algorithms to minimize power consumption of the whole system. An example application was implemented in the first step with a single sensor node.

[2] It is claimed to be possible, however, to create a version that is invisible to the users and assesses their mouse movement while they are using their everyday applications. This would make the monitoring less of a burden for the users.

including measures that relate to the result as well as ones describing the way it was drawn. Their paper also contains a comparison of classification algorithms according to their capability to correctly diagnose dyspraxia. These applications are similar to the application of this project in the sense that they also make assessments based on two dimensional motion, although the complexity of the motion is quite different.

2.3 Mobile Applications for Health and Fitness

The recent rapid growth in the numbers of smartphone users and the increasing capabilities of smartphones in terms of processing power and integrated sensors, as well as their convenient user interface created a vivid market for powerful mobile applications (apps). Health and fitness is an important area of life in industrialized countries, so numerous apps are available in this domain for all smartphone operating systems. However, current exercising apps do not yet exploit the full sensory potential of smartphones.

The most popular workout apps found in the Android Market fall into one of three categories. Members of the first group are essentially collections of exercise descriptions, in some cases with video or animation demonstration how the exercises are correctly done. These apps often include complete workout sessions comprising several exercises. Some of the popular examples of this group are *Daily Ab Workout*[3], *JEFIT*[4] and *DroidFit*[5]
The second group consists of apps that focus on workout routine planning functionality. They support the user in designing their own workouts by choosing exercises and assigning repetition counts or interval times to them. These two groups have an overlap, many of the workout routine planner apps also contain a short description of the exercises, they are, however, not as elaborate as the descriptions found in the apps of the first group. *VirtuaGym*[6] and *GymLog*[7] are typical examples of this kind of apps.
The third group is the group of GPS trackers for outdoor cardio training. Some, like *GPS Cycle Computer*[8] or *RunKeeper*[9] make a difference in their primary audience, but the functionality provided by the apps for different sports is very similar. Examples of more generally oriented apps are *EndoMondo*[10] and *RunTastic*[11]. They keep track of and display on a map the user's movement using GPS positioning, record training duration, calculate average and top speed, and estimate calorie expenditure. They also often support external heart rate monitors.

[3]https://market.android.com/details?id=com.tinymission.dailyabworkoutfree1
[4]https://market.android.com/details?id=je.fit
[5]https://market.android.com/details?id=droid.fit
[6]https://market.android.com/details?id=digifit.virtuagym.client.android
[7]https://market.android.com/details?id=us.picadorproductions.gymlog
[8]https://market.android.com/details?id=com.appannex.gpscycle
[9]https://market.android.com/details?id=com.fitnesskeeper.runkeeper.pro
[10]https://market.android.com/details?id=com.endomondo.android
[11]https://market.android.com/details?id=com.runtastic.android

None of the apps try to make use of the sensors inside the phones to enhance the workout by assessing the users' skills or checking correctness of exercises, and it is not clear whether they exploit sensors to make more accurate assumptions about calorie expenditure. *EndoMondo* and *runtastic* provide ways to differentiate between running, walking or cycling, but the decision has to be made by the user, and consumed energy is calculated based on some average for the movement type. Step counter apps like *Step Counter Free*, which do make use of sensors, on the other hand, received lower ratings from users than GPS trackers due to their inaccuracy.

On the other hand, most apps provide various degrees of feedback to motivate the users. Synthesized speech is used to motivate the users in many apps[12], either just to keep exercising until the time runs out or to push the user for a better time, or to keep up the pace. *Pace Keeper* uses vibration feedback for this end. Live commentary from friends is also incorporated in current GPS tracking apps.[13] Sharing the training results with friends on a social network platform is also a regular feature. This can boost both the user's and the friends' motivation. Another motivational feature is the option to compare the current workout with previous ones. In its simplest form, it is implemented as some kind of a list of results, available for reviewing on the own device. Extensions to this include comparison with own records on a selected track or competitions with friends and others from the community.

2.4 Wii Balancing Board

Nintendo also developed a balance platform for the popular gaming console Wii. It is more similar to the force platform used in stabilometry than to the balance boards used in this project. According to [21], hospitals are already using this platform for rehabilitation, visualizing the patients' COG seems to help them learn how to balance. Patients who used Wii games recovered more rapidly after being discharged from hospital than patients who did the conventional therapy. This difference is attributed to motivational factors.

[22] presents a balancing game designed for rehabilitation at home. The authors claim that the system can adapt to the user's skills, which implies some kind of an assessment. However, it is not detailed in this paper, how this is done.

[12] e.g. VirtuaGym, adidas miCoach or RunKeeper
[13] e.g. EndoMondo, runtastic

Chapter 3

Exercise Assessment Algorithm

In order to assess an exercise the orientation of the balance board has to be established. Section 3.1 discusses the possibilities available in a smartphone, their respective benefits and the chosen solution. Section 3.2 deals with the question how often the orientation needs to be measured in order to make a proper analysis of the board's motion possible. A lower bound is established for the sampling frequency based on data from human motion related literature and the possibilities offered by the Android framework are taken under scrutiny.

In section 3.3 I explain the assessment concept, starting with the scoring system used by a human expert, then introducing the various metrics the automatic assessment uses. For the purposes of assessment the exercises are divided into two groups: static and dynamic exercises. The former group includes exercises where the user has to maintain balance in the same position of the board, whereas the latter is formed out of exercises that require the user to swing the board back and forth or from one side to another in a periodic manner. For these two groups a different set of metrics are calculated that describe how well the exercise was executed.

The first step of the analysis of dynamic exercises is to extract the individual repetitions (also termed cycles in the description) of the exercise, each of which includes one tilting of the board in one direction followed by a tilt into the opposite direction. The method used to find these cycles is described in section 3.4.

Important details related to the Android implementation, such as discrete Fourier transform implementation, partitioning and scheduling of the algorithm, are summarized in section 3.5.

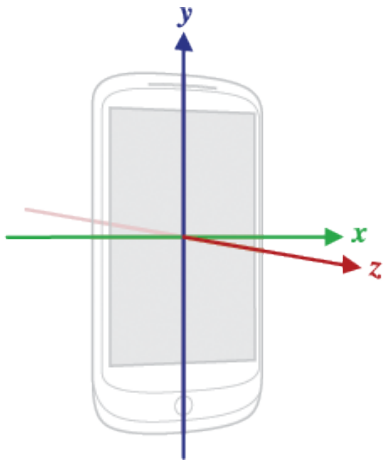


Figure 3.1: Device coordinate system axes
source: Android SDK Reference

3.1 Orientation of the Board

The board's orientation is defined by three parameters. These three parameters can be for example three rotational angles that if performed around predefined axes in a predefined order transform the board into a reference position.
Another possibility is to consider points or vectors in two coordinate systems, one aligned with the board and the other reference coordinate system fixed to the environment, and use the rotational matrix part of the Euklidean transformation that projects a point in one into the other as a descriptor of orientation.
These descriptions are equivalent and can be transformed into each other.

The board's orientation is extracted through laying a mobile telephone onto the board, parallel to the feet of the user. Different smartphones provide different sets of integrated sensors that can be used to calculate the orientation. Most current models have triaxial accelerometers, many of them also have a digital compass (triaxial magnetometers), whereas only few are fitted with a gyroscope yet. For this reason, I only consider the first two possibilities in detail.

Let us define the axes of the board- and phone aligned coordinate system as figure 3.1 shows them: x_b points to the right of the display, y_b points up along the longer edge of the display, while z_b is pointing out of the display towards the user. The placement of the phone is required to be such that y_b is parallel to the feet of the user.

An intuitive reference coordinate system can be defined so that the z_w axis points upward normal to the surface of the Earth, and y_w is tangential to the surface and points into the direction of the magnetic north. x_w is aligned with $y_w \times z_w$ to create a right-handed coordinate system. Figure 3.2 shows the coordinate system at a chosen point on Earth.

Throughout the report I will be using the terms azimuth, pitch, and roll as they are defined

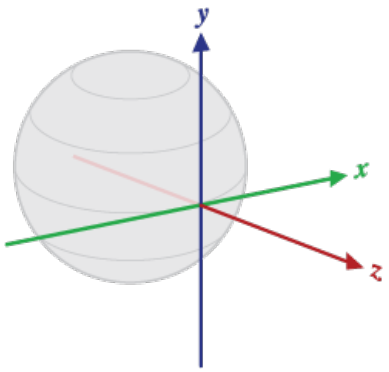


Figure 3.2: Reference coordinate system axes
source: Android SDK Reference

in the Android SDK reference [23]. Azimuth is defined as the angle between the magnetic north direction (y_w) and the y-axis of the device around the z-axis, measured clockwise from the magnetic north to y_b. Pitch is the rotation angle around x_b, measured clockwise when looking from the positive x values toward negative ones. Roll is the rotation angle around y_b, also measured clockwise. All these angles are zero if the two coordinate systems are aligned.

For the purposes of the skill assessment the azimuth value is of no interest, only the pitch and the roll angles are important.

All the sensors have to be calibrated at least once before using in the system, an accelerator only needs a single calibration for determining the offset that arises due to sensor misalignment because of manufacturing, packaging, or soldering imperfections. Moreover, each board that the user uses needs a one-time calibration to determine the limits where the user hits floor deflecting the board too far.

3.1.1 Gyroscope

A simple way to determine change of the angular position around one of the device's angles is to use gyroscopes, if available. Gyroscope readings contain the angular velocity around the axis. The relative orientation around an axis can be computed by integrating the velocity over time. This method requires a calibration to fix the reference position.

3.1.2 Accelerometer

The pitch and roll angle of a stationary cellphone can easily be determined using a triaxial accelerometer, because gravity is everywhere approximately parallel to the z_w axis.[1] It

[1]The Earth is not a perfect sphere and its mass is not evenly distributed under the surface. These two circumstances lead to the varying magnitude of the gravity and its not being perfectly normal to the

should be noted, that the azimuth cannot be determined purely through the gravitational field. Assuming that the gravity is pointing towards the center of the Earth

$$\sin\varphi = -\frac{g_y}{g}$$
$$\sin\psi = \frac{g_x}{g}$$
$$\cos\varphi = \frac{g_z}{g\cos\psi}$$
$$\cos\psi = \frac{g_z}{g\cos\varphi}$$

where φ denotes pitch, ψ stands for roll, whereas g_x, g_y, and g_z are the components of the gravitational acceleration $\mathbf{g}$ parallel to x_b, y_b, and z_b, respectively.

However, the task of computing the angles becomes more challenging if the device is accelerating. Since the accelerometer measures the tension created in the MEMS structure by the superposition of gravity and acceleration it is impossible to separate the two effects without further information. Such information may be a model of the motion. Because the Fourier transformation can describe a signal as a sum of sinusoidal functions I analyzed the dynamics of a balance board moved in a sinusoidal manner. More precisely the pitch was assumed to vary according to $\varphi(t) = \varphi_0 * sin(\omega t)$, as the perfect dynamic exercise is supposed to run. The details of this investigation are described in Appendix B.

The results of the analysis showed that this leads to a nonlinear distortion (second and third order overtones, as well as additional DC component) and frequency dependent gain in the own frequency. In addition to the frequency dependency — which could be compensated using a digital filter — the gain and the magnitude of the distorting terms are dependent on the amplitude of the sine wave.

I do not know of any filter design method that could deal with such specifications, although resonator based recursive Fourier transformation could perhaps be used.[2] Starting from bin one and going through all bins, first computing the bin's real value and subtracting the distortion caused by it from the other bins. This may be a possible solution, but it is computationally very expensive.

A big advantage of the accelerometer measurements over both gyroscopes and magnetic sensors is that it suffices to calibrate the device once to find out about sensor – phone axis misalignment, and after this the pitch and roll angles can be computed independent of the geographical position and azimuth of the device.

surface.

[2]Real-time correction of the measurements is necessary for limit violation and zero-crossing detection. Resonator based recursive Fourier implementations require less computation than the execution of a traditional digital Fourier transform.

3.1.3 Magnetometer

The magnetometer-based method uses the Earth's magnetic field to extract the orientation of the telephone. The magnetic field is measured along three axes, and the direction of the field is used to determine the orientation of the phone. To get the orientation relative to the Earth coordinate system, additional information is needed. This information may come from another sensor or it can originate from constraints on the orientation on the possible orientations the board can take.

If one assumes that the azimuth of the board does not change during the exercise, then the situation is very similar to that faced when analyzing the accelerometer measurements. The pitch and roll angles can be recorded in a known position (practically when the board is in the neutral, horizontal position). Then these angles can be subtracted from each further sample and the resulting (pitch, roll) pairs show the same properties those for accelerometers of a stationary smartphone. They are both zero if the board is in central position and the angle can be computed based on the exact same principle, however, one needs to take the different magnitude of the magnetic field into account if the angles are calculated as $\arcsin(-m_y/m)$. This magnitude is dependent on position of the user on Earth.

The above method can face difficulties in regions near the Equator because the magnetic field of the Earth runs almost parallel to the surface. If the magnetic field runs parallel to one of the axes it is impossible to compute the angle of rotation around this axis, and almost parallel field may cause rounding error.

In these situations either the user is required to turn the board into a more appropriate orientation or other sensor(s) have to be used. A further potential problem with this sensor type is the vulnerability to certain metallic objects (steel, cobalt, etc.) or strong current flows nearby because they modify the magnetic field.

3.1.4 Combination of Multiple Sensors

The Android framework also provides a so-called 'orientation' sensor type, which calculates the orientation of the device relative to the horizon and magnetic north using physical sensors not defined in the documentation. According to my investigations pitch and roll are calculated only from the acceleration values, so it does not really enhance performance for this application.

I used the 'orientation' sensor to speed up prototyping, but due to my above concerns about the influence of dynamics on the accelerometer samples I chose to move to the magnetic field based solution in the final version.

It may be beneficial to implement sensor fusion with the available sensors on the platform, but I did not investigate these possibilities in this current project.

3.2 Choosing Sampling Frequency

In order to use basic tools of conventional digital signal processing like filters and Fourier transform, sensors have to be sampled with a constant sampling period.[3] The choice of the sampling frequency is crucial to the quality of the motion analysis, and it also determines to a large extent the processing load the mobile device has to deal with. The following sections describe the considerations made when choosing the appropriate frequency, and what extensions were necessary to handle the specific properties of the Android system.

3.2.1 Frequency Content of Human Motion

Properties of human motion has been analyzed among others for activity identification, stabilometry and in Parkinson's Disease related research. Although Panjan and Sarabon recommended[8] choosing sampling frequency for stabilometric measurements between 100 Hz and 1 kHz, they also state that humans cannot exceed the 15 Hz limit, so spectral components above this frequency should be filtered out. He also recommends suppressing DC and very low frequency components. These remarks correspond to the characteristic frequencies for tremor in [25] and [11]. This means that sampling frequency bigger than 30 Hz is required for complete reconstruction of the motion. This calculation corresponds to the chosen sampling frequency in [15] ($f_S = 33$ Hz) and in [19] ($T_S = 23$ ms $\leftrightarrow$ $f_S = 43$ Hz).

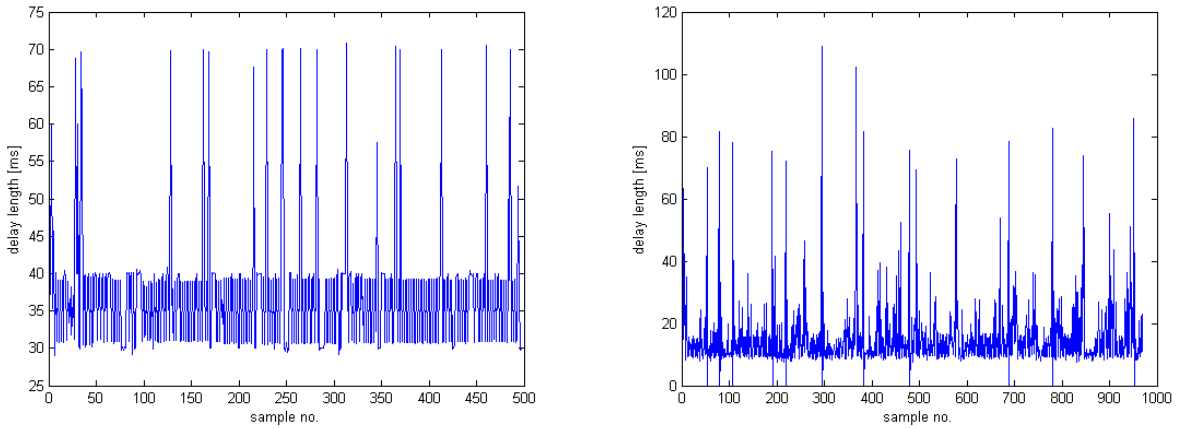
Raising sampling frequency above the minimally necessary level results in higher processor load, which in turn increases power consumption of the device and may slow down the application. On the other hand, it could lead to a decreased quantization error and smaller total noise power if the noise in the additional bandwidth without meaningful signal components is filtered out. Since an interpolation stage was also added to the system (as explained later) additional samples could make interpolation more precise.

Considering the above reasons I chose the sampling interval to 30 ms which means a sampling frequency of approximately 33 Hz.

3.2.2 Reading Sensors in Android Devices

Android devices grant access to sensor values through the SensorManager of the device by implementing callback functions of the SensorEventListener interface. The SensorManager allows the developer to register a SensorEventListener to be notified at a specified interval. However, delivery of the sensor events is not a guaranteed service, but instead the developer has to deal with best-effort delivery. Neither is it specified by the Android framework what

[3] On the other hand, compressed sensing does not require sampling at a constant sampling frequency. When using compressed sensing algorithm, sampling the signals with varying, but known, time intervals may be the wise choice.[24]



(a) Sample recorded exercise in the database

(b) Samsung Galaxy S, running the app

Figure 3.3: Delay between consecutive samples

the shortest delay between events can be,[4] nor can programmers rely on the VM to deliver each sample, one can only give 'hints'.

As a starting point about what an Android device might be capable of I investigated the recorded exercise database. Here the data were recorded using the maximum possible sampling rate, which resulted in a $T_S \approx 35$ ms sampling period with occasionally missing samples. Figure 3.3a shows the variation of the delay between samples. When testing the complete application on a Samsung Galaxy S device the delay uncertainty increased further, although the majority of the samples arrived in less than 20 ms after the previous one. Figure 3.3b shows this case. The delay cannot directly be explained with the varying length of the processing, since processing and sensor event distribution are done on different threads.

In order to deal with the missing data and the potentially varying data rate on different devices I decided to use an interpolator. It is used to approximate the orientation value between two actual samples, and create a time series of equidistant samples. This is necessary to compute the correct spectrum of the sensor data. A linear interpolator is a very simple algorithm and although other methods (e.g. spline, polynomial or sinc based interpolation) may be used, I chose this method to limit complexity of implementation and maximize execution speed. I considered the latter a very important factor in this case, since interpolation is carried out for all samples.

[4]It can be determined in runtime by the application on the device.

aspect	weight
overall impression of correctness	10
touching the ground	10
balancing moves with upper body	10
balancing moves with lower body	10
smoothness	10
injury risk (e.g. knees turned out)	10
holding, leaning with hand	5
stepping off the board	5
exercise repetitions / duration	10
pace	10
posture	10

Table 3.1: The aspects considered by the expert for assessment and their weights.

3.3 Scoring

The scoring scheme was developed based on the aspects considered by the expert who assessed the exercises in the database. He gave a score between zero and ten for all of these aspects, then they were weighted and summed to get the overall score. These aspects with the respective weights are listed in table 3.1.

Not all of these aspects can be considered for automated assessment when only the surface orientation is extracted, but they provided a guideline for designing the own scoring system.[5] In specific, balancing moves, stepping off or leaning with the hands cannot be detected by the application, nor can the posture and injury risk be derived purely from the orientation values.

The next sections describe the exact metrics used to automatically assess the exercises. some metrics are merged together into a common aspect taken from the expert's list to make overall scores as similar to the expert's scores as possible. However, they also get assessed individually and the user gets feedback on each considered parameter, to get a more detailed picture.

Some of the assessment parameters can be adjusted by the user, as it is mentioned in the following paragraphs. More detail on user preferences can be dound in section 4.3

Also note that some of the proposed metrics (e.g. mean value and variation of both deflection angles) are chosen with the assumption that the user is using a balancing board that can be tilted in any direction. There are also such boards that allow deflection only in a single direction. In this case the metrics relying on the above assumption will return

[5]Using the camera of the mobile phone to record the user's position and posture from below might help in these cases. However, investigating these possibilities is outside of the scope of this project.

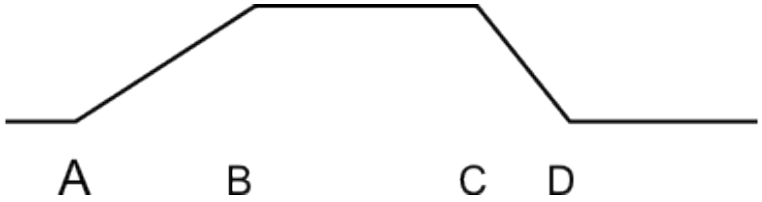


Figure 3.4: Piecewise linear scoring function

aspect	weight	ideally
overall impression of correctness	average pitch angle	0
	average roll angle	0
	deflected too far (complementary angle)	0
	average amplitude	calibrated
touching the ground	deflected too far (primary angle)	0
smoothness	average distortion of repetitions	0
	variance of periodic time	0
	variance of amplitude	0
	variance of repetition mean angle	0
repetitions	repetitions	chosen
pace	average repetition length	1 s

Table 3.2: Assessment aspects for dynamic exercises.

near ideal values. This way — though not deliberately — the intuitive expectation that that using such boards is easier will also be reflected in the final scores.

The chosen aspects are graded on a scale of 0 to 10. With the exception of the repetition count and the ground touching count this is done using a piecewise linear function with a section for the input where the aspect gets full points, sections above and below where the aspect gets zero points and linear transition sections inbetween. Figure 3.4 shows the function with its characteristic points.

The scores are assigned equal weights and added up. The total score is then scaled to get a score range of 0 to 100, which can be intuitively interpreted as a percentage of correctness.

However, in the user feedback the sub-scores of each aspect are displayed, which then have different weights in the total score.

3.3.1 Dynamic Exercises

In the case of dynamic exercises I used five out of the eleven aspects in the expert's assessment. These are listed in table 3.2 along with the metrics used for them. Below I describe them in detail.

Touching the ground : Detection of the user touching the floor with the board is conducted by monitoring the angle of deflection. When the board is tilted too far, it implies that the user touches the floor with the edge of the board. The procedure relies on the one-time calibration the user's board, which was already incorporated in the prototype version of the program. In the calibration process, the user tilts the board all the way to the left, right, forward and backward, and the application registers the pitch and roll angles in these positions. The normal exercising range is defined as all deflection angles that are smaller than 90% of the calibrated maximum deflection, in either direction.

This detection is done for the deflection angle associated with the exercise. For exercises that require forward-backward motion this angle is the pitch, for side-to-side motion exercises it is the roll. Deflection in the complementary angle also influences the total score and it is considered in the *overall impression* aspect.

Each sample for which the primary angle lies outside the normal range a feedback is delivered to the user. However, it is only counted as a new violation of the limit if there were no samples outside of the range within the last hundred milliseconds.

The maximum score for this aspect is ten points, one point is subtracted for each occasion the user touches the ground.

Repetition count : The ratio of the number of actual repetitions and the target repetition count[6] is multiplied by ten to give the sub-score for repetitions. The maximum actual number of repetitions considered for the ration is the target repetition count. The methods involved in finding repetition borders are described in section 3.4.

Pace : The average repetition length. I is not always the exercise time divided by the number of repetitions. A user may finish the exercise when the last beginned repetition is not completed, and the first cycle may start only after a little time lag (see section 4.1 for detail).

Since the repetition counts were established accurately (see Results chapter) the average length of the repetitions can also be assumed to be accurate. The question here was to find a scoring scheme that is coherent with that of the expert. However, this is a difficult task if one looks at the scatter diagram of figure 3.5. It appears that the expert was not clocking the exercises, but gave scores somewhat subjectively, because there are trends to be found, it is not a deterministic mapping from the periodic time to the the scores. The expert noted in the scoring sheet that for this aspect the target is an average repetition length of one second, and there was still one execution where only four points were given despite an average periodic time of 1062 ms. This may be therefore an aspect where the precision of computers is superior to that of human experts.

I wanted to fit a hat-like piecewise linear function optimally for the mapping, as described above, but the nonlinear minimizing function of MATLAB I used[7] converged

[6] can be set in the preferences

[7] `fminsearch`

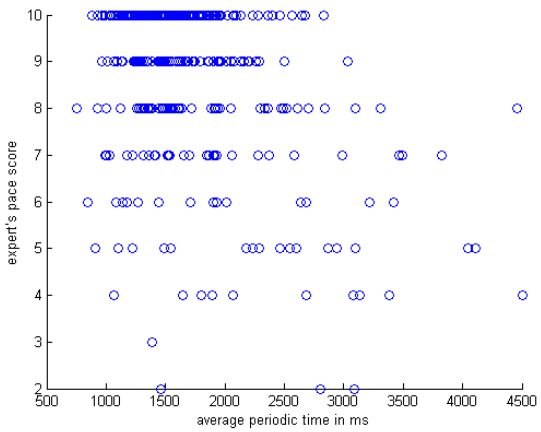


Figure 3.5: Pace score and average repetition time.

to a nonsensical solution even with additional constraints to ensure the one second periodic time gets full points. The resulting function was practically a binary decision function, giving either ten or zero points. For this reason I manually chose a function that I considered appropriate based on figure 3.5. Ten points are given for periodic time between 0.88 and 2 seconds, zero points are given for shorter than 0.5 s or longer than 4.5 s average periodic times.

Smoothness : Smoothness is measured as irregularities among and distortion inside repetitions. The former is characterized by the variance of the repetition length as well as the variance of the amplitude and the DC component in each repetition.
The latter is calculated as an average signal to distortion ratio of the repetitions. This is derived from the Fourier spectrum of a repetition by taking the ratio of the power in the first bin and the power in the next maximum five bins. If the curve of the angle takes the form of a single sine wave all the power in the spectrum resides in the first bin, otherwise overtones also contribute to the signal. I restricted the distorting components to the first five overtones because possibly they are the ones that can still be consciously controlled by the user, the others may come from jerks, tremor, etc. The four mentioned component of smoothness get weighted with equal weights.
I also compute the noise power residing in the other bins and use the white noise assumption to gain a metric independent of the samples in the actual repetition. This is *not* used at the moment, but it might be an indicator of the users condition, and thus it might sometime be used for tracking the user's fitness indirectly and not just through assessing exercises.

Overall correctness : I introduced several metrics not explicitly mentioned among the expert's criteria, but which I considered might contribute to the quality of the exercise execution. These are

- average pitch angle
- average roll angle
- number of deflections too far in the complementary direction (complementary angle)
- the average amplitude of the fitted sines in each repetition, computed as the square root of the first bin in the Fourier spectrum

The number of deflections too far in the complementary angle is counted in the same manner as the ground touching works, with a smaller limit. This limit is by default $\pm 5°$, and it can be adjusted by the user to make the exercises easier.
These sub-scores are summed with equal weights and divided by four to give the overall correctness score.

3.3.2 Static Exercises

Assessment of static exercises is very similar to dynamic ones, however, not all aspects can be taken into account. Also, due to the lack of repetitions, some of the metrics cannot be computed either.

Touching the ground : Since the goal of these exercises is to stay in a central position it is very rare to actually tilt the board so far that its edge *touches the ground*. Instead of deviation as far as the maximum position I decided to check if the user moves out of a narrower range than for dynamic exercises. This threshold checking is done for both deflection angles with the same threshold angle, there is no differentiation between primary and complementary angle. The threshold is by default at $\pm 5°$, but the user may choose wider angle.
In the first version I used the average of the violations in the two directions, after some experimenting I changed this to the maximum of the two, because it better matched the results of the expert.

Exercise length : The length score takes the place of the repetitions score of dynamic exercises. It is ten times the ratio of the actual exercise length and the target length, maximized to ten.

pace : I *do not* use this aspect for static exercises, because I could not find a rationale, although there is variation in the expert's assessment in this aspect for static exercises as well.

Smoothness : Since there are no repetitions the smoothness is instead interpreted as a stability descriptor. It is computed based on the variance of the angles.

Overall correctness : All other aspects considered before in the overall correctness being either undefined or assigned to another score, this aspect is scored based on the

aspect	weight	ideal
overall impression of correctness	average pitch angle	0
	average roll angle	0
touching the ground	deflected too far (both angles)	0
smoothness	variance of pitch	0
	variance of roll	0
exercise duration	total length	0

Table 3.3: Assessment aspects for static exercises.

average pitch and roll values only.

Each of the above aspects get a score between zero and ten each, they are summed and scaled to span the 0 to 100 interval also used for dynamic exercises. A summary of the assessment aspects is shown in table 3.3

3.4 Zero-crossing and Cycle Detection

Dynamic exercises have one primary direction, either forward–backward or left–right. The important orientation angle is pitch in the first case and roll in the second case. Zero-crossing and cycle detection is only executed for the important direction, it is assumed that at correct execution the other angle stays approximately the same during the exercise.

Users might be tilting slightly further into one direction than the other while doing the exercise. This makes accurate isolation of individual repetitions more difficult if it depends on finding zero-crossings — as it does in my algorithm. For this reason the angle measurements may be high-pass filtered if the user wants to by subtracting the moving average from the actual angle. In the further parts of this description the filtered angle is referred to as the orientation angle, and zero-crossing of the filtered angle means for the unfiltered angle crossing the moving average level. This filtering does not in any way modify the angles used for DFT computation or other steps of the assessment, its effects are restricted to finding the starting and end points of repetitions.
Using this high-pass filtering step my results for repetition count did not enhance the correlation with the expert's scores for repetitions considerably.

The assessment of dynamic exercises relies on finding the borders of each repetition. Zero crossings have to be detected and they have to be classified whether they separate repetitions or not. In ideal case each zero-crossing marks the end of a half-repetition (a time period when the board is tilted to one side) and each repetition contains one zero-crossing at each end, plus one in the middle. However, the algorithm also has to deal with the inherent noise of the human motion, which may lead to inconvenient phenomena:

- Multiple zero-crossings occur when the user only intents to cross neutral position once.
- Zero-crossings take place when the user is unable to keep steady on one side.

These phenomena are visualized on figure 3.6. These problems are far greater and more frequent if an unfiltered accelerometer is used compared to magnetometer measurements, the reason for which is described in section 3.1. The proposed algorithm was developed in the prototyping phase when I was using the accelerometer based 'orientation' sensor, and they are robust enough to deal with these effect. However, since the proposed algorithm is relatively lightweight I continued to use it after making the transition from the acceleration-based orientation in the prototype to magnetic field in the final version. I may be useful if acceleratometers are to be used as back-up resources.

If these zero-crossings were classified as limits of half-repetitions a huge discrepancy would be created between the human-perceived cycles and the machine-processed cycles. The app couldn't help the user, because it would require the users to exhibit supernatural joint control[8] and the users would probably perceive the app as faulty. In short, the algorithm

[8]People naturally exhibit different amounts of uncontrollable tremor.

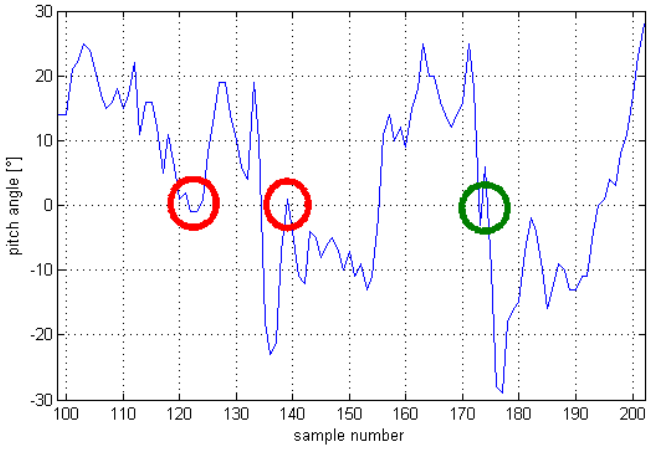


Figure 3.6: Problematic zero-crossings
multiple zeros for single transition (green) and zero crossing due to instability (red)

has to suppress the additional zero-crossings.

Noise effects are often countered with filters. One way to use them is to suppress any signal power that lies outside the intended signal bandwidth. While testing different filters on the recorded database I failed to find one that really helped make zero-crossing detection more robust for accelerometer based orientation angles. Using a 20-tap FIR lowpass filter with a cutoff frequency of 7.5 Hz still couldn't suppress the oscillating motion in the most difficult exercises, and further decreasing the passband width suppressed too many of the transitions in other recorded exercises. At this point I started to investigate the effects of motion onto this way of computing orientation, which later lead to my preference for magnetometers, as described in section 3.1. I did not, however, fully abandon accelerometer based measurements, and I chose to work with another set of methods to find transitions.

First I introduced a narrow interval of $\pm 2^\circ$ around zero degree, and all orientation (pitch or roll) angles inside the interval are classified as zero-crossings. Angles outside the interval that have the opposite sign as the previous angle sample are also classified as zero-crossings, otherwise a quick transition could skip the zero-crossing zone. This system is computationally less demanding than the filtering solution and yielded similar results.

As a second step zero-crossings are examined in the time domain, and if two zero-crossing samples are separated by only a few nonzero measurements the gap is filled. This way I managed to create a single contiguous section for transitions with multiple zero-crossings, both for a real transition or in the case where the user lost balance. The maximum gap size was determined experimentally by visually analyzing the results on the database exercises, and it was set to three samples, which corresponds to 90 ms time.

Finally the midpoint of each zero-crossing section is chosen to become the single representing zero-crossing point of the section, and this is the point that is further investigated.

If the last non zero-crossing angle before the zero-crossing section had the same sign as the first one after the zero-crossing section then the zero crossing does not represent a real transition. If the sign is opposite the zero-crossing marks the end of a half-repetition. This means that if the user makes a transition with the board from one side of the horizontal to the other, and then returns within the above 90 ms interval, it is not counted as a proper half-repetition. Half-repetitions are counted, and at the end of every second half-repetition the end of a whole repetition is marked. The first repetition is started when the initial setup time passes and the orientation angle first moves out of the zero-crossing range.

3.5 Implementation

3.5.1 Overall Considerations

Throughout the design and implementation phase I was working with a goal to make the application as resource efficient as possible without compromising code readability and maintainability. For this reason I made considerable effort to optimize critical parts of the algorithm in java, but did not use the possibilities offered by the Android Native Development Kit (NDK).[9] I also tended to create individual classes for subtasks instead of bloating existing classes. This meant that sometimes additional messaging became necessary and additional object creation was also inevitable.

3.5.2 Storing Sensor Data

Sensor values are provided as float values in a SensorEvent object and all samples taken during the exercise have to be buffered for analysis. Since this means storing a few hundred floats efficiency of the storage can influence overall resource efficiency considerably. when choosing the type of data storage I considered the following requirements (in the order of priority):

1. minimal overhead for appending data
2. ability to grow, low average overhead due to this feature
3. quick random access to the stored floats
4. small memory footprint

I prioritised speed over memory footprint because samples have to be stored in real-time. Also, if cycle analysis including DFT computation can be achieved during the exercise, after-exercise feedback can be provided much quicker, improving the user experience. However, since the application targets a handheld device, memory footprint is always a critical issue. Simple java arrays are ideal in almost all aspects, however, they do not grow automatically. For this reason I wanted to have a higher level wrapper around arrays.

Java provides the List generic interface and standard implementations thereof (like LinkedList and ArrayList). These have the benefit that they support a standard interface and thus it is familiar to most java developers. Since there is no need to use multithreading (except separating computation from UI control) synchronization is not necessary, using synchronized implementations would create unnecessary overhead.

[9]NDK allows developers to implement parts of their application in C or C++ and have it compiled. It also contains standard C libraries and an implementation of the java native interface (JNI).

The array backed ArrayList is a good option, the only slight problem with it is that the List interface only supports collections of objects – which has two major drawbacks. Firstly, the sensor value samples are provided as floats and not Float objects, so inserting them into a List includes object creation, a costly operation. Secondly and more importantly performing a lot of operations on Floats instead of floats in a DFT computation slows down the transform considerably. To avoid this a new float array needed to be created for the relevant samples and the value of the Float objects needed to be copied.

I decided to implement a new float array backed storage class (ManagedFloatArray) that is similar to a ListArray but avoids the conversions and allows classes of the containing package to access the internal array. This way the duplication of arrays is also omitted.

3.5.3 Discrete Fourier Transform

The discrete Fourier transform of a one dimensional signal is defined by equation 3.1 where k is the frequency bin index, n is the time domain sample index and N is the number of samples. Frequency bin k corresponds to the frequency $f_S \cdot k/N$.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1 \tag{3.1}$$

Direct implementation is computationally expensive, so I exploited a few possibilities to speed up the algorithm. Fast Fourier Transform is a common tool to accelerate DFT computation but it relies on the assumption that the number of the samples is an integer power of 2.[10]

In the case of the proposed algorithm, the cycles may have a length that is not a power of two, so either the array had to be filled up with zeros to an appropriate length or the FFT cannot be used. Filling up the array with zeros does not, per definition, falsify the transformed bin values. However, if the window length of the DFT is not a multiple of the wavelength of a frequency component, it cannot be mapped exactly onto a single bin, and its power is distributed over all the DFT bins (most power is assigned to bins near to the real frequency). Considering that the dominant frequency of a cycle corresponds to a wavelength equal to the cycle length, having a different window size would lead to an increased level of perceived noise and distortion.

Keeping this in mind I decided not to use FFT. I instead resorted to optimizing the computation of a traditional DFT. Starting from the naïve implementation of the DFT, where each bin value is independently computed according to equation 3.1 I gradually improved the computation efficiency. The biggest improvement was achieved in the first

[10]This is true for radix-2 FFT. There are also radix-4, radix-8 and in general radix-2^n FFT variants that rely on a sample array size that is an exponent of 4, 8 or 2^n. Higher radixes also include more elaborate initialization, which means using such algorithms might not be beneficial for small sample arrays.

step by creating a sine and a cosine array for the specific function values that will be used in the computation. This eliminates the need for frequent calls to *math.sine* and *math.cosine* functions. A further improvement can be achieved if the fact is exploited that the input is real – as opposed to the general case of complex valued signals. This results in a quasi-symmetric spectrum ($X_k = X^*_{N-k}$ where * denotes complex conjugation), so only half of the bins actually need to be computed. Since the application only uses the power at each bin the analysis algorithms can also incorporate the fact that the power at k is equal to the power at N_k so no copying of the values is needed, and the output array size can also be halved.[11]

For three of the above steps table 3.4 shows the execution times. The source code for the first and for the final final version is included in Appendix A.

type	N=50	N=31	N=16
naive	44,743	16,900	4,549
sine table	7,769	3,230	1,021
tuned	3,318	1,344	0,452

Table 3.4: Comparison of DFT implementations execution time in μs.

3.5.4 Architecture of the Assessment Implementation

The exercise analysis and assessment is distributed into two Android activities and it is done in five classes that have their own separate responsibility.

The first activity the user encounters when starting an exercise, even if he or she doesn't choose to receive assessment feedback, is the *AcceleroReader*. This activity is active from the time the user taps the 'Start Exercise' button on the exercise detail screen until the exercise finishes — in any way described in section 4.1. The software structure of the motion analysis is depicted on figure 3.7. During this activity sensor samples are read and they are sent by the *AcceleroReader* object to the *FeatureDetector*, which first has an *Interpolator* interpolate samples to adhere to a uniform sampling frequency. The *FeatureDetector* itself detects low level features such as tilting too far (using the raw sensor data) and zero-crossing (on the interpolated data), as well as repetitions. The latter two features are only searched for in dynamic exercises. The processing is done on a worker thread from the interpolation stage in order not to block the main thread.

The most complex interaction of the modules takes place if the exercise is a dynamic one, and it is visualized on figure 3.8. The *AcceleroReader* activity receives the sensor samples

[11]There are faster ways to compute the DFT. A DFT of a big array can in general be decomposed into smaller transforms. For example, the FFTW library — which also forms the basis of the `fft` function of MATLAB — decomposes the transform into smaller parts in an iterative manner until it reaches small transforms, of which the results are computed and stored at compile time. (`http://www.fftw.org/`)

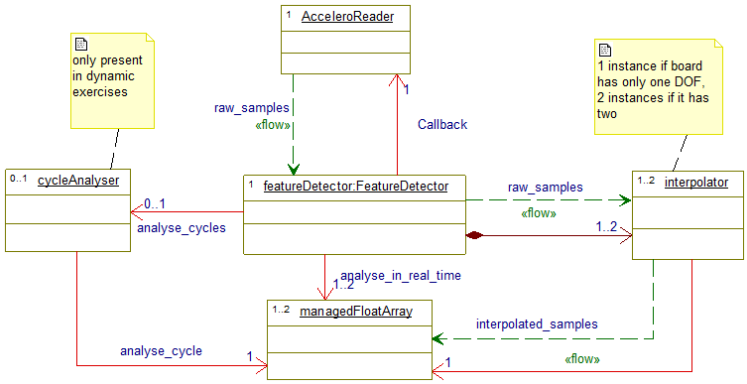


Figure 3.7: Structure of analysis software in the AcceleroReader activity

on the UI thread, and forwards the calculated orientation angles to the FeatureDetector. Most of the analysis is then done on a separate thread, where first the samples are interpolated, then checked for repetition borders. If a repetition is detected the *CycleAnalyser* is notified to analyze the last repetition. This includes the calculation of DFT and other per-repetition properties discussed in section 3.3. Once the exercise finishes the overall metrics are computed. The *AcceleroReader* activity stops at this point, the *AssessmentActivity* is started if required by the preferences with the metrics relevant for the assessment added to the Intent as extras.

The scores are assigned to the metrics in the *ExerciseScoring* class, which is instantiated inside the AssessmentActivity. This activity is responsible for displaying the scores and other user feedback. Creating a separate class for scoring made automated testing possible without starting activities.

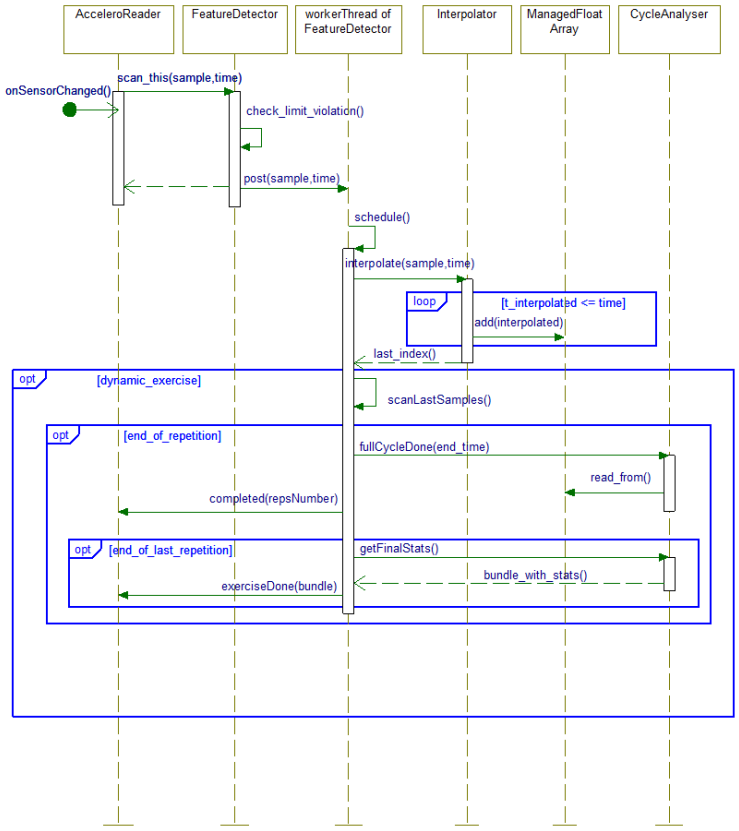


Figure 3.8: Schematic sequence diagram of class interactions for dynamic exercises

Chapter 4

User Interaction

4.1 Starting and Stopping the Exercises

After the user presses *start* on the detailed description page of the exercise the user has one to five seconds time[1] to mount the board. This period is not included in the assessment. In dynamic exercises this excluded period lasts as long as the user dose not move the board out of a narrow range of orientations around the horizontal position (referred to as the 'zero-crossing' interval in section 3.4).

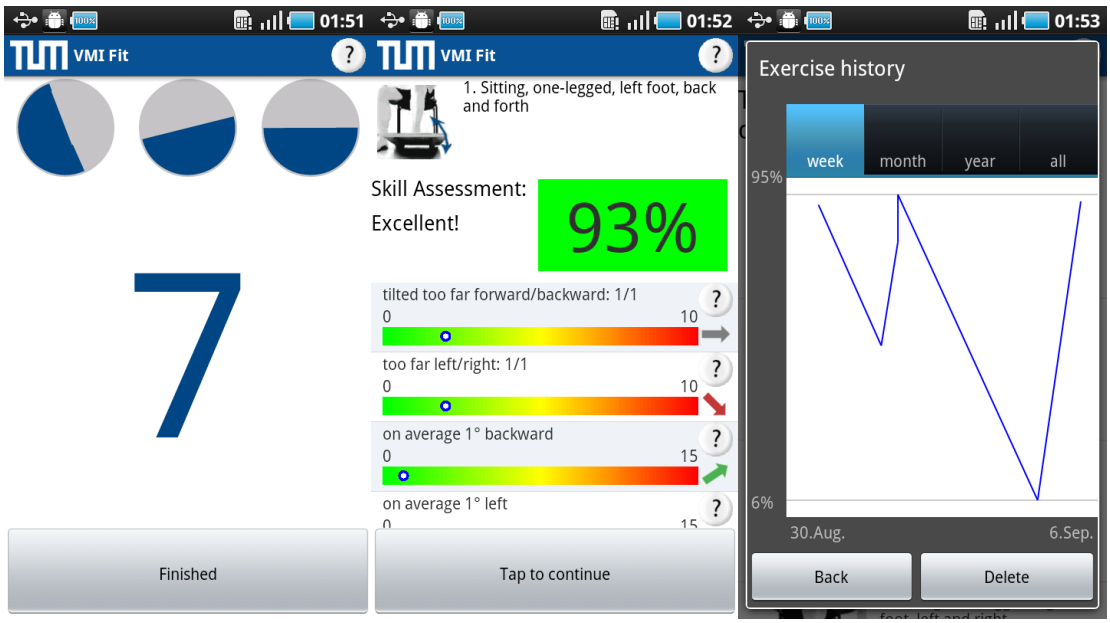
An exercise can be stopped by tapping the *Finished* button. If the user enabled the *Training Support* option in the preferences menu, then the exercise automatically stops when the user completes the predefined number of repetitions (for dynamic exercises) or the predefined exercise time (for static exercises). A dynamic exercise can also be stopped, if the board is left in horizontal position for at least one second.

4.2 Feedback to the User

In order to help users improve their technique it is inevitable that feedback is provided. It may also serve as a source of motivation. Feedback in the implemented application can be divided into three groups based on their timing, which also influences their prime goal.

Figure 4.1a shows the screen that the user sees during an exercise. During exercise instantaneous feedback is provided for two reasons. One goal is to help the user keep track of progress, i.e. count the repetitions already completed or the time passed. These data are displayed as digits on the screen of the AcceleroReader activity. If the user activated the *Training Support* option the counting is done from the target time or number of repetitions

[1] configurable in the app preferences



(a) Exercise screen (b) After exercise feedback (c) Exercise history graph chart

Figure 4.1: App screenshots

down towards zero until it is reached. If the support option is disabled the user is free to do any number of repetitions or stay on the board as long as he or she wants to, so counting is done from zero upward. An optional audio feedback component notifies the user when the initial waiting period elapses and when the exercise ends.

Intantatious feedback is also provided if the user violates some angle limits, either hitting the floor in dynamic exercises or tilting the board out of the central position in a direction where no intentional movements should be done. The feedback has a visual element, coloring the inside of the blue-white disks red if a limit is hit, the central disk (graphically displaying the actual pitch angle) is colored if a pitch limit was crossed, and the right one is painted red if the roll value exceeded the limit. It is possible, however, to add further action to either the limit violation events or the counting events; audio feedback is generally more helpful, since the user is supposed to keep an upright stance during the exercises.

After each exercise an assessment screen is showed to the user if enabled in the app preferences. Figure 4.1b shows the assessment screen. This screen shows the name of the exercise and a descriptive image related to it on the top of the display. The overall rating of the exercise is displayed next both as percentages and as a three-level categorization (weak, medium, good). This category is displayed in two forms, first, a text is displayed telling the user about which category the exercise fell, and second, the background of the percentage display is also set accordingly. Currently these are red for weak performance, yellow for medium and green for good scores, but they can be localized if necessary. In China, for example, it might be a good idea to use red as the positive color.

The scores for the individual metrics are shown below, each score is displayed using a display element with four main parts. The first part is a short textual announcement of the metric and either the score given for difficult to explain metrics like the signal-to-noise ratio or the physical quantity for concepts like mean deflection angle or average periodic time.
The second part, right below the text, is a colored bar that helps the user determine whether the current score is good or bad, and which way he or she should improve the score.
The third part of the panel is a small information icon in the top right corner where the user can get additional information about the rationale of the score. The fourth component is an arrow in the lower right corner that shows whether the user improved or got worse in the score since the last time. If one taps onto it a graph chart is displayed about the historical changes in the score.

Both the individual scores at the end of an exercise and the overall scores are stored in a database and they can be displayed on the said graph. The graph can be shown for all past data, for exercises within the last year, month or week. On opening the shortest time period is chosen in which there was at least one recorded training. The display with the graph chart is shown on figure 4.1c.

4.3 User Preferences

Android apps provide a unified interface to the user to set application-wide preferences or options. This is termed 'preferences' and it can be accessed in this app from the exercise choosing screens.

The user can find the following options in the 'Preferences' menu:

Counting mode : the user can choose to use high-pass filtered samples, this way the app alos counts repetitions that are done only on one side of the board's neutral position, making it easier for the user.

Countdown : After starting the exercise the user needs some time to get into the starting position. Here he or she can choose this time during which the sensor data are not assessed. Possibilities are 1 s, 3 s and 5 s.

Balancing limit : the user is warned if he or she moves out of the central porition too far while balancing. The threshold angle can be chosen here.

Capture sensor data : Sensor data can be saved into a log file on the 'External storage' of the Android device, which is easily accessible from an attached computer.

Training support : the user can enable various helping functionalities this way. These are

Specifying target repetition number : The user can choose the number of repetitions he or she wants to complete for the dynamic exercises. This and the next option influences the assessment of the 'repetitions/length' aspect, as described in section 3.3.

Specifying target exercise length : The user can choose the target length for static exercises.

Counting down : instead of displaying the number of repetitions already done or the time already done the user sees the amount of time left. The exercise is automatically stopped if the target is reached.

Acoustic feedback : the user can enable acoustic feedback for the occasions when the balancing limit is crossed.

Automatic assessment : Show the automatic assessment at the end of each exercise. The exercises are only assessed, and the scores only stored in the database if this option is enabled.

Self-assessment : the user can assess herself or himself after each exercise. This can be later used for example to adjust the difficulty of the exercise or the automatic assessment may be more lenient with beginners. This data is not used at the moment.

4.4 Logging

In order to evaluate the usability of a smartphone application it is useful to have a way to log the user interactions. If several applications are to be logged, it is useful to develop an easy-to-use interface for the application developers that allows flexible logging of various information. As a part of my project I investigated the possibilities that the Android framework offers, and implemented a logging facility that leverages these possibilities.

Three main requirements were set for the development of this framework:

1. The logging module should be as easy to use as possible for the application developer.
2. It should make remote access to the logs available for the developers, so that users or testers do not have to manually copy or send log files.
3. Various kinds of data will be logged, including activity changes, touch gestures, later possibly even sound. Therefore, the interface should be as general as possible.

Android devices are shipped with two different sets of logging classes, one stemming from the java language and one developed for Android.

The java language itself comes with a general and modular logging framework inside the *java.util.logging* package that is intended for logging development data. The API is based

upon three classes performing different tasks.

Logger the class that the developer accesses when he or she wants to log data. It offers shorthand methods for logging entry to or exit from methods, or for logging messages with seven different severity levels. The levels themselves are defined in the Level class.

Handler takes LogRecord objects form a Logger and writes its content into some specific output. A subclass of Handler is defined among others for storing records in the memory, printing it onto a console or writing into a socket, but it can also be subclassed by developers.

Formatter is used by the Handler to create a String out of the LogRecord that can be printed to the selected output.

Individual Loggers can and should be assigned to each logged entity. Handlers for each Logger can filter the LogRecords in a flexible way, and the virtual machine automatically determines the class and method where the LogRecord was logged.

The Android framework offers, in addition to the java logging API, the *Log* class and supporting tools in the SDK. This is less powerful but easier to use than the java API. The Log class has static logging functions, using which messages and throwables can be logged into a few central circular buffers, and these buffers can be read using the *logcat* tool of the Android SDK. When logging a message a developer is required to choose a tag and a severity level for the message, and the displayed messages can be filtered later in logcat using these two properties.

Since one of the goals was the ability to send logged messages over the Internet the pure Android approach did not fit my purpose. Instead, I decided to build my logging framework around the java logging API, and cater for automatic routing of the messages either to a socket connected to a logging server, or to a log file. If the connection to the server is active, the data is sent there, otherwise the user activity and other events are logged into a logfile on an external storage. The latter has two reasons. Firstly, it is easily accessible from a computer which is attached to the mobile device, whereas the application's dedicated local storage is only accessible from within the application process. Secondly, applications have a restriction on how much data they can occupy on the device. Since the logged data can grow quickly if there is no Internet connectivity, this limit may be reached quickly, which could result in the application running out of storage for the core functionality, or – since program objects in the virtual machine and data on the storage count together towards the limit – it may even crash. The latter can not happen, if the logfile is stored on the so-called external storage of an Android device. There the logfile is stored in the library structure as recommended by the Android developers' guide, so that the data is removed when the application is uninstalled. If neither the server, nor the data file can be accessed for modification (the latter can happen if the device is mounted to a computer, for instance), then the facility falls back to the Android logging buffers, and messages are logged according to the severity of the log message.

Currently, the logging functionality works as follows. I have developed a subclass the Android Service class, LogService, which has to be included in the application manifest. This service includes a BroadcastReceiver and it keeps track of the configuration changes in connectivity and external storage status, and calls the appropriate FileHandle or SocketHandle to actually log the message. It receives messages from a member of the PluginHandler class, a subclass of Handler, which has to be called in the application software as any Handler to log a message. A PluginHandler starts the LogService if it is not running yet, and sends the published record to the Service to deal with it.

This architecture was chosen to keep the often-instantiated part (the PluginHandler) small and simple, so if it has to be included in multiple logged classes, the overhead is kept at a minimum. Having a single object access the file system and write to the socket also avoid problems arising from concurrent access. If the service is made public, it could even be shared between applications.

The original idea was to automatically log various user interactions, without the developer manually including any code. Unfortunately, as far as I know, this cannot be accomplished using the java language due to the structure of Android Activities. In order to log click events, for example, a line of code has to be included into the onClick method of a View's OnClickListener. This could be done automatically, if this OnClickListener object could be accessed. Then a new OnClickListener could be constructed that first logs that the event has happened, and after that call the old onClick method. However, there is only a setter method in Views, and no getter. This means that although all Views could be reached automatically, starting from the root View of an Activity and traversing the whole View tree using getChilds method, the OnClickListener objects cannot be modified so easily. It would require an additional parser to be run at compile time to find Views for which an OnClickListener is defined and expand the body of the onClick method. In addition, possibly possibly all non-overriden Views would need an overridden OnClickListener log if a user wants to interact with a View that is not designed to be interactive. OnClickListeners are just one example, the same holds for any other kind of user interface event.

Chapter 5

Results

The algorithm was evaluated using the recorded and assessed database. I constructed a simple app that run the whole chain of assessment for each expert-assessed exercise.

The relevant database items were copied to the phone (the expert did not assess the whole database), and the resulting scores — both the overall scores and the scores for individual aspects — were stored into text files. These files were then loaded into MATLAB where the analysis of the results took place. The aspects that cannot be assessed by the app (as described in section 3.3) were removed from the overall score of the expert, and this new overall score was compared to that of the automatic assessment.

The assessment was conducted twice with the same algorithm using the magnetometer the first time and the accelerometer the other time to investigate the differences in their performance. One difference was introduced between these two sensor types, I increased the allowed deflection for the checking ground touching for the accelerometer based version to 100% of the calibrated maximum to account for some of the distortion caused by motion. The magnetometer measurements usually brought more accurate results. As expected, the score difference was pronounced for dynamic exercises but hardly noticeable for static ones, where the distortion of accelerometer measurement caused by motion was not as severe. I expected first and foremost the repetition count, ground touching and smoothness scores to suffer from the disturbing effects of motion discussed earlier. However, the repetition count did not show big differences, which I attribute to the robustness of the proposed zero-crossing detection algorithm. The main characteristics of the scores are summarized in table 5.1 and table 5.2 for dynamic and static exercises respectively.

Since the assessment was done on a previously established database it was impossible to accurately calibrate the magnetometer sensors. I looked for time periods in the accelerometer readings where both pitch and roll were very close to zero[1], and took the average of magnetic field components during these periods to compute the offset pitch and roll. If

[1] absolute value smaller than 3 degrees

there were no such magnetometer samples, which occurred 17 times out of the $24 \times 20 = 480$ recorded exercises, I took the average of the magnetic field sensor readings.

For dynamic exercises, I first investigated the scores related to the repetition count, because this is a score that cannot be fine-tuned by changing parameters of the last step of the scoring system, but also because the further scores in the assessment depend on the appropriate identification of repetitions. The results for both sensor types show that in the majority of the exercises the repetition count was accurate (zero difference from the expert's count). For a considerable number of test cases the automatic detection returned one or two repetitions fewer, but on looking into the reasons I found that much of this error was due to the unfortunate way the records were stored in the database. For many of them the last half repetition is cut off, which would be necessary for detection of the last complete cycle. Another usual phenomenon is that the start of the record shows distinctly that the user has already deflected the board before the recording started.

I checked all the assessed executions of the first, second and twentieth exercise individually to find out whether the repetition counter is really as unreliable as the result statistics hinted.[2] Among the twenty-four executions of each were 6, 4 and 7 cases respectively where the automatic counter returned one repetition less then required, and upon visual assessment I only found one case where a human could have found a repetition not counted by the app, however, it was also a fairly incorrect cycle. In the set for the second exercise there were also two cases where the automatic counter lagged the expert's counting by two counts, but these cases were also unfortunately subject to truncated recording. I therefore think that the actual performance of the functionality is much better then what the statistics of the assessment of the repetitions suggest.

I did not explicitly test the correlation of the actual exercise length and the expert's score on this aspect. This is an aspect that can be measured very precisely with smartphones, and therefore I do not think this can be of any concern.

Next I compared the expert's pace score for dynamic exercises with the own score. For more than 68% of the exercises the automatic scoring with magnetometer had smaller than or equal to 2 points error, despite the difficulties described in section 3.3. This figure is about 65% for accelerometer, the difference is not too pronounced.

Touching the ground aspect scores did not reliably mimic the expert's scores. About 75% of the assessed dynamic exercises were given the accurate score using magnetometer samples, whereas only 43.8% of the scores were accurate for accelerometers. However, both had a very poor correlation with the human assessment, achieving a cross-correlation coefficient of 0.269 with magnetometers and 0.036 with accelerometers.

The total score of the system had a less then or equal to 10% error in 77.9% of the cases with magnetometers and a maximum of $\pm 20\%$ error 94.3% of the time. Acceleratometer based scores' figures were 65.9% and 90.1% respectively. However, the correlation coefficient was

[2] I only checked a few random examples among the other exercises this way.

aspect	bias	$\|\Delta\| < 2$ points	correlation
ground touched	-0.30	92.71%	0.269
repetitions	-0.38	96.88%	0.759
smoothness	-0.08	75.78%	0.445
overall correctness	-0.39	68.49%	0.293
pace	-0.59	57.81%	0.440

(a) Aspect scores, magnetometer

bias	$\|\Delta\| < 2$ points	correlation
-1.78	69.79%	0.036
-0.23	97.40%	0.760
-0.84	69.53%	0.572
-0.23	68.49%	0.334
-0.43	58.07%	0.419

(b) Aspect scores, accelerometer

	bias	$\|\Delta\| < 10$ points	$\|\Delta\| < 20$ points	correlation
magnetometer	-3.48	77.86%	94.27%	0.512
accelerometer	-7.01	65.89%	90.10%	0.585

(c) Total scores

Table 5.1: Comparison of sensors, dynamic exercises
Bias: average of (automatic score - expert score); $|\mathbf{\Delta}| < \mathbf{n}$ **points**: percentage of automatic scores this far from expert scores; **correlation**: Pearson's linear correlation coefficient

found to be higher for accelerometer based assessment.

The automatic scores for static exercises also correlated well with the human assessment scores. As table 5.2 shows, here accelerometer based assessment outperformed magntetometers in some respects. 90.6% of accelerometer scores fell into the ± 10 point range, a slightly bigger proportion then that of magnetometer measurements, but the mean difference of the automatic and the human scores were -0.99, whereas for magnetometers the average difference was smaller than 0.1 points. These differences are, however negligible considering that the weighting of the individual metric scores might be further optimized.

The results show that neither sensor type has overwhelming advantage over the other.

During field tests in the laboratory the magnetometer based solution did not robustly recognize the angles that correspond to the board hitting the ground, the app did not show an alert. This behavior was dependent on the exact location of the board inside the laboratory. I attribute this to the effect mentioned earlier, that in an environment with plenty of metal objects around the phone the assumptions about a homogeneous magnetic field do not hold.

aspect[3]	bias	$\|\Delta\| < 2$ points	correlation
ground touched	0.25	98.96%	
overall correctness	1.64	76.04%	0.310
smoothness	-1.04	76.04%	0.664

(a) Aspect scores, magnetometer

bias	$\|\Delta\| < 2$ points	correlation
0.19	97.92%	0.116
1.35	77.08%	0.237
-1.20	77.08%	0.676

(b) Aspect scores, accelerometer

	bias	$\|\Delta\| < 10$ points	$\|\Delta\| < 15$ points	correlation
magnetometer	0.08	89.58%	98.96%	0.764
accelerometer	-0.99	90.63%	97.92%	0.778

(c) Total scores

Table 5.2: Comparison of sensors, static exercises
Bias: average of (automatic score - expert score); $|\mathbf{\Delta}| < \mathbf{n}$ **points**: percentage of automatic scores this far from expert scores; **correlation**: Pearson's linear correlation coefficient

Chapter 6

Conclusions

In the presented project I developed an automatic assessment system for balancing board exercises, including both signal processing and user feedback in the application.

Taking the scoring system of an expert instructor as a starting point I developed a set of aspects and related physical metrics to assess the exercises. The metrics included time domain descriptors, such as exercise length, repetition count or variation of repetition length, statistical moments of the pitch and roll angles' distribution, and frequency domain descriptors of individual repetitions in dynamic exercises. The metrics were then scored individually and an overall score was given to the exercise by weighting and summing the individual metrics' scores.

The automatic assessment with a smartphone can be executed based on the orientation of the phone. The orientation can be computed using various types sensors, I have investigated accelerometer based and magnetometer based solutions in detail. The former assumes that the measured acceleration is equal to the gravity of the Earth, the latter relies on a stable, locally homogeneous magnetic field, typically the magnetic field of the Earth. Both methods have advantages and related difficulties. The acceleration based method is influenced by the real acceleration of the mobile phone, but it works in all regions of the world, and it is not influenced by the environment in which the exercises are done. The magnetic field measurements are not corrupted by the movement dynamics of the device, but it might be unusable when the magnetic field runs horizontally, and it is also useless when the magnetic field is distorted by nearby metallic objects (e.g. steel table and chair legs) or strong currents.

A comparison of the automatic scores with those given by the expert showed strong correlation using either above sensor, especially for static exercises.

The feedback used in the app can be divided into three groups according to their time horizon. Instantaneous visual and audio feedback is provided during the exercise for warning the user if he or she deflects too far and if the exercise is started or completed. The

repetitions are also counted in dynamic exercises.
After-exercise feedback shows the overall assessment, and also summarizes the user's achievments in individual aspects. The scores are color coded for easy comprehension, but numeric results and the ideal range are also displayed. Additional information is provided for each aspect.
Historic data is used to maintain long-term motivation by enabling tracking of the results. Overall score development over time can be accessed from the exercise choosing screen, changes in the detail scores is only accessible directly after the exercises, from the assessment screen. The historical feedback is built up of two components, firstly, an arrow next to the assessment indicating the change from the last execution, and secondly, a multi-scale graph, where the results for the last week, month or year, or all results of the user are displayed.

The app is now capable of providing useful feedback for the user, however, there is room for further improvement.

One important possibility for improvement is the combination of multiple sensors. The current application relies on the magnetometer sensor, and it requires only minor changes to use accelerometers instead. However, the application cannot combine data from multiple sensors, nor is it capable of automatically determining which sensors are available. Sensor fusion could mitigate the drawbacks of each investigated sensor and benefit from the advantages of both. Conditionally integrating a gyroscope would enhance the measurement reliability in devices that are shipped with such a sensor.

The scoring functions and their relative weights may also be improved using statistical learning algorithms for nonlinear regression or classification, as in [17]. This approach would first require a selection of the correctly stored exercises from the database.

The user interface may also be modified based on user acceptance. Feedback from a user study could be useful for evaluating the current concepts and making the interactions more intuitive. This not only includes the graphical part, the audio feedback during exercises may require changes, e.g. differentiation between beeps due to limit violation according to the limit violated.

A further question that may be worth investigating is whether user compliance can be improved if an adaptive scoring system or an adaptive training goal is used. User feedback for wellness applications in [9] suggests that this might be motivating. However, the first option also means a dilution of the standard scores, and makes long term graphs less meaningful. Currently the software makes self-assessment possible, so users might set the difficulty this way, if the self assessment levels are not just logged (as it is the case now) but also influence scoring or a future goal setting and achievement tracking subsystem of the app.

Appendix A

DFT variants

```
/** original, simple implementation */
public static final double[] dftMagnitudeSimple(double[] input) {
    int N = input.length;
    double[] mag = new double[N];
    double[] c = new double[N];
    double[] s = new double[N];
    double twoPi = 2 * Math.PI;

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            c[i] += input[j] * Math.cos(i * j * twoPi / N);
            s[i] -= input[j] * Math.sin(i * j * twoPi / N);
        }
        c[i] /= N;
        s[i] /= N;

        mag[i] = Math.sqrt(c[i] * c[i] + s[i] * s[i]);
    }

    return mag;
}
```

```
/** final optimized DFT */
public static final float[] dftPowerRealInput(float[] input,
        final int firstIndex, final int N, float[] output) {
    final int validOutputBins = N / 2 + 1;
    final int lastIndex = firstIndex + N;
    float c;
    float s;
    double twoPi_per_N = 2 * Math.PI / N;

    float[] sines = new float[N]; // contains sine values
    float[] cosines = new float[N]; // contains cosine values

    // build sine table
    sines[0] = 0;
    cosines[0] = 1;
    for (int i = 1; i < validOutputBins; i++) {
        // Exploit only symmetry related to f(t) <-> f(2*PI - t),
        // not f(t) <-> f(PI - t), because they might not be both
        // present in the array.
        sines[i] = (float) Math.sin(i * twoPi_per_N);
        sines[N - i] = -sines[i];

        cosines[i] = (float) Math.cos(i * twoPi_per_N);
        cosines[N - i] = cosines[i];
    }

    // check if offered array is sufficient
    final float[] dftBin;
    if (output == null)
        dftBin = new float[validOutputBins];
    else if (output.length < validOutputBins)
        dftBin = new float[validOutputBins];
    else
        dftBin = output;

    // handle k=0 separately
    dftBin[0] = 0;
    for (int j = firstIndex; j < lastIndex; j++) {
        dftBin[0] += input[j];
    }
    dftBin[0] /= N;
    // compute power
    dftBin[0] = (float) ((double) dftBin[0] * (double) dftBin[0]);
```

```
    // write X[N-k] and X[k] in the same cycle
    for (int i = 1; i < validOutputBins; i++) {
        c = s = 0;
        for (int j = 0, inpIdx = firstIndex; j < N; j++, inpIdx++) {
          c += input[inpIdx] * cosines[(i * j) % N];
          s -= input[inpIdx] * sines[(i * j) % N];
        }
        c /= N;
        s /= N;

        dftBin[i] = (c * c + s * s);

    }

    return dftBin;
}
```

Appendix B

Acceleration due to Motion

B.1 General Formulae

The balancing board is built up of a cylindrical or spherical section that rolls on the floor and a flat surface to stand on. The cross-section of the board is depicted on figure B.1 It is characterized by the radius R of the spherical or cylindrical section, the distance r of the flat surface from the center of the curvature and the maximum deflection φ_{max}.

Let us that the cellphone is laid on the board so that the y axis of the device, as introduced in section 3.1 is parallel to the plane of the cross section and the x axis is running into the plane. The distance of the accelerometer from the center of the flat board is denoted with d.

The world coordinate system (x_w, y_w and z_w) is aligned with the board when it stands in the neutral position. Let us investigate the case when there is movement only in the y direction, i.e. rotation around the x axis. This corresponds to the *pitch* angle. For this reason the positions, movements and accelerations will be described only with two coordinates y and z.

Let us first describe the motion of the board and the sensor in the world coordinate system. When the board rolls on the ground the ground the center of curvature stays at the same height, its motion is purely translational. When the board is tilted with a positive pitch angle φ it moves along axis y_w by $R\varphi$. In this case the measured acceleration due to gravity is[1]

$$\mathbf{a_g} = g \begin{bmatrix} -\sin\varphi \\ \cos\varphi \end{bmatrix} \tag{B.1}$$

[1] with the assumption of constant g on the Earth

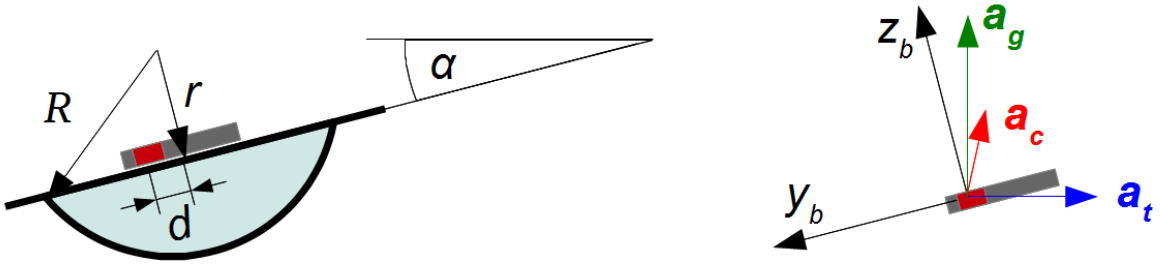


Figure B.1: Cross-section of a balancing board
The smartphone is depicted as a gray rectangle lying on the board, the accelerometer is positioned at the red rectangle. The different acceleration components are shown on the right for this specific position (a_t may also point into the opposite direction).

The motion of the sensor can be regarded as a superposition of a translation together with the curvature center and a rotation around this center. This means a rotation with the same angle φ as the deflection of the surface of the board and moving on a circle with a radius $\sqrt{r^2+d^2}$.

Let us denote the accelerations originating from these two motions $\mathbf{a_t}$ and $\mathbf{a_c}$ for the translational acceleration and the centripetal acceleration respectively. They are investigated in the board coordinate system, so that it can be established what the sensor measures.

The translational acceleration is horizontal, since the curvature center only moves on a horizontal line.

$$\mathbf{a_t}(t) = \frac{d^2}{dt^2}(R\varphi(t)) \begin{bmatrix} \cos\varphi(t) \\ \sin\varphi(t) \end{bmatrix} = R\ddot{\varphi}(t) \begin{bmatrix} \cos\varphi(t) \\ \sin\varphi(t) \end{bmatrix} \tag{B.2}$$

Since the centripetal acceleration always points to the center of the rotation the direction of $\mathbf{a_c}$ is determined by the unit vector

$$\frac{1}{\sqrt{r^2+d^2}} \begin{bmatrix} -d \\ r \end{bmatrix}$$

irrespective of the $\varphi(t)$ function. Its magnitude is

$$\|\mathbf{a_c}\| = \omega^2\sqrt{r^2+d^2} = \dot{\varphi}^2(t)\sqrt{r^2+d^2}$$

where $\dot{\varphi}$ denotes the first derivative of φ with respect to the time. These two equations can be merged into

$$\mathbf{a_c} = \dot{\varphi}^2(t) \begin{bmatrix} -d \\ r \end{bmatrix} \tag{B.3}$$

B.2 Sinusoidal Motion

If $\varphi(t)$ is a sinusoidal function we can explore the phenomena further.

$$\begin{aligned}\varphi(t) &= \varphi_0 \sin(\omega t)\\ \dot{\varphi}(t) &= \varphi_0\omega \cos(\omega t)\\ \ddot{\varphi}(t) &= -\varphi_0\omega^2 \sin(\omega t)\end{aligned}$$

From this the acceleration components are as measured by the accelerometer

$$\begin{aligned}\mathbf{a_g}(t) &= g\begin{bmatrix} -\sin(\varphi_0 \sin(\omega t)) \\ \cos(\varphi_0 \sin(\omega t)) \end{bmatrix}\\ \mathbf{a_t}(t) &= -\varphi_0\omega^2 R \sin(\omega t)\begin{bmatrix} \cos(\varphi_0 \sin(\omega t)) \\ \sin(\varphi_0 \sin(\omega t)) \end{bmatrix}\\ \mathbf{a_c}(t) &= \varphi_0^2\omega^2 \cos^2(\omega t)\begin{bmatrix} -d \\ r \end{bmatrix}\end{aligned}$$

This shows that the acceleration has a nonlinear effect on the signal. Figure B.2 shows the effect of this distortion for one and two hertz motion. Considering that for small φ using the fist terms of the Taylor series expansion $\sin\varphi \approx \varphi$ and $\cos\varphi \approx 1-\varphi^2$ one gets to the equations

$$\begin{aligned}a_y &= -\frac{d\varphi_0^2\omega^2}{2} + \left(-g\varphi_0 - R\varphi_0\omega^2 + \frac{3R\varphi_0^3\omega^2}{8}\right)\sin(\omega t) - \\ &\quad -\frac{d\varphi_0^2\omega^2}{2}\cos(2\omega t) + \frac{R\varphi_0^3\omega^2}{2}\sin(3\omega t) \qquad \text{(B.4)}\\ a_z &= g - \frac{g\varphi^2}{4} + \frac{r\varphi_0^2\omega^2}{2} - \frac{R\varphi_0\omega^2}{2} + \left(\frac{g\varphi^2}{4} + \frac{r\varphi_0^2\omega^2}{2} + \frac{R\varphi_0\omega^2}{2}\right)\cos(2\omega t) \qquad \text{(B.5)}\end{aligned}$$

Assuming that the board never flips upside down one can always be sure that z component of a_g is always positive. For this reason one doesn't need to consider the z component for computing the pitch (or roll) angle.

$$\varphi = \arcsin(a_y/g)$$

The dominant part of a_y are the components with ω angular frequency. This can be filtered if an appropriate filter is designed. The goal of the filter is to invert this effect. The specification is therefore

$$|H(\omega)|^2 = \frac{1}{(1+\omega^2/p^2)^2}$$

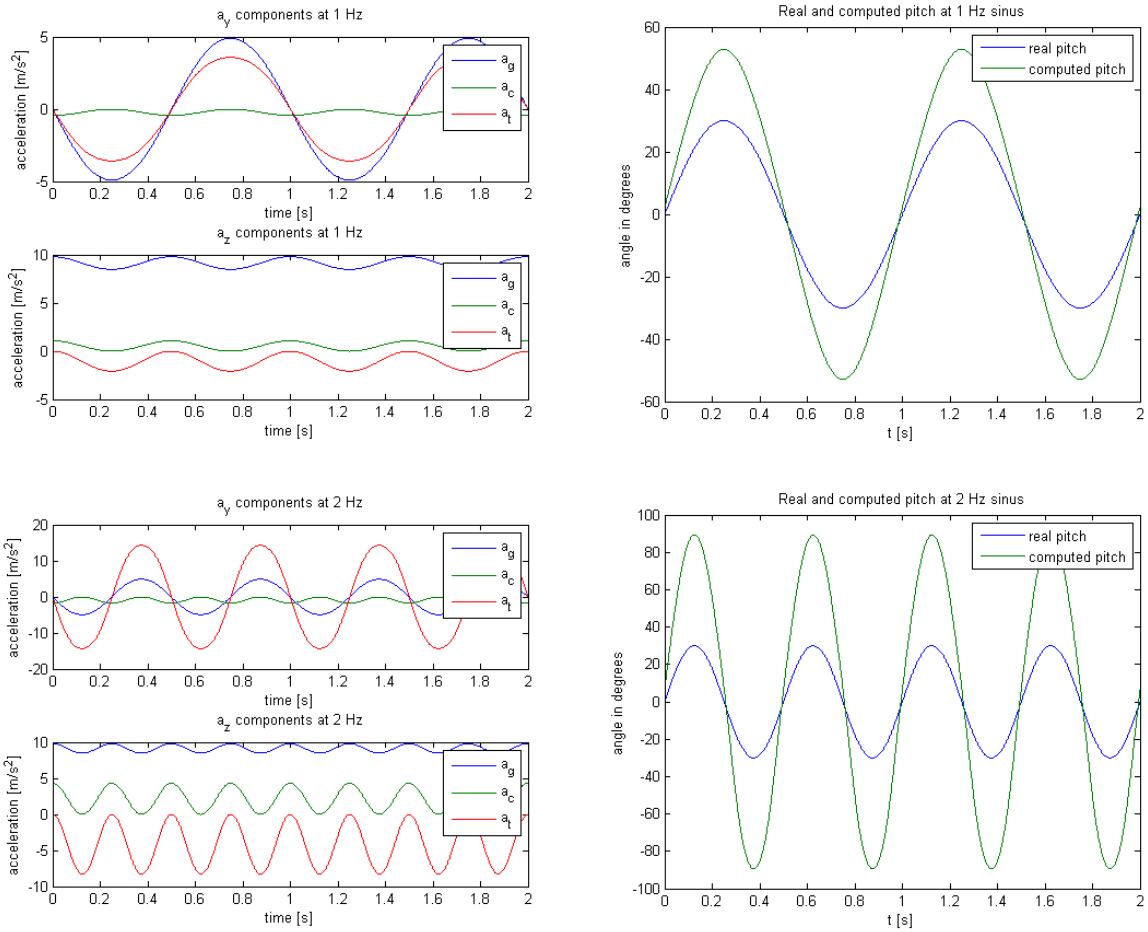


Figure B.2: Influence of sinusoidal motion on orientation calculations
Simulation results with $R = 20$ cm, $r = 10$ cm and $d = 4$ cm.

where

$$p^2 = \frac{g}{R\varphi_0\omega^2(1 - \frac{3\varphi_0^2}{8})}$$

$$p = \sqrt{\frac{g}{R\varphi_0\omega^2(1 - \frac{3\varphi_0^2}{8})}}$$

This means the specification relies on the knowledge of the board, but also on the knowledge about the movement (φ_0 amplitude at the specific frequency), even with the assuption of sinusoidal motion around the center. This specification describes a non-linear filter. Since I do not have any knowledge or experience about designing non-linear filters I did not continue investigations along this path, and turned to magnetic field measurements.

List of Figures

List of Tables

Bibliography

[1] V. H. Hildebrandt, P. M. Bongers, J. Dul, F. J. H. van Dijk, and H. C. G. Kemper. The relationship between leisure time, physical activities and musculoskeletal symptoms and disability in worker populations. *International Archives of Occupational and Environmental Health*, 73:507–518, 2000. 10.1007/s004200000167.

[2] Patricia Heyn, Beatriz C. Abreu, and Kenneth J. Ottenbacher. The effects of exercise training on elderly persons with cognitive impairment and dementia: A meta-analysis. *Archives of Physical Medicine and Rehabilitation*, 85(10):1694 – 1704, 2004.

[3] William L. Haskell, I-Min Lee, Russell R. Pate, Kenneth E. Powell, Steven N. Blair, Barry A. Franklin, Caroline A. Macera, Gregory W. Heath, Paul D. Thompson, and Adrian Bauman. Physical activity and public health : Updated recommendation for adults from the american college of sports medicine and the american heart association. *Circulation*, 116(1):1081–1093, 2007.

[4] K. Söderman, S. Werner, T. Pietilä, B. Engström, and H. Alfredson. Balance board training: prevention of traumatic injuries of the lower extremities in female soccer players? *Knee Surgery, Sports Traumatology, Arthroscopy*, 8:356–363, 2000. 10.1007/s001670000147.

[5] A. Caraffa, G. Cerulli, M. Projetti, G. Aisa, and A. Rizzo. Prevention of anterior cruciate ligament injuries in soccer. *Knee Surgery, Sports Traumatology, Arthroscopy*, 4:19–21, 1996. 10.1007/BF01565992.

[6] Philip J. van der Wees, Anton F. Lenssen, Erik J.M. Hendriks, Derrick J. Stomp, Joost Dekker, and Rob A. de Bie. Effectiveness of exercise therapy and manual mobilisation in acute ankle sprain and functional instability: A systematic review. *Australian Journal of Physiotherapy*, 52(1):27 – 37, 2006.

[7] L. J. Mazur, R. J. Yetman, and W. L. Risser. Weight-training injuries. common injuries and preventative methods. *Sports Medicine*, 16(1):57–63, 1993.

[8] Andrej Panjan and Nejc Sarabon. Review of methods for the evaluation of human body balance. *Sport Science Review*, XIX(5-6):131–163, 2010.

[9] A. Ahtinen, E. Mattila, A. Vaatanen, L. Hynninen, J. Salminen, E. Koskinen, and K. Laine. User experiences of mobile wellness applications in health promotion: User study of wellness diary, mobile coach and selfrelax. In *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1 –8, april 2009.

[10] A. Khasnis and R. Gokula. Romberg's test. *Journal of Postgraduate Medicine*, 49(2):169–72, 2003.

[11] Károly János Bretz. *Measurement Technique of Human Force, Coordination and Tremor*. PhD thesis, Budapest University of Technology and Economics, Department of Measurement Technique and Information Systems, 2010.

[12] Yuriy Terekhov. Stabilometry as a diagnostic tool in clinical medicine. *Canadian Medical Association Journal*, 115:631–633, 1976.

[13] Ross A. Clark, Adam L. Bryant, Yonghao Pua, Paul McCrory, Kim Bennell, and Michael Hunt. Validity and reliability of the nintendo wii balance board for assessment of standing balance. *Gait & Posture*, 31(3):307 – 310, 2010.

[14] A. Schouten, T. Boonstra, F. Nieuwenhuis, F. Campfens, and H. van der Kooij. A bilateral ankle manipulator to investigate human balance control. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, PP(99):1, 2011.

[15] E. Sabelman, A. Fiene, and A. Timbie. Accelerometric activity identification for remote assessment of quality of movement. In *Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE*, volume 2, pages 4781 –4784, sept. 2004.

[16] Robert Diemer and Samarjit Chakraborty. Mobile phone assisted cooperative on-node processing for physical activity monitoring. In *8th Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS)*, Waidhofen, Austria, 2010.

[17] P.E. Taylor, G.J.M. Almeida, T. Kanade, and J.K. Hodgins. Classifying human motion quality for knee osteoarthritis using accelerometers. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 339 –343, 31 2010-sept. 4 2010.

[18] L. Cunningham, C. Nugent, G. Moore, D. Finlay, and D. Craig. Identifying fine movement difficulties in parkinson's disease using a computer assessment tool. In *Information Technology and Applications in Biomedicine, 2009. ITAB 2009. 9th International Conference on*, pages 1–4, nov. 2009.

[19] Min Wang, Bei Wang, Junzhong Zou, Lanlan Chen, F. Shima, and M. Nakamura. A new quantitative evaluation method of parkinson's disease based on free spiral drawing. In *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*, volume 2, pages 694 –698, oct. 2010.

[20] S. Hoque, M.C. Fairhurst, and M.A. Razian. Modulating population granularity for improved diagnosis of developmental dyspraxia from dynamic drawing analysis. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 26 – 31, oct. 2004.

[21] M. T. Ma and H. Burgsteiner. Validating the use of gaming console sensors for telemonitoring of physiotherapy exercises. In *Tagungsband der eHealth2011*, may 26-27 2011.

[22] J.-A. Gil-Gomez, J.-A. Lozano, M. Alcaniz, and S.A. Perez. Nintendo wii balance board for balance disorders. In *Virtual Rehabilitation International Conference, 2009*, page 213, 29 2009-july 2 2009.

[23] Android Open Source Project. Android SDK Reference. http://developer.android.com, accessed 29. Aug. 2011.

[24] E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489 – 509, feb. 2006.

[25] Mario Manto, Giuliana Grimaldi, Thomas Lorivel, Dario Farina, Lana Popovic, Silvia Conforto, Tommaso D'Alessio, Juan-Manuel Belda-Lois, Jose-Luis Pons, and Eduardo Rocon. Bioinformatic approaches used in modelling human tremor. *Current Bioinformatics*, 4(2):154–172, 2009.