Detection of equestrian falls using smartphone sensors

Malin Magnusson

2017

Master's Thesis in
Biomedical Engineering

LUND
UNIVERSITY

Faculty of Engineering LTH
Department of Biomedical Engineering

Supervisor: Josefin Starkhammar

Abstract

Horseback riding is enjoyed by millions of people worldwide, both as a competitive sport and as recreational activity. Horses have a congenital habit to flee from potential danger and the nature of the sport carries with it a risk of falling from the horse. Equestrians are often riding alone and the additional risk of being unaided after a severe fall accident is prominent. The need of a smartphone application able to sense a fall and automatically contact preselected relatives is identified. An equestrian fall detection algorithm using smartphone sensor data is developed in this master thesis. The work was made from scratch, including data recording of both normal horseback riding activities and simulated fall events. Additional signal representations were derived and features calculated. Their contribution of interesting information was evaluated and the most suitable features made the final threshold based fall detection algorithm. The algorithm was implemented as an Android application running on real time sensor data for evaluation. In total, 53 out of 55 simulated falls were detected and no false alarm was obtained during eleven hours of normal riding activity. The algorithm is considered as suitable for its purpose of increasing equestrian safety.

Keywords: Fall detection, Smartphone sensors, Equestrian safety

Preface

This report presents the master thesis ending five years of Biomedical Engineering studies. The work is based on knowledge from several courses and plenty of years of equestrian life. In addition, this work has been a learning process, giving important insights for the future. The work has sometimes been arduous, but the opportunity to combine engineering with a field close to the heart has been a pleasure.

A huge thank to my supervisor Josefine Starkhammar for always being available and supportive throughout the whole process of work. Your positivity has helped me overcome the heavy parts of this project. Also a huge thank to all riders and horse owners helping with both data recordings and evaluation, the work had not been possible without you.

Lund, maj 2017
Malin Magnusson

Contents

1 Introduction

This section describes the background, need and purpose of this project. It ends with a brief explanation of the report structure.

1.1 Background

Horseback riding is a sport enjoyed by millions of people nationwide [1]. Sweden has the highest number of horses per capita in the European Union [2]. Horseback riding is one of the most popular sport activities among children and teenagers, half a million swedes are riding regularly. People are enjoying horseback riding as an recreational activity, exercise or a competitive sport [3]. There are a lot of different equestrian disciplines, but the largest and most known are dressage, show jumping and eventing [4].

The evolution of the horse started 50 million years ago and it has developed from having three toes and being the size of a dog. They started to become domesticated 6000 years ago and today the size varies by breed. A riding horses range in hight from 1.5 m up to 1.8 m and weighs from 500 kg to 800 kg, ponies are smaller [5]. Horses are flight animals and it is a congenital habit to flee from potential danger [6]. In addition, the animal's movements can be irregular and unpredictable [7]. Because of the nature of the sport, horseback riding is associated with a certain inherent risks. It is not so much a matter of if a rider is going to fall, but whether such a fall will result in an injury. Most equestrian-related injuries occur during schooling or noncompetitive riding. Novice riders have been found to be at an overall increased risk of injury. However, professional riders suffer from more severe injuries. They ride at faster speeds and train and compete over significantly increased obstacle heights in jumping disciplines. Particularly at the professional level, there is a tendency to ride and train less experienced or more volatile horses [8].

Falling from the horse is the most common cause of equestrian injury, with fractures and head injuries as consequences. When mounted in the saddle, the rider's head is approximately 2.7 meters over the ground and a fall can deliver 100-300 g of deceleration to the upper body and head. An added risk of injury is that the horse land on, or trample the rider after the fall [7] [8]. Horseback riding related injuries tends to be serious [9]. Eventing is often considered the most dangerous of competitive horse sports and in the cross-country phase of eventing alone 59 deaths occurred between 1993 and 2015 [10]. 2016 was a dark year for eventing with five more deaths [11].

For equestrian activities, protective equipment has primarily focused on the use of appropriate helmets for head and neck protection. The use varies between regions and disciplines [12]. The Swedish Equestrian Sport Federation (Svenska Ridsport Förbundet) has decided that helmets must be worn during all horseback riding within the federation's activities [13]. Other protective equipment are safety vests and stirrups designed to avoid getting stuck during a fall. A relatively new technology is inflatable air vests attached to the saddle. If a fall occurs and the rider leaves the saddle, the vest deploys [14].

Most falls occurs away from competitions [15] and equestrians are often riding alone, both in the forest and in the paddock or drill hall. It is recommended to bring a mobile phone when riding [13] [16], but if the rider falls and becomes unconscious or immovable, the smartphone is today useless. The risk of an injured rider to be unaided after a severe fall accident is prominent.

1.2 Purpose

The need of a smartphone application able to automatically sense an equestrian fall from the horseback is identified. Such an application could be able to send messages to preselected relatives in case of a fall, for increased security.

The specific aim of this master thesis was to develop an equestrian fall detection algorithm usable in such a future smartphone application. The algorithm must be functional on real time smartphone sensor data and be able to distinguish different types of falls from various normal horseback riding activities.

1.2.1 Scope of this project

The focus in this project was on usage of sensor data and algorithm development. The resulting fall detection algorithm was implemented as an Android application to evaluate real time feasibility. No work on alarming functions or user interface of the application in vision was made.

The development process was made for Android only. Later implementations of the algorithm for iOS was not taken into consideration in this project.

1.3 Report structure

This report starts with a chapter called *Theory* in order to present related work and basic knowledge of the smartphone sensors. This to give the reader a short introduction to the field and facilitate further understandings.

The work within this project has consisted of four separate and succeeding steps. The methods and deliverables of these steps are presented individually in separate chapters since the result of one step forms the basis for the next one and so on.

The development work is based on a set of smartphone sensor data from both simulated fall events and normal horseback riding activity. The method of recording this data is presented in the chapter *Data Recording* together with the total amount of data used in this project.

The work of successively developing the equestrian fall detection algorithm is described in the chapter *Algorithm Development*. It is the heaviest chapter of this report. It ends by presenting the result of this master thesis, the final equestrian fall detection algorithm, in section 4.8.

The chapter *Java Implementation* presents the implementation of the algorithm to an Android application. The challenge of handling real time sensor data is described and the final structure of the application is presented.

A small evaluation of the Android application is done within this project. Both methods and results of these tests are presented in chapter *Evaluation*.

Discussions of overall methods and results are presented in chapter *Discussion*. Thoughts about future work and possibilities are included, as well as other interesting reflections made during the project process.

The last chapter of this report, *Conclusion*, presents a summary of the work made in this master thesis together with conclusions about the obtained result.

2 | Theory

This chapter presents the related work and it also gives an introduction to the smartphone sensors associated with this project.

2.1 Related products and work

Some work have been done aiming to solve the identified risk of unaided equestrians after severe fall accidents. To my knowledge there is no product or application widely used among riders today. When searching for applications on Google Play Store, only "Horse Rider SOS" and "Safe Rider" were found. Non of them has more than 5 000 downloads worldwide. Both applications relies on GPS data only and the alarm is activated if the position does not change for a defined length of time [17] [18] [19]. This means no fall detection algorithms are implemented. In addition to fall accidents, there are several reasons for being immovable when riding. For example waiting to pass a road with driving cars or similar cases. If the alarm is activated in a stressful situation it can do more harm than good.

In 2015 four students at Chalmers University of Technology made an application called ”Ryttarjalp” as their bachelor thesis. As conclusion they considered the application ready for launching on Google Play Store, but it has not happened yet. Due to secrecy the total algorithm design is not published in their report. In short it uses the magnitude of accelerometer and gyroscope data combined with two verification steps to finally decide whether a fall has happened or not. The first step consists of a comparison between the current signal and the features of each gait. If a gait is found the ride is considered as still ongoing. If not, the second step is activated where the algorithm uses the speed and accelerometer data to see if the device is moving. If not, a fall is detected. The algorithm finds 16 out of 17 simulated falls and gives one false alarm every tenth hour in total, but one false alarm every hour when riding in the paddock or drill hall [20].

A system sold to riders in Sweden is ICE dot, where ICE is an acronym for In Case of Emergency. It has an external sensor unit mounted on the helmet and is used together with a smartphone application. It costs about 890 SEK and won the prize "Equestrian gadget of the year” in 2015, awarded by a large Swedish horse magazine named Hippson. It is not customized for horseback riding but rather sold to cyclists originally [21] [22]. The product is expensive in comparison to downloading an application for the same purpose.

One advantage of the external sensor unit is to be able to adapt the choice of quality of sensors when constructing the system, to make them suit the fall detection problem.

Some riders use an alarm application without automatic fall detection when riding alone. An example is the application "Allone". It was invented by a father worried about his 16 years old daughter, both when riding and in late nights. This application has three functions. "Main Alarm" alerts a loud sound and sends information about who you are and where you are to everyone having the application installed in a certain radius. The secure alarm, "My Alarm", sends information to pre-selected relatives. The function "Watch Me" allows relatives to follow the rider's GPS position in real time using the same application on their own device [23] [24].

Fall detection is an important and growing technology in other fields as well. Bikers, cyclists and elderly are also exposed for the risk of falling with severe injuries as consequence. Both smartphone sensors and external sensors have been used for detecting these falls in the literature, see [25]. An other related field is motion classification, dealing with analysis of different movement patterns. Unlike fall detection it is often a non binary problem, but other similarities and useful information are found, see [26]. Due to the amount of research in these areas, they are used as a source of inspiration in this project.

Even an application classifying horse movements is on the market today. ”Equilab” uses smartphone sensor data to determine the amount different gaits during a ride. It also provides the user with other types of interesting information such as energy consumption and stride length [27]. The prize given to ICE dot in 2015, "Equestrian gadget of the year”, was awarded to Equilab in 2016 [28]. Technical products is a highly topical field in the equestrian sport today.

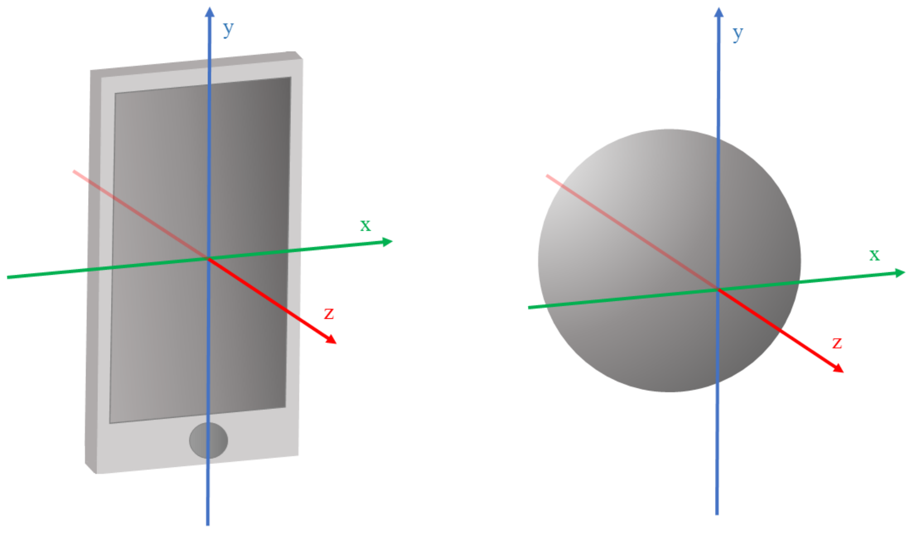
2.2 Smartphone sensors

There are many large smartphone vendors, but only two operating systems (OS) shares around 99 % of the market today [29]. Apples own OS for iPhones is called iOS while almost every other large brand uses Android. This project was limited to focus on developing an application for Android only, not for iOS. It is more accessible to start building applications for Android and the used language is Java, which I know from previous courses. Hence the theory in this section is mostly gathered from the Android Developer site, and some information yields for Android only.

Sensor availability varies from device to device. Some sensors are hardware based and some are software based. Hardware sensors are physical components and most smartphones nowadays have accelerometers, gyroscopes and magnetometers to register motion and position. Software sensors are not physical units, they derive their data from one or more of the hardware based sensors. They are sometimes also referred to as virtual sensors or synthetic sensors [30].

Sensor data can be used either individually or combined. Sensor fusion is the combining of sensory data from disparate sources such that the resulting information is in some sense better than would be possible when these sources were used individually [31]. Some of the software sensors, e.g. the orientation sensor, are the result of sensor fusion already made by Android. Of course sensor fusion techniques can also be implemented later by the developer.

In general the sensor framework uses a 3-axis coordinate system to express data values. For most sensors, the coordinate system is defined relative to the device. The x-axis is horizontal and points to the right of the device, the y-axis is vertical and points up, and the z-axis points towards the outside of the front face of the screen [30], see figure 2.1a.



(a) Coordinate system of the device. (b) Coordinate system of the earth.

Figure 2.1: Coordinate systems used for sensor data.

Some sensors provides information about the phone's position relative to the earth. In the earth's coordinate system the z-axis is perpendicular to the ground, and thus the negative z points to the direction of the earth gravity. The x-axis roughly points to east and the y-axis points to magnetic north [32], see figure 2.1b .

2.2.1 Accelerometer

An accelerometer is an electromechanical device that measures acceleration forces. The unit of acceleration is m/s^2. There are a lot of different types of accelerometers [33], but the fundamentals is that they have two parts; a fixed housing attached to the object whose acceleration we want to measure and a mass that while tethered to the housing, can still move [34].

Accelerometers in smartphones needs to be really tiny. Hence MEMS (Microelectromechanical system) technology is used and the accelerometer is made out of silicon. The appearance is illustrated in figure 2.2 where the gray background represents the housing. The red comb-like section is the mass that can move back and forth thanks to the flexibility of the silicon as tethering. Around this comb-like section there are three fingers making up a differential capacitor, illustrated in blue in the figure. When the mass moves, current will flow and it is possible to correlate the amount of flowing current to acceleration [34] [35].

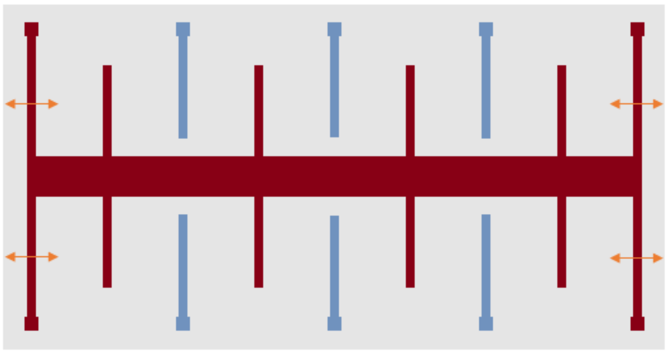


Figure 2.2: Illustration of an accelerometer. The red part is the mass that can move back and forth. When doing so, the capacitance between the blue "fingers" change and it is possible to correlate the amount of flowing current to the acceleration [35].

The accelerometer in the smartphone measure the acceleration applied to the device in all three directions of the phone's own coordinate system and gives three dimensional data. A sensor of this type measures all acceleration affecting the device, including the force of gravity that is always influencing the measurements [32].

2.2.2 Gyroscope

Gyroscopes are physical sensors that detect and measure angular velocity, usually expressed in degrees or radians per second (rad/s). The type of components used to measure rotation in electronic devices can be quite varied, but vibrating structure gyroscopes of MEMS technology are often used in smartphones. The angular velocity is measured using the Coriolis effect. If a mass moves in a particular direction, with a particular velocity, and an external angular velocity is applied to the mass, a Coriolis force which will cause a perpendicular displacement of the mass appears [35].

A model of the vibrating gyroscope is shown in figure 2.3. The blue parts are fixed while the green part is allowed to move in one direction by the black flexible tethering. When an angular velocity is applied the red part moves perpendicular to the green part by the Coriolis force. Similarly as in the accelerometer these displacements will cause changed capacitances in the comblike sections which will be measured and processed to correspond to a particular angular velocity [35].

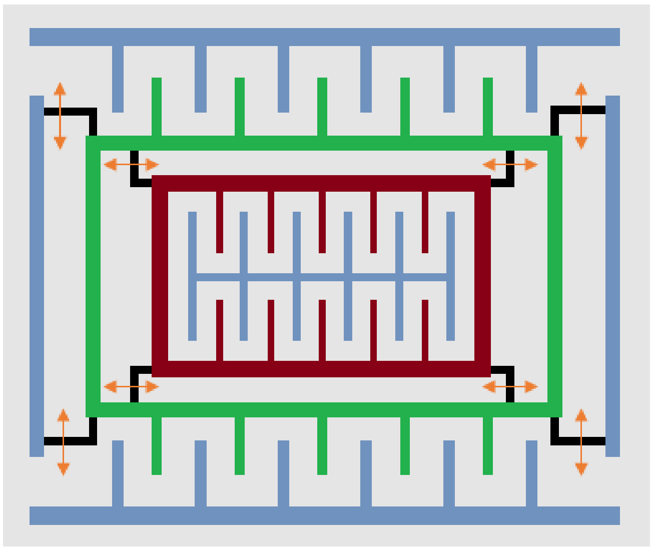


Figure 2.3: Illustration of a gyroscope. The blue parts are fixed while the green part can move in one direction. The red mass moves perpendicular to the green frame when an angular velocity is applied. This will cause changes in capacitance, corresponding to particular angular velocity [35].

For an Android device the three-axis gyroscope measures the rate of rotation around the device's local x, y and z axis and the unit is radians per second. The rotation is positive in the counter-clockwise direction [32].

2.2.3 Magnetometer

A magnetometer measures the magnetic field affecting the device in microtesla (μT). This information helps to detect the magnetic north, providing information about which direction the device is facing on the earth. Magnetometers can work in a number of different ways. Some uses the Hall effect where the magnetic field affects a current to cause a voltage that is measured. Some uses the magneto-resistive effect where the resistance of a material sensitive to the magnetic field is measured [35] [36]. In Android the magnetometer sensor is called Geomagnetic Field Sensor and it provides raw magnetic field strength data for each of the three coordinate axes of the device [37].

2.2.4 Linear acceleration and gravity

The total acceleration of a device can be divided into linear acceleration and gravity. Android provides both as software sensors. The linear acceleration sensor gives the acceleration force applied to the device on its own three physical axes (x,y,z), excluding the force of gravity. The gravity sensor on the other hand indicates the direction and magnitude of gravity applied to a device in the same directions. Both sensors provides information in m/s^2. The sum of the vectors containing linear acceleration and gravity should in theory be equal to the total acceleration [32].

2.2.5 Orientation

The orientation of a phone is used to monitor the position of the device relative to the earth's frame of reference. The sensor provides information about three orientation angles in degrees. The output is calculated using the magnetometer combined with the accelerometer data. All orientation angles are shown in figure 2.4.

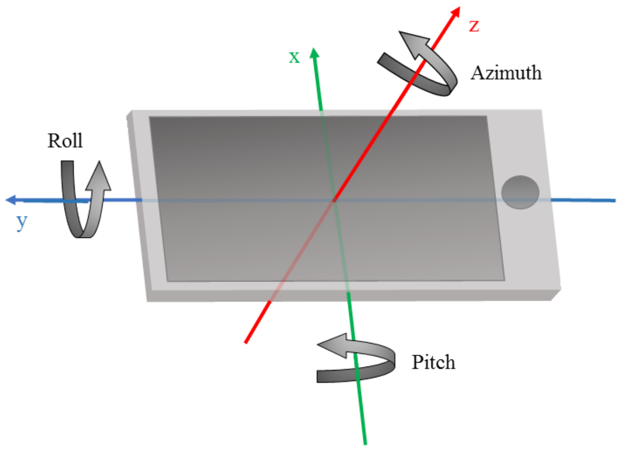


Figure 2.4: The angles azimuth, pitch and roll expressed by the orientation sensor provided as a software sensor by Android.

The degrees of rotation about the z-axis is called *azimuth*. It expresses the angle between the device's current compass direction and magnetic north. The output ranges from 0° to 360°. It is 0° when the top edge of the device faces the magnetic north and 180 ° when it faces south. The degrees of rotation about the x-axis is called *pitch* and ranges from -180° to 180°. The angle is zero when the device is parallel to the ground and positive when tilting the top edge toward the ground. *Roll* is the degree of rotation about the y axis. The range of values are -90° to 90°. Positive values correspond to a tilt of the left edge of the device toward the ground [37].

2.2.6 Other sensors

Smartphones are equipped with several additional sensors and positioning systems. These are not used in this project and hence not further described. Some of them, as the software speed sensor and the GPS, are used in related works. Others are environmental sensors for other purposes. Examples of these are the ambient temperature and pressure, and a proximity sensor to determine how far an object is from the device [30].

3 | Data Recording

This chapter describes the work of data recording, both from normal riding activities and from simulated fall situations, starting with a description of the overall recording method. This chapter only handles recordings of data used for algorithm development. The method for testing the resulting algorithm is described later.

3.1 Recording method

To get a perception of smartphone sensor data in general and from horseback riding and fall incidents in specific, it was necessary to have recordings of data. The data formed the framework for future work of algorithm development, and the quality of data was important.

Only one smartphone was used for all recordings during this project. Not all smartphones have the same sensors, the variety of data was thus not captured within the scope of this project. The used smartphone was of model Nexus 5X, made by LG Electronic. This device is equipped with all hardware sensors described in the theory chapter, namely accelerometer, gyroscope and magnetometer. These are all components from Bosch. The accelerometer and gyroscope is found in a combined measurement unit called BMI160, see [38]. The magnetometer is provided in a separate unit called BMM150, see [39]. The properties of the hardware sensors are presented in table 3.1.

Sensor:	Accelerometer	Gyroscope	Magnetometer
Resolution:	0.002394 m/s^2	0.0005326 rad/s	$\sim$ 0.3 μT
Max. range:	78.45 m/s^2	17.45 rad/s	$\pm$1300 μT (x, y-axis) $\pm$2500 μT (z-axis)
Min. sample times:	2.462 ms	2.462 ms	19.69 ms

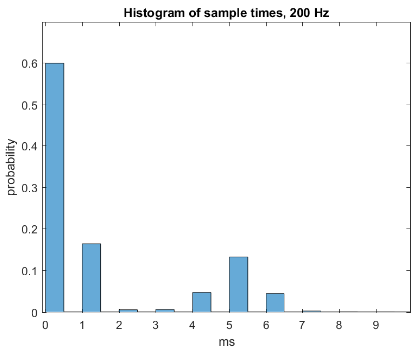
Table 3.1: Properties of hardware sensors in the used smartphone, a LG Nexus 5X, with measurement units from Bosch.

The resolution and maximum range of the accelerometer and gyroscope are variable in the measurement unit [38]. These values were provided by the application "Sensors Multitool", see [40]. The maximum sample rate was derived by the differences between sample times when setting the sensor rates as fast as possible handling them in Android Studio, a program for building Android applications.

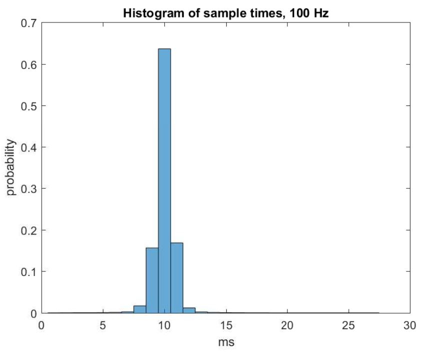
The OS version of the smartphone was upgraded during the project from 7.0 to 7.1.1. This change had no impact on sensor recordings in this project. Besides data from hardware sensors, data from three software sensors was saved as well. It was the linear acceleration, gravity and orientation. The maximum sample rate of these sensors was discovered in the same way as for the hardware sensors, i.e. using time differences. The maximum sample time was 5.000 ms for all software sensors.

To start the process of data recording early on in the project, the decision was to use one of several free applications for sensor recordings. The application of choice had to be able to collect data from several sensors, both software and hardware, at the same time. The data had to be saved in a format loadable into MATLAB, the program used for data analysis and algorithm development in this project. After some research and tests of different applications, the choice was on AndroSensor, made by Fiv Asim, see [41], [42]. It saves data as comma-separated values (CSV)-files, automatically sent by email to the user when a recording is ended.

The maximum sample time when writing to CSV-files in AndroSensor was 5 ms, i.e 200 Hz. The number of decimals was adjustable up to six. A problem using too high sample rate or too many decimals was discovered. It seemed like the application can not handle too much information when writing to CSV-files and it affected the signals. An example of this problem is shown in figure 3.1 where the origin times between two samples are presented as histograms for two different sample rates.



(a) The sample time when writing to CSV-file with a frequency of 200 Hz. The expected sample time is 5 ms.



(b) The sample time when writing to CSV-file with a frequency of 100 Hz. The expected sample time is 10 ms.

Figure 3.1: Evaluation of origin sample time between two samples at different sample rates using the application AndroSensor.

To the left, in figure 3.1a, are the sample times using 200 Hz shown. The expected value is 5 ms and to show the result in a visual manner, the x-label is limited up to only 10 ms, there are bins representing higher values. This spreading of values shows that the sample rate does not work. The histogram in the right figure, 3.1b, is centered around the expected value 10 ms, even though it is not perfect. The choice was to use a sample rate of 100 Hz and four decimals when recording the data to avoid the illustrated problem but still get as detailed signals as possible.

How the placement of the smartphone impact sensor data was not evaluated in this project. Most likely the result changes with this placement, but it is left for future work to evaluate how big the impact actually is. Most riders have their phone in a pocket of their jacket when they ride [20] and that was the placement during data recording in this project. The start orientation of the phone relative to the ground was not specified and varied in the recordings.

Data were recorded for different purposes. Some files were recorded only to evaluate sample rates, get a first hint of sensor data and so on, while others represented daily horseback riding and simulated fall incidents used for algorithm development.

3.2 Normal horseback riding activity

It is important for the algorithm to be able to distinguish a fall from the normal activity related to different types of horseback riding. Too many false alarms would lower the reliability for the application and make the user tired of false notifications. They are disturbing and even a danger because of the riders attention towards the application instead of the horse in stressful situations.

The horse used for the recordings was a gelding with a hight of 170 cm of the breed Swedish Warmblood. This horse has normal gaits and is calm and collected. He does what he is told so the data is not interrupted by unexpected events. Most data was recorded with my self as a rider. Some jumping data was recorded by another equestrian with an even larger horse of the same breed.

The goal during data recording of normal riding activity was to get as much difficult data as possible and go for quality before quantity. Difficult data means data containing more sudden movements being similar to a fall, such as jumps and dismounting the horse. High quality data here refers to data recorded during controlled circumstances.

The recordings of normal riding activity resulted in totally 341 minutes of data with mixed content presented in table 3.2. The mixed dressage recordings includes data where each gait, i.e. walk, trot and canter, is recorded separately to be able to study their properties in the sensor data. Less common gaits belonging to individual breeds, such as flying pace and "tölt" are not included in this project.

The mixed jumping data is the result of two separate training sessions with the smartphone placed in a pocket of the jacket worn by the rider. Both files includes the warm up and then different types of jumps over fences up to 1 m. The drawback of this data was that it is difficult to determine what are jumps and what is just canter between the fences. Therefore an other rider was employed to wear the smartphone during a jump session while the times of all jumps were notified, using an additional clock. This made it possible to study separate jumps.

The decision was made that it was unnecessary to record more data with the same horse and rider during the same circumstances. The guess was that it would be difficult to separate data during mounting and dismounting from fall data. Hence was some of these events recorded separately and they are included in table 3.2. The other data of course contains mounts and dismounts as well, these are only the additional recordings. It was assumed that movements during trail riding are similar to the ones when riding in the paddock and no such data was recorded.

Type	Total Time (min)	Number of events
Mixed Dressage	167	
Mixed Jumping	120	
Monitored Jumping	54	
Mounts		5
Dismounts		7
Total	341	12

Table 3.2: Recorded data of normal horseback riding activity used for algorithm development.

3.3 Fall simulations

For ethical reasons it is not possible to record real falls within this project. Hence the falls must be simulated in some manner. There are many different ways to perform fall simulations in the related literature. Use of mattresses, mechanical bulls, stunt mans, and virtual human bodies are some examples [20], [43], [44], [45]. Much effort was put on the choice of how to perform fall simulations. To use some kind of mathematical simulation would have been difficult, especially to get smartphone sensor data from such simulations. If a human would have done the fall simulations there must be some kind safety equipment as mattresses and it would have been difficult to record the falls from a real horse. Due to these reasons, the idea of placing the smartphone on something else than a real human person during fall simulations turned out to be the best way forward in this project. It was desirable to have something making the smartphone hit the ground in similar ways as it does when placed on a human body. For example changing direction when one body part hits the ground or being subdued by the body.

It was not possible to within the given time frame find a human body model to borrow or rent. The final choice was to make my own human body model to use for fall simulations. A body frame of fabric was bought and then filled with papers and pillow material to make it as like a real human body as possible. The model was 1.80 m tall but would have been almost impossible to handle if it had a true body weight. As long as the aerodynamic drag can be excluded it should not affect the sensor data remarkably to have a lower weight if the body proportions are true. The total weight was 9.5 kg and the approximate proportions are presented in table 3.3. Bags of flour was added to specific body parts to get true proportions.

Body part	Proportion of human body weight [46]	Weight of model
The trunk	43 %	4 kg
Head and neck	7 %	0.7 kg
Upper limbs	13 %	1.3 kg
Thighs, lower legs and feet	37 %	3.5 kg
Total	100 %	9.5 kg

Table 3.3: Proportions of the human body model used for fall simulations. The total weight was 9.5 kg and it was 180 cm tall.

The used human body model is shown on a horse in figure 3.2. The horse is relatively small, only 155 cm high, and hence does the model look a bit big. It is fully equipped with riding gear including helmets and boots.

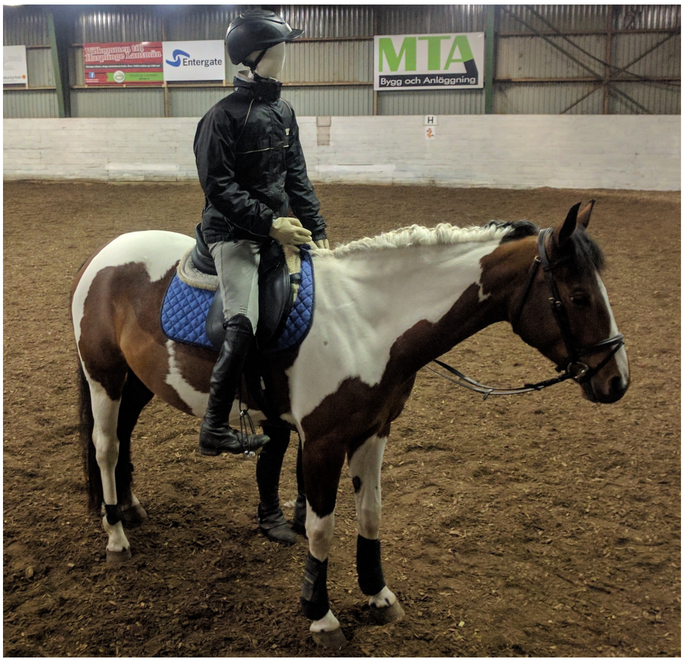


Figure 3.2: The Human model used in this project on the very patient horse called Sessan, during one of the fall recording sessions.

By use of literature, movies of falls and my own experience some common fall situations were identified. The reason for falling varies and also the appearance of a fall, but the most common ones were imitated using the human body model and specified in table 3.4

The original plan was to let the model fall from a horse in all simulated events. The model could not sit in the saddle it self. I tried to ride with the model in front of me in the saddle and then suddenly stop the horse and let it fall of. It worked decent but I obstructed the model from falling free. Additionally it was not right towards the horse to do too many recordings of this type, it got tired of the situation pretty quick. Instead simulations were made having the model on my own shoulders. I ran and drop it off to make it fall similarly as when falling from a horse.

Unfortunately there was a problem with the sample rate, similar to the one earlier mentioned when having too high sample rate during some fall recordings, making them useless. Some simulations from the horse were included in this problem. Because of this, the amount of falls from the horse usable for algorithm development was reduced to only three.

Totally 41 usable falls were captured. The number of different types is presented in table 3.4. Simulations of falling forward, backward and sideways were recorded on grass. The falls imitating a rider doing a somersault over the horses neck, being stuck in the stirrup and dragged after the fall and falling into a fence were recorded on a ground similar to the ones in paddocks or drill halls.

The number of different types of falls does not represent how common they are, the goal was rather to catch five of each sort. An exception was falls representing when being dragged after a fall. In the end there was eight usable falls of this type.

Type of fall	Number of falls
From horse, walk	2
From horse, trot	1
Forward	5
Backward	5
Same side as the device's placing	5
Opposite side to device's placing	5
Somersault forward	5
Fall in fence	5
Being dragged after the fall	8
Total	41

Table 3.4: Number of successfully recorded fall simulations of each type.

4 | Algorithm Development

This section begins with a description of algorithm development and ends by presenting the resulting equestrian fall detection algorithm of this project. The work is described in succeeding steps, even though the work has not been that straight forward in reality.

4.1 Choice of sensors

The first important decision was to determine which recorded signals could be of interest to use when building the algorithm. Except the study of raw signals, inspiration was obtained from related literature.

An equestrian fall consists of six independent movements, both of the rider her or him self and the device worn by the rider. These movements are illustrated in figure 4.1. It is a composition of three linear movements and three orthogonal rotation movements [33], i.e. it has six degrees of freedom.

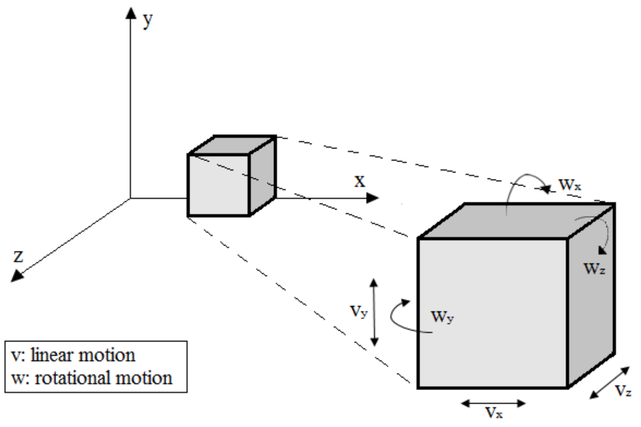
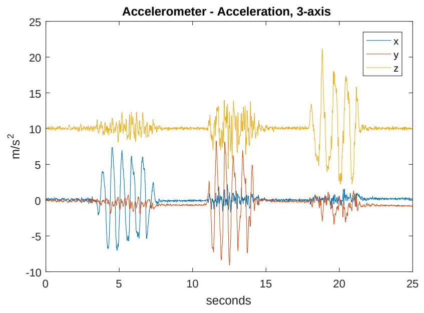


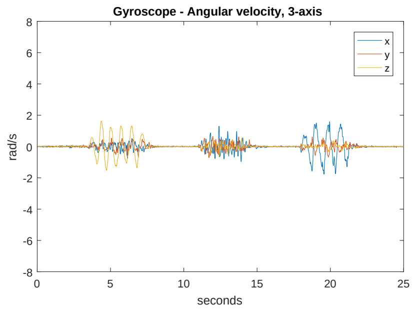
Figure 4.1: Movements of an object has six degrees of freedom, three linear and three rotational.

The accelerometer measures linear movements while the gyroscope data represent rotational movements. Examples of raw three axis sensor data from both these sensors are shown in figure 4.2 and 4.3 for different types of movements.

Figure 4.2 presents the output data when the device is shaken along one axis at the time, first in x direction, then in y-direction and last in z-direction of the smartphone's coordinate system. By the acceleration data, figure 4.2a, it is possible to determine the direction of the shake. That is not the case when studying the angular velocity in figure 4.2b, even though the movements are visible in these signals as well.



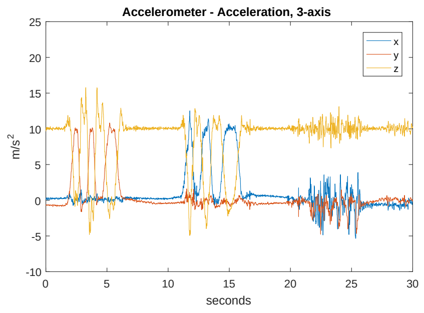
(a) Acceleration



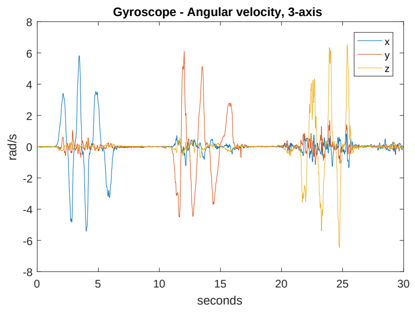
(b) Angular velocity

Figure 4.2: Sensor data when moving the device linearly in one direction at the time.

Figure 4.3 shows the signals of the sensors when instead rotating the device 90° back and forth around each axis. Now these movements are clearly visible by studying the gyroscope data, figure 4.3b. Of course, these movements cause accelerations of the device as well, shown in figure 4.3a, but it is not easy to interpret how these rotations were made.



(a) Acceleration



(b) Angular velocity

Figure 4.3: Sensor data when rotating the device back and forth around each axis.

An other thing worth to mention is how the earth's gravity clearly affects the accelerometer data. Since the screen was parallel to the ground in initial position, the z-axis data has an offset corresponding to the gravity constant g ($\sim 9.82\ m/s^2$). If the initial position would have been different, the gravity would have been affecting an other axes instead.

The accelerometer is most used in studies of elderly fall detection, and sometimes combined with the gyroscope data [25]. The outcome of different hardware sensor signals for activity recognition have been evaluated, the result was that both the accelerometer and the gyroscope are capable of taking the lead roll individually [47]. The choice of using both these sensors in this project was made.

The magnetometer is seldom used in elderly fall detection algorithms [25]. Some of the normal riding activity recordings in this project were made with the smartphone in a phone case having a magnetic lock. This magnet destroyed the magnetometer data in these recordings. The choice was hence to exclude the magnetometer data from further use.

The orientation sensor derives its data by processing the raw sensor data from the accelerometer and the magnetometer. When the magnetic field sensor is deceptive, that yields for the orientation sensor as well. Even when the magnetic field signal is correctly recorded, the accuracy and precision of the orientation sensor is limited, due to the heavy processing that is required. The orientation sensor is therefore no longer recommended by Android Developers to use. Instead they suggest to use a provided rotation matrix and own calculations [37]. This matrix was not saved during data recording in this project and the decision was made to exclude the orientation sensor from further use.

Henceforth the acceleration and angular velocity was studied together with the use of the software sensors linear acceleration and gravity when it was motivated to use these. It is preferable to use the raw hardware sensor data as far as possible. The software data is processed in a way controlled by Android. It is difficult to totally understand what is happening and if there is any differences when making an application for e.g. iOS later on.

4.2 One dimensional representation of signals

From the raw three axis data it is possible to make some calculations to present the data in other manners. The appearance of the three axis data depends on the orientation of the device relative to the earth's coordinate system. The goal was to construct an algorithm that works independently of this orientation. Additional signal properties were derived to reduce the amount of data to later perform feature extractions on.

4.2.1 Magnitude of signals

The magnitude of a vector in an orthonormal coordinate system is calculated using equation 4.1. The result gives the length of the vector and it is always positive, if not zero [48].

$$|M| = \sqrt{x^2 + y^2 + z^2} \tag{4.1}$$

The magnitude of the three-axis acceleration (A_M) in x, y and z direction of the device (A_x, A_y, A_z) in one point of time is calculated to get the total acceleration according to equation 4.2

$$|A_M| = \sqrt{A_x^2 + A_y^2 + A_z^2} \tag{4.2}$$

The magnitude of the three axis angular velocity (G_M), around the x, y and z axis of the device (G_x, G_y, G_z), is defined by equation 4.3.

$$|G_M| = \sqrt{G_x^2 + G_y^2 + G_z^2} \tag{4.3}$$

An advantage of the accelerometer and gyroscope magnitude data is the independence of the orientation of the device.

4.2.2 Vertical acceleration

The accelerometer data can be decreased by deriving the acceleration in the vertical direction only, i.e. in the z-direction of the earth's coordinate system. Two ways to perform these calculations are found in the literature. Both of them are here presented, even tough only the latter described one is used in this project.

The first approach is to calculate the contribution of data from each accelerometer axis to the total vertical acceleration, and then add them together. The orientation sensor provides the angles revealing information about the device's position relative to the earth's coordinate system. As mentioned in section 4.1, this sensor is not recommended to use, but there are other ways to calculate the angles azimuth (θ_z), roll (θ_y), and pitch (θ_x) [49]. The absolute vertical acceleration (A_V) is then calculated according to equation 4.4 [50].

$$|A_V| = |A_x sin\theta_z + A_y sin\theta_y - A_z cos\theta_y cos\theta_z| \tag{4.4}$$

The accurate information about these angles was not included during data recording. Hence this method is not implementable for the recorded data and the following method is used instead. A vector can be projected on an other vector by use of the scalar product [48]. The software sensor gravity gives the vertical direction. By projecting the three axis acceleration on the three axis gravity vector the vertical acceleration is obtained.

It is possible to use both the raw acceleration and the linear acceleration produced by Android Studio for this calculation. The choice in this project was to use the linear acceleration. The result is hence the vertical linear acceleration. The gravity data and the linear acceleration data seems to be derived form the same point of time. The raw acceleration data seems to be derived from an earlier point of time. The probable explanation is that no calculations needs to be performed for this signal. The vertical linear acceleration (A_{VL}) is calculated by equation 4.5, using the linear acceleration (L) in all directions (L_x, L_y, L_z) and gravity (g) in all directions (g_x, g_y, g_z) [45].

$$A_{VL} = \frac{L \cdot g}{|g|} = \frac{L_x g_x + L_y g_y + L_z g_z}{\sqrt{g_x^2 + g_y^2 + g_z^2}} \tag{4.5}$$

The vertical linear acceleration shown in red in figure 4.4, to be compared to the linear acceleration magnitude shown in blue. The origin of data is the same as in figure 4.2 where the device is first shaken along the x-axis, then in the y-direction and last in the z-direction. The z-direction is here the same as the vertical direction and as supposed the red signal has a larger amplitude for this shake movements.

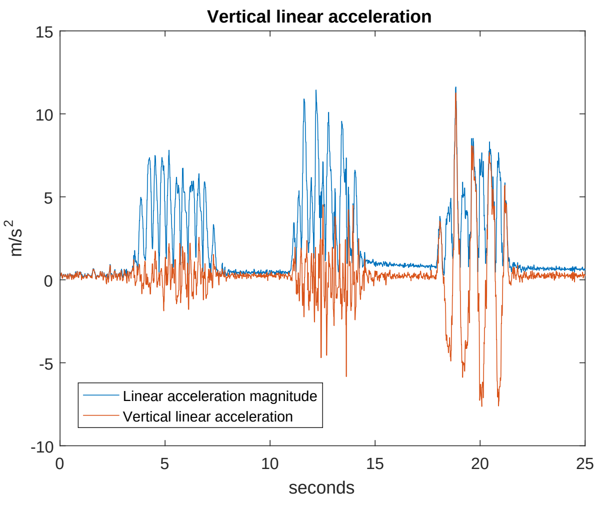


Figure 4.4: The vertical linear acceleration to be compared to the linear acceleration magnitude when the device is moving linearly in each of the three directions. As expected the movements along the z-axis affects the vertical acceleration the most.

4.3 Sample rate and preprocessing

The used sample rate during data recording was 100 Hz, less than the maximum rate of the used sensors. The fear of using too low sample rate is to miss narrow peaks representing important information and obtain false frequencies due to aliasing.

If it would have been possible to record and save data with the maximum sample rate, it would have been interesting to study the frequency spectra and seek the highest existing frequency component. The choice, following the Nyquist theorem, would then have been to choose at least twice the highest frequency component as sample rate. It would have been a computational advantage to use a lower sample rate, but due to fast decelerations during falls the choice was to keep 100 Hz as sample rate further on.

To make sure that no filters are applied to the sensor output provided by the application AndorSensor, the creator of the application was contacted. When saving CSV-files with a rate of 100 Hz it is the current value provided of each sensor that is taken, no filters are applied. One method to evaluate if the noise needs to be reduced is to calculate the signal-to-noise (SNR) ratio. SNR is often presented in decibel (dB) it can be calculated using equation 4.6. P is the average power, calculated by equation 4.7, for the noise and the signal respectively.

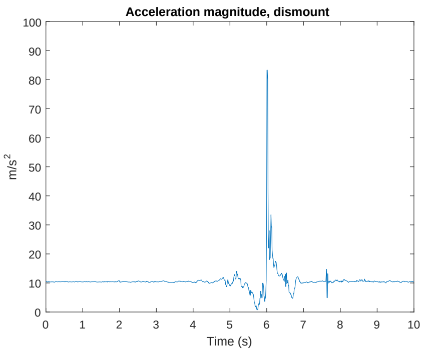
$$\mathrm{SNR} = 10\log_{10}\left(\frac{\mathrm{P}_{Signal}}{\mathrm{P}_{Noise}}\right) \tag{4.6}$$

$$\mathrm{P} = \frac{1}{n}\sum_{i}^{n} x_i^2 \tag{4.7}$$

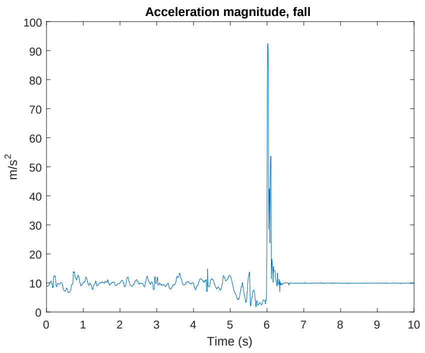
The noise of the signals provided by the accelerometer and gyroscope was estimated by recording the output when the smartphone was placed on a table. The SNR value differs dependent on the characteristics of the recoded movements. It was calculated for both some normal riding activities and simulated fall events. The magnitude data was used and the signals were made zero-mean before computing the SNR. The value varied between 30 - 60 dB. The large SNR values told that there was no need to remove the noise by filtering. The conclusion was to not apply any filters when handling the data.

4.4 Visual studies of data

Both raw three axis sensor data and the one dimensional representations were continuously analyzed i MATLAB after recordings. Figures of extra interesting data, such as all three gaits, jumps, dismounts and different types of falls, were saved to give an overview of the problem to detect equestrian falls. Examples of both normal riding activity and fall situation data are presented in Appendix A, including all three one dimensional signals. An example presenting the difficulties of algorithm development is presented in this section. Figure 4.5 shows the acceleration magnitude when dismounting the horse to the left, figure 4.5a. The corresponding signal when falling from the horse is shown in the right figure 4.5b.



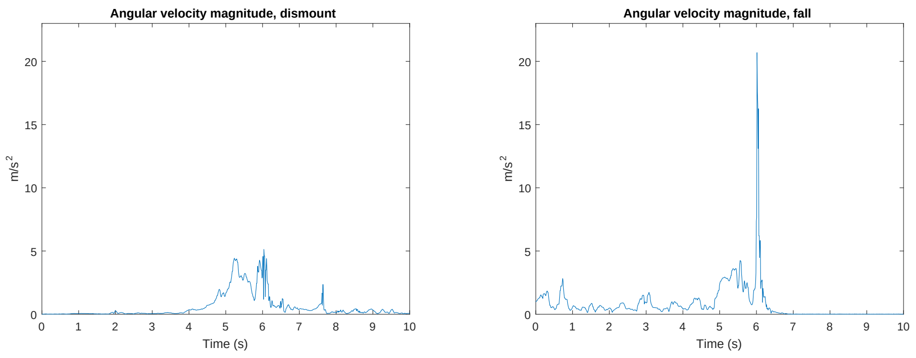
(a) Acceleration magnitude when dismounting the horse



(b) Acceleration magnitude when falling from the horse.

Figure 4.5: Acceleration magnitude of both a dismount of the horse and a fall. They seem very similar when only studying this data.

The acceleration magnitude signals seemed to be very similar. This similarity appeared also when comparing the vertical linear acceleration for the two events, so figures of the vertical linear acceleration are not presented here. But if the angular velocity magnitude is studied, more differences are found. This signal for the same episode of time and for both events are presented in figure 4.6. The peak is much higher during the fall than during dismounting the horse.



(a) Angular velocity magnitude during dismount of horse.

(b) Angular velocity magnitude during a simulated fall.

Figure 4.6: The angular velocity magnitude of both a dismount of the horse and a fall. These signals differs more than when studying the acceleration magnitude data.

The accessibility to several different signals, with large variations within each class, made it difficult to start designing the algorithm. A methodical way of work was needed, revealing the contribution of interesting information from each signal.

4.5 Method of algorithm development

The task of determine if a fall has happened or not can be solved in several ways, using different approaches and techniques. Most elderly fall detection systems uses fixed threshold values [25]. Since detecting falls from a horse is a binary classification problem for real time purposes, this approach was also used here.

A fall generates a deceleration of the rider when he or she hits the ground after being thrown from a horse. This deceleration affects the accelerometer of the smartphone placed on the rider and the recorded large peak values have been used in both elderly and equestrian fall detection [25] [20]. The initial choice was to use a threshold value of 40 m/s^2 for the acceleration magnitude to have a possible fall.

Exceedings of this threshold value were detected in all normal riding activity data. By not allowing additional detections in 30 samples (0.3 s) after one detection the problem of detecting one peak several times was avoided. The length of this time interval was determined after studying the regularity of different gaits. Trot was the fastest, having a cycle time of about 0.4 s. An interval shorter than this was wanted.

Data surrounding the time points for these detections were saved separately. A window of 599 samples before and 300 samples after the exceeded threshold value was saved for all three one dimensional signals. This method reduced the total amount of normal riding activity to 9618 events, represented by 900 samples of all three signals.

All fall events had an acceleration magnitude far above this threshold. Also the fall data was saved in the same manner separately, resulting in 41 events represented by 1000 samples of all three signals. The classification problem was now reduced to building an algorithm able to separate these saved windows.

4.5.1 Limited battery time and computational power

The smartphone has limited computational power and the balance between efficiency and avoiding heavy calculations must be kept in mind when constructing the algorithm. Beyond this, the battery consumption increases when the amount of active sensors is increasing [47]. The algorithm needs to be relatively simple to be runnable on a smartphone device. An advantage compared to elderly fall detection is that the future application will be used during a shorter period of time. A normal ride goes on for about 0.5-2 hours while elderly fall detection systems needs to be continuously running. To use several sensors if necessary is therefore not a problem, but it is preferred to use as few sensors and as less computational power as possible to not drain the user's smartphone battery totally.

4.6 Feature calculations

Several features of the extracted signals were calculated to get single values representing the characteristics of the signals surrounding the acceleration magnitude peak. This made it possible to later set fixed thresholds when constructing the algorithm.

Many different features, both from relevant literature and own ideas, were tried. Not all will be presented in this report. As an example, the correlation coefficient between data from before and after the trigger event was calculated. The frequency spectra of different parts of the signals was also derived and analyzed. The conclusion of these studies was that the result was not good enough to motivate the use of these more complex calculations.

The features were calculated using different parts of the event windows with 900 samples each. The different windows are presented in figure 4.7. The first set of feature calculations was made on data surrounding the trigger event, shown in yellow in the figure. It is a narrow window of 0.6 s in total, evenly distributed with 29 samples before the trigger event and 30 after. This length allow peaks in the normal riding data to be detected individually. In addition, this number of samples allows a possibility of 50 % overlap of peak windows when having two trigger events as close as possible to each other.

Some features were calculated using data captured both before and after a trigger event. Data from the ongoing ride, both before an assumed fall and when landed on the ground, was of interest in the feature calculations. But data from the time periods when the rider is unbalanced or ending the fall movements is not of interest. Hence was 2 s before the trigger event and 1 s after the trigger event excluded. The lengths of these time intervals were determined after studying several movies of equestrian falls found on the Internet. The time of being unbalanced is in general longer than in some elderly fall detection algorithms

[45]. The green area of figure 4.7 represents the time period when the ride is still considered as ongoing, 4 s is used for this purpose. The gray areas are the time periods when the rider might be unbalanced or ending the fall movements.

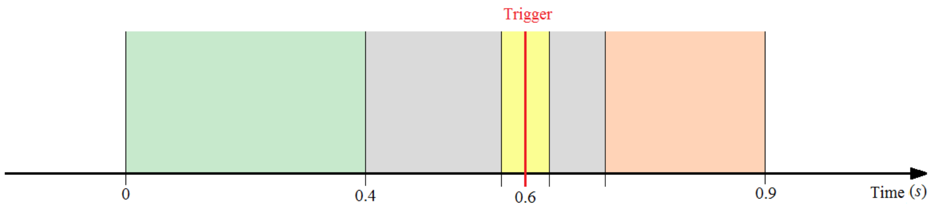


Figure 4.7: Illustration of windows surrounding a detected trigger event. The yellow area represent the narrow widow used when analyzing the actual peak. The green area corresponds to assumed normal riding activity and the red area may represent when the rider is lying on the ground after a fall. This is the time period when the rider might be unbalanced or ending the fall movements.

4.6.1 Peak features

The features calculated on the narrow window surrounding the trigger event was derived using simple time-domain methods on all three one dimensional signals. This to further study the signal appearance close to a peak of acceleration magnitude and to fast get additional information about the event. The calculated features were:

- Maximum value
- Minimum value
- Mean
- Median
- Standard deviation
- Root mean square (RMS)

4.6.2 Post trigger event features

If the trigger event actually captures a true fall, the appearance of the sensor signals was assumed to differ from normal horseback riding activities, both if the rider is still on the ground or being dragged after the horse. The problem of detecting lack of movements have been solved using the software speed sensor and the GPS signal [20]. In this project an attempt to use the already handled sensor data was made.

For simplicity, the same features calculated for the peak window was used also for this purpose. They were calculated on the data sampled in the red window in figure 4.7, corresponding to a time period of 2 s after the trigger event.

4.6.3 Change of orientation

If a rider is exposed to a fall, a significant difference between the initial and final position of the device is expected, including a rotational change. For example, a smartphone is placed in the pocket of the rider's jacket in an upright position. If the device can not change position inside the pocket and a fall occurs, the only way the device can be in the same direction afterwards is if the rider stands or sits straight up on the ground. If he or she lies down, an orientation change of the device in comparison to before the fall has occurred.

The angle, α, between two vectors, a and b, can be derived using the definition of the scalar product, according to equation 4.8 [48].

$$a \cdot b = |a|\ |b|\ cos\ \alpha \implies cos\ \alpha = \frac{a \cdot b}{|a|\ |b|} \tag{4.8}$$

For this purpose the raw three axis accelerometer data was used, not the magnitude. The mean acceleration in each direction (x, y and z) before (A_B) and after (A_A) a trigger event was calculated. The green and red windows in figure 4.7 were used respectively for this purpose. The choice of using a larger amount of samples before the trigger event was based on the assumption of larger changes of acceleration during riding than after a fall.

$$\begin{aligned} a &= A_B = (\bar{A}_{Bx}, \bar{A}_{By}, \bar{A}_{Bz}) \\ b &= A_A = (\bar{A}_{Ax}, \bar{A}_{Ay}, \bar{A}_{Az}) \end{aligned}$$

An estimation of the orientation change (OC) in degrees is then given by equation 4.9 [45].

$$OC = \frac{180}{\pi}\left[cos^{-1}\left(\frac{A_B \cdot A_A}{|A_B|\ |A_A|}\right)\right] \tag{4.9}$$

Since the vectors are orthonormal with the bases x, y and z the change of orientation can also be expressed by equation 4.10.

$$OC = \frac{180}{\pi}\left[cos^{-1}\left(\frac{\bar{A}_{Bx}\bar{A}_{Ax} + \bar{A}_{By}\bar{A}_{Ay} + \bar{A}_{Bz}\bar{A}_{Az}}{\sqrt{\bar{A}_{Bx}^2 + \bar{A}_{By}^2 + \bar{A}_{Bz}^2}\sqrt{\bar{A}_{Ax}^2 + \bar{A}_{Ay}^2 + \bar{A}_{Az}^2}}\right)\right] \tag{4.10}$$

4.7 Selection of features and threshold values

All features presented in the last section were calculated using their corresponding time windows, for both normal riding activity and simulated fall events respectively. Histograms were used to evaluate all features possible contributing to the algorithm. The feature values of both classes were presented in the same histogram, but in different colors. Blue bins represented normal riding activity while red bins represented simulated fall events. The more separated the blue and red bins were, the more coveted was the feature to use when building the algorithm.

All histograms of features are shown in Appendix B, and the ones of special interest are presented in this section. Worth to notice in the following figures is that the plots shows the probability of a value and not the actual number of corresponding events. The blue bins represent a total of 9618 values while the red bins corresponds to 41 values. Even though the blue bins are really small, they contain more events than a red bin of equal size. There might be a small bin of normal riding events hidden among the fall events, barely visible.

The algorithm was constructed using fixed thresholds. The choice of threshold values was also obtained using the histograms. It was seen as equally important to catch every fall event as it was to avoid false alarms. The reasoning when choosing the threshold values differed dependent on how the feature was used in the algorithm. The overall goal was to build an algorithm that performs well, not only on the recorded data but also in new situations. It was hence important to not adapt the threshold too much to the recorded data set.

4.7.1 Trigger threshold

The decision to use the magnitude of acceleration as a trigger feature has already been presented. The remaining question was how to set the threshold in the final algorithm. Figure 4.8 shows the histogram of the maximum acceleration magnitude for the peak windows of every registered event.

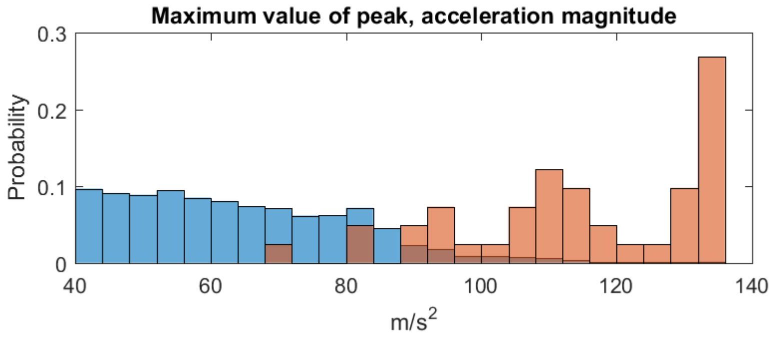


Figure 4.8: The maximum value of acceleration magnitude in the peak window. Blue bins represent normal riding activity and red bins represent simulated fall situations.

Since the threshold for detecting an event was set to 40 m/s^2, obviously all values in the histogram was above that value. The maximum value of the acceleration magnitude was widely spread both for normal riding events and for simulated falls. In this first step of the algorithm it was important to include all fall events. The falsely detected normal riding events can be neglected later.

Only one smartphone was used for data recording with the risk of making the algorithm too adapted to the sensors of this device. Of course it was preferable to build an algorithm workable on other devices as well. Some smartphones have a lower range of the accelerometer, for example up to 40 m/s^2, equal to the maximum magnitude of 60 m/s^2. The threshold needs to be lower than that value for falls to be detected on such devices. The final choice was to use 50 m/s^2 as a trigger threshold value for the algorithm. The number of normal riding events was then decreased from 9618 to 7400, all fall events was still captured.

4.7.2 Peak features

It was desirable to reveal as much interesting information as possible in the small peak window, to quickly reject more false events. As when setting the trigger threshold it was still important to catch all fall events with some margin in this early step of the algorithm. As a rule of thumb the minimum feature value of the fall events minus the standard deviation of the fall event feature values was chosen as threshold for the peak features.

By continuing to study the acceleration magnitude peak features, the standard deviation seemed to be of most interest. The histogram of this feature is presented in figure 4.9. Even though there is a clearly visible overlap of values representing normal riding activity and fall events, the expected values differ and some falsely detected events can be excluded. The threshold was chosen to be 12 m/s^2.

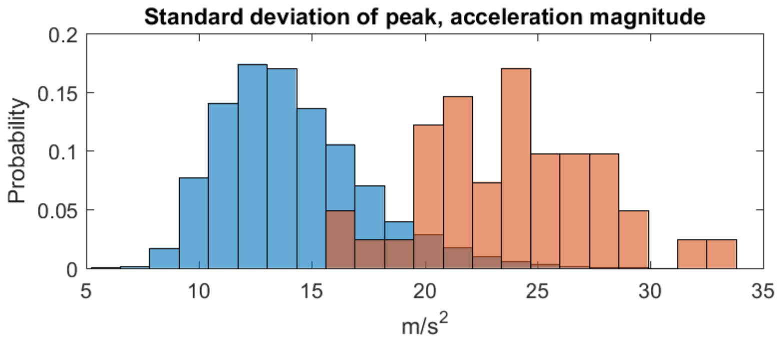


Figure 4.9: The standard deviation of acceleration magnitude in the peak window. Blue bins represents normal riding activity and red bins represents simulated fall situations.

No more features representing acceleration magnitude was further evaluated. The guess was that the difference between the two groups in these features was largely covered by the use of the maximum value and standard deviation. Also all features representing vertical linear acceleration of the peek was excluded from further evaluation. They did not seem to add any important information.

The features representing angular velocity magnitude in the peak window provide information about rotational movements and can contribute with additional information about the event. There were three features of the gyroscope data that seemed to be interesting for further studies. These were the maximum value, the mean and the RMS, presented individually in figure 4.10.

The feature representing the maximum value was widely spread for the fall events. The assumption was made that the other two features, representing the mean value and the RMS, was preferable to use further on. The threshold of the mean was set to 2.2 rad/s and the threshold of the RMS-value was set to 2.8 rad/s.

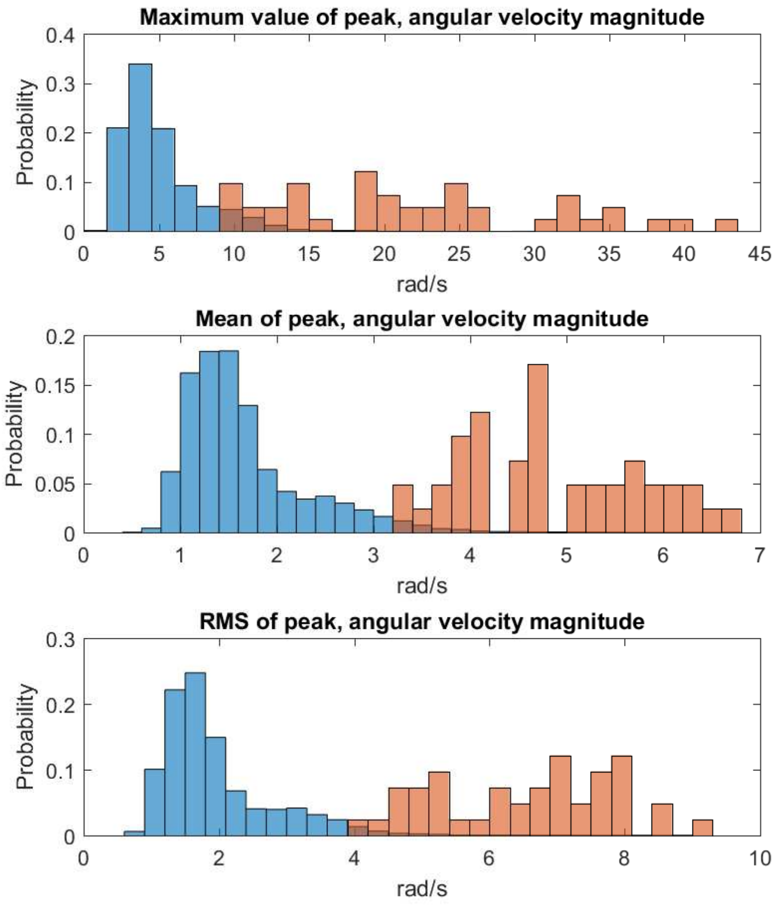


Figure 4.10: The maximum value, mean and RMS of the angular velocity magnitude in the peak window, respectively. The blue bins represent normal riding activity and the red bins represent simulated fall situations.

Combining the raised trigger threshold with the different peak features gave the result presented in table 4.1. All true fall events was still detected. One disadvantage of the method of histograms was that it is impossible to follow one specific trigger event trough all different feature values. For example, it would have been interesting to know if the fall with highest acceleration magnitude also had the highest angular velocity magnitude. Although, the table helped choosing the combination of features to use.

Feature	Nbr of false positive remaining
Only std of A_M ($12\ m/s^2$)	6491
Only mean of G_M ($2.2\ rad/s$)	1281
Only RMS of G_M ($2.8\ rad/s$)	1175
Std of A_M and mean of G_M combined	1045
Std of A_M and RMS of G_M combined	968
Mean of G_M and RMS of G_M combined	1132
All three selected peak features combined	930

Table 4.1: The total number of included normal riding events when having the acceleration magnitude threshold at 50 m/s^2 was 7400. Adding one or a combination of peak features gave the result presented in this table.

The table shows that the best options were to use either all three peak features or the combination of standard deviation of the acceleration magnitude and the RMS-value of the angular velocity magnitude. Due to the simplicity of the feature calculations the choice was to use all three features further on. Since the result when only using the peak features was insufficient, more features had to be added. This was preferable to tighten the thresholds, in order to avoid making the algorithm too adapted to the recorded data set.

4.7.3 Change of orientation

The orientation change feature (OC) calculates the angle between the mean position of the device before and after a trigger event. These feature values are presented in figure 4.11. There are some overlapping values between normal riding activity and fall events, even though it is barely visible in this figure.

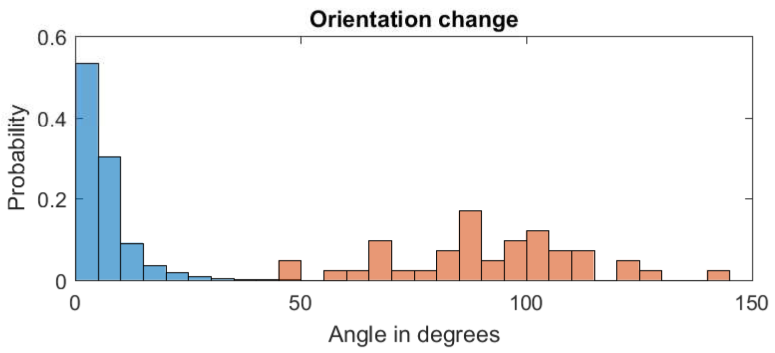


Figure 4.11: The angle of orientation change during a trigger event. The blue bins represent normal riding activity and the red bins represent simulated fall situations. The separation of the colors reveals that the orientation change is a desirable feature.

An advantage of this feature is that it is easy to interpret. By studying the histogram it was chosen to use two thresholds for this feature. If the orientation change is below 30° the event is rejected as a normal riding event and no further calculations will be done. If the orientation change is above 75° the event is definitely assumed to be a fall without need of further calculations. If the feature value is between 30° and 75° some more calculations needs to be done before it is possible to decide whether it is an equestrian fall or not.

This feature leaved 38 normal riding activities and 9 fall events for further evaluation. 32 falls was correctly classified after use of this feature in combination with the peak features. No false positives was obtained.

4.7.4 Post trigger event features

The rider often becomes still after a severe fall accident. Even tough it is not always the case, it is important for the algorithm to not miss those cases. It can be a sign of unconsciousness or other severe states of the rider. The lack of movements can bee seen for several features calculated on the red window of time in figure 4.7. The mean of angular velocity and standard deviation of the acceleration seemed to be most interesting for this purpose, see figure 4.12.

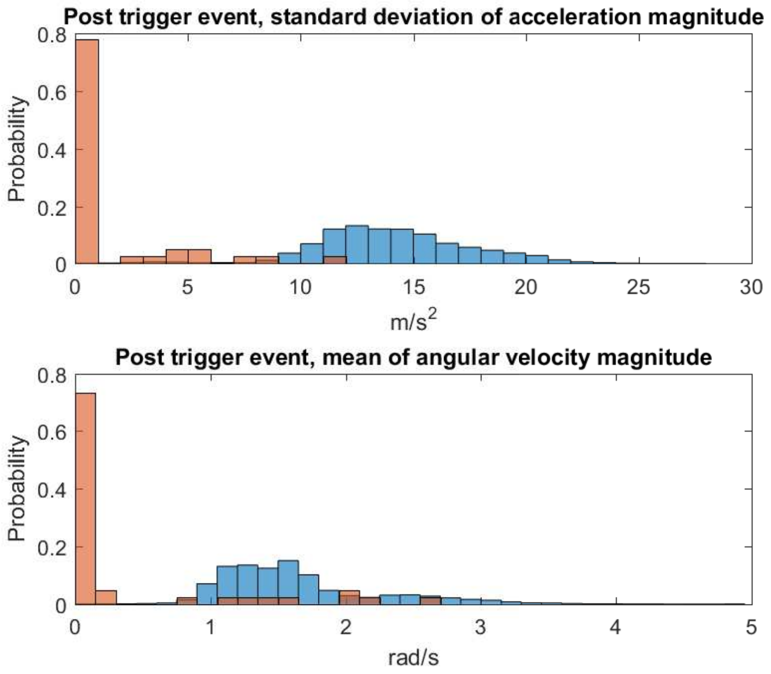


Figure 4.12: The standard deviation of acceleration magnitude and the mean of angular velocity magnitude, of the time period after a trigger event. The blue bins represent normal riding activity and the red bins represent simulated fall situations.

The features are derived from different sensors, but seemed to be able to distinguish exactly the same fall events. The choice was to only use one of these features. One difference between the human body model and a real human when being still is the breathing. After a fall the model is totally still, if not affected by outer forces. Hopefully a real person is still breathing, causing small movements of the device. To decide which feature to use and a suitable threshold an additional recording was made. The smartphone was placed on a person lying still on the ground breathing. The gyroscope data was more stable and therefore it was chosen to use. The threshold was set to 0.3 rad/s. The reason for this relatively low threshold, not including all fall events, was the risk of detecting false positives. By this feature no events are rejected, it is only added as a possible way to detect falls. After this feature 2 falls remained undetected among still 38 normal riding events.

An other interesting feature was the median of vertical linear acceleration after a trigger event, shown in figure 4.13. As seen in the figure, the values representing fall events are centered around zero but more spread out for normal riding activity.

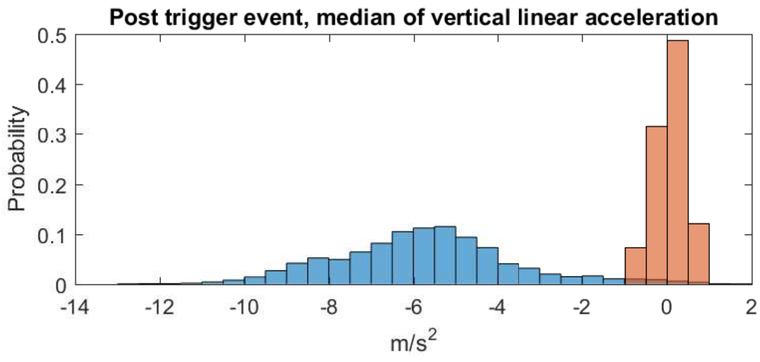


Figure 4.13: The median of vertical linear acceleration during the time period after a trigger event. The blue bins represent normal riding activity and the red bins represent simulated fall situations.

As seen when looking closely into the figure some blue bins are visible behind the red ones, i.e, it is not only true fall events having these feature values. Since the algorithm consists of several earlier steps, this feature is added as one last chance to decide whether it is a fall or not. The absolute value of the median vertical linear acceleration hade to be less than 0.5 m/s^2 to be detected as a fall, otherwise it was rejected as normal riding activity. Together with all previous steps this last feature made all events correctly classified.

4.8 Final equestrian fall detection algorithm

The work of evaluating features gave the result of this master thesis, the final equestrian fall detection algorithm. It is illustrated in figure 4.14 where the blocks represents different features or feature sets, presented i the last section. The arrows shows possible outcomes dependent on the calculated feature values. The thresholds of the different steps are presented in table 4.3.

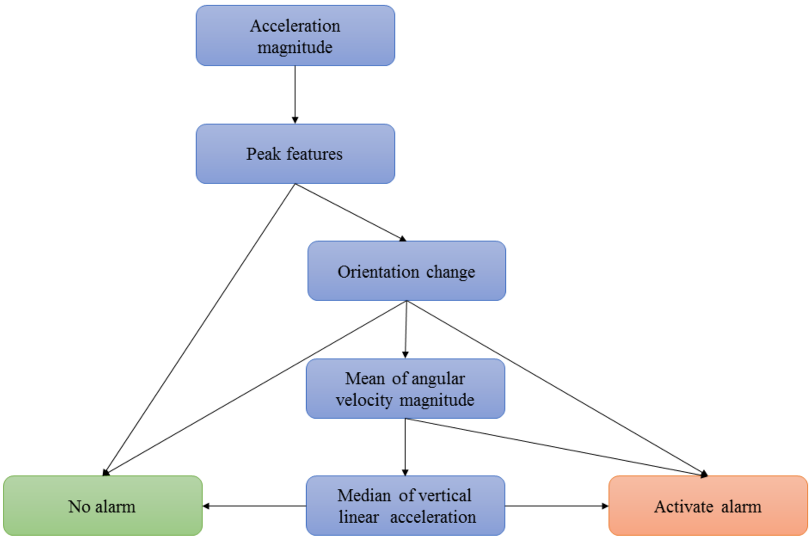


Figure 4.14: Illustration of the equestrian fall detection algorithm. There are several possibilities to both reject and detect events as falls.

Feature	Threshold value
Acceleration magnitude	$> 50\ m/s^2$
Peak features	
Standard deviation of acceleration magnitude	$> 12\ m/s^2$
Mean of angular velocity magnitude	$> 2.2\ rad/s$
RMS of angular velocity magnitude	$> 2.8\ rad/s$
Orientation change	$> 30^\circ\ \&\ > 75^\circ$
Post trigger event features	
Mean of angular velocity magnitude	$< 0.3\ rad/s$
Median of absolute vertical linear acceleration	$< 0.5\ m/s^2$

Table 4.2: Fixed threshold values in the final equestrian fall detection algorithm.

5 | Java Implementation

This section describes the Java implementation of the algorithm presented in the last section of chapter 4. The result is an application for Android devices, making real time tests of the algorithm possible.

5.1 Method of work

The smartphone application was implemented using Android Studio. Small steps of implementation was made early on in the project process. This to get a hint of how the sensors and sensor data are handled when building an Android application. The heaviest part of the implementation was made as soon as the the overall structure of the algorithm was known. It was to build a framework saving sensor data where the final algorithm later could be implemented. My knowledge in how to develop applications was very limited before starting up this project. Most problems was solved by much effort and most knowledge was gained by searching on the Internet.

5.2 Handling of real time sensor data

Arrays of data were needed to be able to perform feature calculations. The handling of real time sensor data was one of the largest challenges compared to developing the algorithm in MATLAB. The limited memory capacity had to be in mind during this process.

It is possible to set the rate of output data from each sensor, but different sensors do not release their data values at exactly the same point of time. Hence the choice was on using some type of sampler to get sensor values at the speed of 100 Hz. After some testing of different suitable Java objects the final choice was to use a Timer Task, see [51], with an extended run method for this purpose. All used sensors produced data as fast as possible, and every 10:th ms the Timer Task grabbed the current value from each sensor of interest.

The need of stored values differs for various sensors, e.g. all three axis data needs to be stored for the accelerometer, linear acceleration and gravity while only the calculated magnitude needs to be stored for the gyroscope data. The amount of values also differ between the various sensors dependent on how their data is used in the algorithm. To not burden the memory of the device more than necessary, queues were used to save only the latest and maybe needed sensor values. Values, no longer of interest, are deleted and instead new values are added to the queues. Table 5.1 shows the sensor data stored by the Timer Task and the length of the corresponding queues.

Sensor data	Size of queue
Acceleration magnitude for peak calc.	60
Angular velocity magnitude for peak calc.	60
Acceleration, three axis	900
Vertical linear acceleration, three axis	270
Gravity, three axis	270
Angular velocity magnitude for final calc.	270

Table 5.1: Sizes of different queues for storing data captured by the sensors of the smartphone.

When a data set is needed the current values in the queues could be transformed into Array Lists, possible to perform calculations on.

When the orientation change calculations are made, data from six seconds before the trigger event is needed. Hence is the application unable to register a fall during the first six seconds after the application is started. Also, data from three seconds after a possible fall is needed for the final calculations, making a delay of three seconds for the alarm activities to be activated after the trigger event.

5.3 User interface

The application made within this project is made only for testing the algorithm. Hence is the effort on the user interface (UI) minimized. All that is needed for testing the algorithm is a button to start and stop the sensors and the calculations. It is also important for the user to know if the algorithm is running and to be notified when a fall is detected. The three states of the user interface is shown in figure 5.1. The left image presents the UI when the application is just opened or after the algorithm is stopped manually. The image in the middle is the UI when the algorithm is running and the last image is when a fall event has been detected.

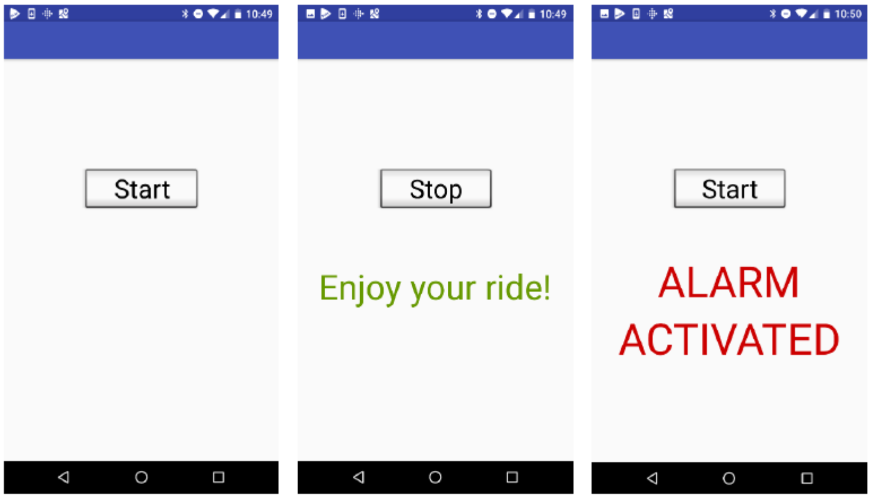


Figure 5.1: The user interface of the application during the three different states of the application. The left print screen is when the application is just opened or when the user has pressed ”stop”. The middle image is when the algorithm is running and the right one is when a fall has been detected.

5.4 Structure of application

When making the structure of the implementation, one big question was how to initiate different steps of the algorithm. Except an exceeded threshold, there is a need to wait for more data before a following step of the algorithm is activated. After testing some different methods, e.g. to use an observer object, the choice was to actually make these kinds of decisions inside the already existing Timer Task object.

After exceeding a trigger threshold, data for another 30 samples must be collected before the peak features can be calculated. To avoid several triggers for one possible fall event the algorithm can not detect additional trigger events during these 0.3 seconds of time, similar to the earlier MATLAB implementation. When the criteria of an exceeded trigger threshold together with having enough data is fulfilled, the peak features presented in previous chapter is calculated.

If the trigger event is still considered to be a possible fall event the sensors must collect 270 additional values before the next calculations can be done. Unlike the earlier waiting situation the algorithm must be able to capture new trigger events during these 2.7 seconds of time. The corresponding data for these later fall events must also be captured and waited for in parallel processes. This problem could be solved using different threads for the different trigger events but due to my limited programming skills and fear of nasty bugs another method was chosen.

The sensor data needed for these final calculations is always stored in queues and available to grab as presented in chapter 4. Every time the Timer Task is activated, i.e. every 10:th ms, an integer count value is incremented by one. When the peak calculations finds a possible fall the current count value is added to an additional queue. Count values corresponding to later possible falls are also added to this queue. When the queue is not empty the difference between the constant growing count value and the first value in the queue is calculated. When it reaches 270 all needed data for the final calculations of the first trigger event are available and these calculations are performed. This first stored count value is deleted from the queue, meaning that if the queue has additional values stored the comparison of time difference is now made on the secondly registered possible fall and so on. This, together with the rest of the Java implementation is illustrated in figure 5.2.

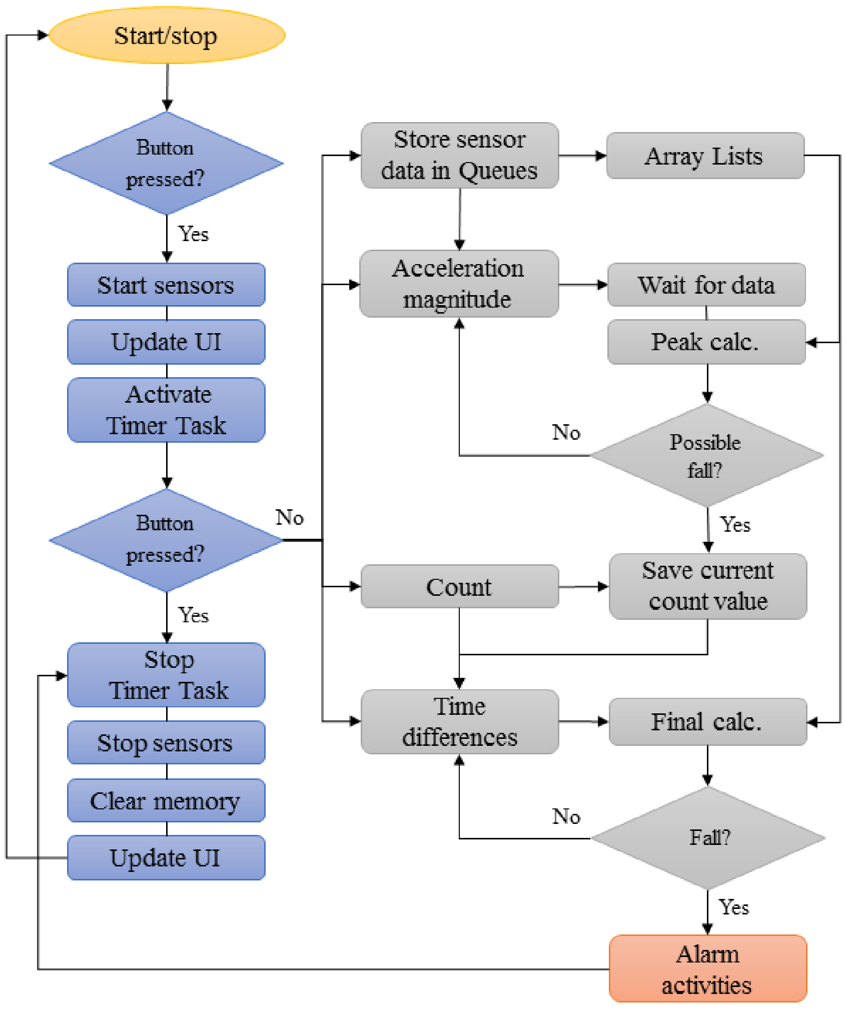


Figure 5.2: Overview of the Java implementation. After the application is started the algorithm is running until an event is classified as a fall, or as long as the process is not manually stopped.

When a user presses the start button both the sensors and the Timer Task starts to capture sensor data. The user interface is updated to let the user know that the algorithm is running. The algorithm is running as long as the button is not pressed to manually stop the fall detection process. Except from capturing sensor data the Timer Task decides when to start both the peak and final calculations. The alarm activities is activated if an event is classified as a fall. In this first test version of the application this only means that the UI is updated and the sensors and Timer Task is stopped in the same way as when done manually by pressing the button. Also a beep to notify the test rider that the alarm has been activated is produced. More alarm activities can easily be added later in this stage of the implementation.

6 | Evaluation

This chapter describes the work of algorithm evaluation, done using the application presented in last chapter.

6.1 Evaluation methods

The developed algorithm was able to classify all recorded training data correctly, but the accuracy on new data was also of interest. The Android application described in last chapter was used to test the algorithm. It would have been possible to perform an evaluation by recording new data in the same manner as the training data, then test the algorithm on it in MATLAB. But by using the application instead some advantages were obtained. The most important was the feeling of "holding the algorithm", e.g if it was possible to activate the alarm by different non fall movements. It is pretty simple to shake the device heavily to activate the alarm, especially when knowing how the algorithm works. Although this is the case, the alarm is not activated when handling the smartphone in normal manners, like putting it in a pocket and so on. An other advantage was that the calculations were ensured to be able to be handled by a smartphone device. Also a perception of the battery consumption of the algorithm was obtained, presented last in this chapter.

To further evaluate the algorithm, it had to be tested on data not used for the development process. The amount of tests on both normal riding activity and simulated falls was to small too obtain a strong statement of the performance. The appearance of both falls and normal riding activities has shown to be widely spread, contributing to the need of more evaluation to be done.

6.2 Normal riding activity

The application was tested on normal riding activities in a similar way as when recording the data. Four different riders with different horses rode with the application running on a smartphone in their pocket. The approximate recording durations and results are presented in table 6.1. The first three riders were equipped with the smartphone used during the development process. The forth rider installed the application on her own Android device. Rider 1 is the same rider and horse as used for most training data recordings, i.e. myself.

Totally 340 minutes of riding activities was made, including mounting and dismounting the horse. The result was great, not a single fall was detected during this time, even though new horses and riders were used.

Rider	Approx. duration (min)	Nbr of detected falls
Rider 1, dressage	2*45	0
Rider 1, jump session	75	0
Rider 2, dressage	50	0
Rider 3, dressage	70	0
Rider 4, dressage, new smartphone	55	0
Total:	340	0

Table 6.1: The evaluation result on normal riding activity for real time sensor data using the implemented application.

6.3 Fall simulations

The algorithm was tested on 14 new fall events using the application. The same method as during most recordings of training data was followed, using the human body model on my shoulders and no horse. Only my own LG smartphone was used for the fall evaluation. The result on different types of falls is presented in table 6.2. One new type of fall was added, simulating a horse landing on or stamp the rider after a fall.

Type of Fall	Nbr of simulations	Nbr of detected falls
Forward	2	2
Backward	2	1
Sideways	2	2
Somersault forward	2	2
Fall in fence	2	2
Being dragged afterwards	2	1
Being stamped on after the fall	2	2
Total:	14	12

Table 6.2: The result on simulated falls when evaluating the algorithm on real time sensor data using the implemented application.

Two falls were unfortunately not detected. It was already known that the ones simulating being stuck in the stirrup and dragged after the fall are more difficult to detect. The other missed fall was when simulating being thrown backwards of the horse. The fall was more smooth and slower than the ones of this type in the training data. The body model also attenuates the deceleration force by its mass when landing on the back. The guess was that the trigger threshold or some peak features were not fulfilled in this fall.

6.4 Battery consumption

For the algorithm to be useful in the future it is important that it does not drain the smartphone battery totally. A small test evaluating the battery consumption was performed. The application was started on a smartphone with fully charged battery. It ran for one hour. During this time the battery of the device decreased from 100 % to 89 %. Only two of these eleven percentage was due to the application according to the information provided by the device. The screen had used five percentage and the operating and Android systems had used the rest. No problems of using the application for a long period of time was neither found. Due to usage of limited sized queues in the Java implementation this is not expected to be a problem. To only use 2 % of the battery when running for one hour is considered as a great result.

7 | Discussion

This chapter discusses both methods and the final result of this master thesis. It ends by presenting thought about ethical aspects and the future of the algorithm.

7.1 Method

The work within this project is of widely spread nature, from figuring out how to make a human body model to handling real time data in Java. This made the work stimulating and fun but sometimes also heavy. When one problem was solved there was always several others on wait, dealing with totally new knowledge for me. The fact that this project was a one-man project, and not done together with an other student has sometimes felt like a drawback, missing someone to always be able to discuss ideas with. Maybe it had an inhibitory effect on the work.

When the project started, a detailed schedule of the working process was made. It was followed the first couple of weeks but because of different circumstances a delay between reality and the schedule arose. The work was then done in a more desultory manner. Even though the work is presented in separate chapters, the different processes have been worked on as parallel processes. As an example, both some algorithm development and some implementation in Java were done before all data was recorded.

7.1.1 Data recording

The method of data recording was chosen based on its feasibility during the limited period of time of this project, but leaves several questions for future work to answer. The limitation to only use one smartphone for recordings could have made the later developed algorithm too adjusted to the used sensors. There are for example smartphones having a lower range of the accelerometer. This was later taken into account, but it would have been desirable to have data of such a sensors as well.

The variety of normal riding data in this study was limited. Only two different horses and riders was used for recordings. Of course, the movement patterns recorded by the sensors varies with both horses and riders. The riders recording the data are quite experienced, maybe there are more movements affecting the device when worn by an unexperienced and unbalanced rider. An advantage was that the used horses are relatively large. The guess is that their large stride lengths makes the rider move more than on a small pony. The drop when dismounting the horse is larger, making them cause accelerations more similar to a fall.

The method of recording simulated fall events using a human model brings both advantages and disadvantages. One disadvantage is that it did not act totally like a human body. The advantage was that true ground proportions could be used. If a real human would have done the simulations, mattresses or something similar would have been needed. It was difficult to determine what is preferable but in this project the true ground properties was chosen rather than the right body behavior.

Since not all fall recordings was made using a horse, the appearance of the signals before a fall event did not have true properties. This would have been difficult to capture using other methods as well. A fall is often the result of unexpected movements of the horse, impossible to totally imitate. The data from before a trigger event was only used to calculate the change of orientation. The closest two seconds before a fall was not used at all, making the consequence of this problem small. When choosing the threshold for the feature representing lack of motion, additional data including breathing was recorded to reduce the impact of not having a real human.

The amount of different recorded falls did not represent their presence probability. For example, the rider is not dragged after the horse in eight out of 41 falls. The simulated data was in some sense more difficult than reality. Also the recordings of normal riding activities was not recorded to have true probabilities. These choices had a positive impact on the development process, but it makes it more difficult to say something about the actual result of this master thesis.

7.1.2 Algorithm development

To start the algorithm development process was one of the most difficult steps during this project. The amount of different signals, both raw and later calculated one dimensional data, made it hard to know where to begin. I had not worked with smartphone sensor data before and it required some time to just get familiar with the field. Reading relevant literature, especially of elderly falls, was very helpful and gave insights about possible solutions. Different features and algorithms were tried without the methodical way of work. The result was actually worked quite good but it was impossible to say if the problem was solved in the best possible manner of if any other signals or thresholds were more suitable. The presented method of separating events with high acceleration magnitude was found to solve this problem. The use of histograms to evaluate features and set threshold values was very helpful. I am pleased with this methodical working process.

7.1.3 Java implementation

The implementation in Java was made with more effort than knowledge. There are surely more decorous ways to implement the algorithm, but the application is runnable and the algorithm testable. This means that the aim of the implementation was fulfilled. The methods of handling data and being able to manage several trigger events simultaneously were some of the most difficult problems to solve. I am satisfied with the presented solutions for these problems.

It was a good decision to start the implementation in Java early on. It gave understandings of the sensors and the sensor data, useful during the algorithm development. It made the focus on an algorithm workable on real time sensor data strong from the beginning.

It is desirable to get more output from the application when a fall is detected, for example the feature values. Now it is not possible to know the reason for detecting a fall when an alarm is activated. To be able to improve the algorithm in the future this is important information to have. If more time was spent on the application this would have been the first thing to do.

7.1.4 Evaluation

The method of evaluation is the same as for data recording but made with the built Android application instead of AndroSensor. Unfortunately the evaluation is small, but some more riders and horses were used than during earlier data recording. The spreading of appearances of both fall situations and normal riding activity increases the amount of data needed to make a strong statement about the result.

One possible method to make a larger evaluation within the time boundaries of this project would have been to send the application to more people. Unfortunate most equestrian people in my surroundings have an iPhone, impossible to install the application on. If this method would have been used, it would have been an advantage to have an application able to save and send information about the length and results of recordings. Difficulties getting correct results is otherwise a risk.

7.2 Result

The proposed equestrian fall detection algorithm, presented in section 4.8, is the result of this master thesis. It is a relatively simple threshold based algorithm using a total of seven different features, calculated using totally four different sensors. The appearance of both normal riding activities and falls varies a lot. The solution to this problem was to both detect and reject trigger events in several manners in the algorithm.

The final features used in the algorithm captures different movements during and after a possible fall. The use of both accelerometer and gyroscope data gives advantages of capturing both linear and rotational movements close to the trigger event.

The calculated orientation change was very important for the result of this project. This feature is originally used in elderly fall detection, but is here adapted to suit equestrian falls by changing the window lengths. Maybe this feature is even more useful for equestrian than elderly fall detection. The natural events of orientation change affecting the device are assumed to be fewer during normal riding activities than in daily activities of elderly.

The last step of the algorithm uses software sensors to derive the feature value. As mentioned it is preferable to use hardware sensors. The median of vertical linear acceleration facilitated detection of falls where the rider is dragged after the horse. This is not captured by any other feature and therefore was the choice to use this feature anyway as a last decision possibility.

The Java implementation of the algorithm seemed to work as expected and was used for evaluation. No fall was detected in the normal riding recordings of this project. Both training and testing data were handled correctly by the algorithm. All simulated falls used for building the algorithm were detected but unfortunately two falls during evaluation were missed by the algorithm. In total 681 minutes, above 11 hours, of normal riding activity was performed within this project without false alarms. 53 out of 55 simulated falls were detected by the algorithm in total. Based on these results, I personally should feel safer using the future application with the proposed algorithm when riding alone. This feeling contributes to the assertion that the result of this project is successful, even though more tests needs to be done.

7.3 Ethics

Horses were used in this project, both during the first data recording and the final evaluation of the implemented algorithm. The health of the animal must always come first, and be taken into account when planning the recordings. An examples of having this aspect in mind was when the choice to not do all fall simulations from a horse was made. An other example was that no mounts on the horse was made several times in a row, just to get data. These recordings were made during separate sessions when mounts had to be made anyway, this to not cause any unnecessary stress of the horse's back.

It is important to not expose humans for danger or risk of injuries. This was one of the factors influencing the choice of using a body model for fall simulations and also which horse to use for these simulations. If a horse with more nerve would have been used, the risk of injuries would have increased.

If the algorithm is later implemented in a public application it is important to not make the rider feel more monitored than necessary. The handling of sensor data from the users smartphone must be made without affecting the integrity. As the algorithm is implemented today, no data is saved for a longer time period than 9 seconds, it is hence not possible to recreate a total ride.

7.4 Future work

No effort was made to analyze the need of the future product in mind. When the idea of this master thesis has been presented many positive words have been said. Both by people having an interest in horses and not. Examples are people wanting the future application for their relatives. Someone has seen the possibility to earn money on an application with this purpose. The result of this master thesis is satisfying enough to want the algorithm implemented in an application able to contact relatives. The plan is to finish the application to be fully workable, at least for own purpose, in the short future.

8 | Conclusion

This chapter presents a short summary and the conclusions of this master thesis.

The work within this master thesis was produced on the identified need of a safety system for equestrians riding alone. The aim was to develop a fall detection algorithm based on smartphone sensor data customized for horseback riding. Necessary data from both simulated falls and normal horseback riding activities were recorded and used for the algorithm development.

The choice of sensors, signal representations and features were successively made. The result of this project is the equestrian fall detection algorithm presented in section 4.8. It is a threshold based algorithm, using in total seven features, derived from four sensors. The algorithm is adapted to the widely spread differences found in the fall data by being able to detect falls in three different manners.

The final algorithm was implemented as an Android application running on real time data for evaluation. Not a single false alarm was obtained during eleven hours of normal riding activities. All 41 simulated falls used during the development process were correctly classified, but unfortunately only 12 of 14 evaluation falls were detected. To make a stronger statement about the result more evaluation sessions needs to be done. Based on the results within this project the algorithm is considered to have possibility to increase equestrian safety in the future.

Bibliography

[1] Srinivasan V., Pierre C., Plog B., Srinivasan K., Petraglia A.L., Huang J.H. *Straight from the horse's mouth: neurological injury in equestrian sports.* Neurological Research vol. 36 no. 10 pp. 873-877, 2014.

[2] Liljenstolpe C. *Horses in Europe*. Swedish University of Agricultural Sciences (SLU), 2009.

[3] Myndigheten för Samhällsskydd och beredskap. *Hästar och olyckor*. `https://ida.msb.se/dokument/infoblad/Hast.pdf` (Accessed 2016-11-04).

[4] International Olympic Committee. *Equestrian Sport: History of Equestrian Sport at the Olympic Games*. Olympic Studies Centre, 2015.

[5] Eriksson A., Nyman S., Nyberg J., Lundström T., Van Dooren R., Wallén A. *Hästens förutsättningar för arbete under utbildning och träning.* Equitellus AB 2:a upplagan, 2016.

[6] Svenska Ridsport Förbundet. *Flykt, flock och fortplantning.* `http://www.ridsport.se/Hastkunskap/Beteende/Instinkter/` (Accessed 2016-11-08).

[7] Kriss T.C. and Kriss V.M. *Equine related neurosurgical trauma: A prospective series of 30 patients* Journal of Trauma-Injury Infection & Critical Care, vol. 43, No. 1, pp. 97-99, 1997

[8] Havlik H. S. *Equestrian Sport-Related Injuries: A Review of Current Literature* Current Sorts Medicine Reports, vol. 9, No. 5, pp. 299-302, 2010.

[9] Carrillo E.H., Varnagy D., Bragg S.M., Levy J., Riordan K. *Traumatic Injuries Associated with Horseback Riding* Scandinavian Journal of Surgery, vol. 96 pp 79-82, 2007.

[10] O'Brien D. *Look Before You Leap: What Are the Obstacles to Risk Calculation in the Equestrian Sport of Eventing?* Animals, vol. 6, No. 13, 2016.

[11] Horse and hound *Fifth event rider dies in seven months*
`http://www.horseandhound.co.uk/news/event-rider-santiago-zone-.dies-cross-country-fall-599812` (Accessed 2017-05-09).

[12] Carmichael S. P., Devenport D. L., Kearney P. A., Bernard A. C. *On and off the horse: Mechanisms and patterns of injury in mounted and unmounted equestrians.* Injury Int. J. Care Injured vol. 45 pp. 1479-1483, 2014.

[13] Svenska Ridsport Förbundet. *Säker med häst.* http://www.ridsport.se/ImageVault/Images/id_340/ImageVaultHandler.aspx (Accessed 2016-11-08).

[14] Hippson. *”Säkerhetsvästens passform är A och O”.* https://www.hippson.se/blogs/redaktionenspryltips/sakerhetsvastens-passform-ar-a-och-o.htm (Accessed 2017-05-09).

[15] Medical Equestrian Association. *Concussion.* http://www.medequestrian.co.uk/rider-safety/benefits-and-risks-of-riding/concussion/ (Accessed 2016-11-08).

[16] Saddle Up Safely. *Horse-Related Injury.* http://www.albertaequestrian.com/wp-content/uploads/2015/03/Horse-Related-Injury-brochure.pdf (Accessed 2016-11-08).

[17] Google Play. *SafeRider.* https://play.google.com/store/apps/details?id=nl.equinovum.apps.saferider (Accessed 2016-10-06).

[18] Google Play. *SafeRider Lite.* https://play.google.com/store/apps/details?id=nl.equinovum.apps.saferider.lite (Accessed 2016-10-06).

[19] Google Play. *Horse Rider SOS.* https://play.google.com/store/apps/details?id=com.pocketapp.thinkequus (Accessed 2016-10-06).

[20] Bergbom L., Engelbrektsson C., Granberg S., Streling L. *Ryttar jalp.* http://studentarbeten.chalmers.se/publication/219261-ryttarjalp-sakerhetsapp-for-ryttare (Accessed 2016-10-06).

[21] ICE dot. *ICEdot crash sensor.* http://site.icedot.org/site/ (Accessed 2016-10-06).

[22] Hippson. *Ice-dot Crash Sensor blev Årets pryl 2015!* http://www.hippson.se/artikelarkivet/utrustning/ice-dot-crash-sensor-blev-arets.htm (Accessed 2016- 10-06).

[23] allone. *allone app.* http://www.alloneapp.com/ (Accessed 2017-03-27).

[24] mynewsdesk. *Ny app-gratis olyckslarm i mobilen för ryttare.* https://www.mynewsdesk.com/se/allone-ab/pressreleases/ny-app-gratis-olyckslarm-i-mobilen-foer-ryttare-1194919 (Accessed 2017-03-17).

[25] Habib M. A., Mohaktar M. S., Kamaruzzaman S. B., Lim K. S., Pin T. M., Ibrahim F. *Smartphone-Based Solutions for Fall Detection and Prevention: Challenges and Open Issues* Sensors vol. 14, No. 4, pp. 7181-7208, 2014.

[26] Shoaib M., Bosch S., Incel O. D., Scholten H., Havinga P. J. M. *Fusion of Smartphone Motion Sensors for Physical Activity Recognition.* Sensors vol. 14, No. 6, pp. 10146-10176, 2014.

[27] Google Play. *Equilab Hästträning, Ridsport.* https://play.google.com/store/apps/details?id=horse.schvung.equilab\&hl=sv (Accessed 2017-05-09).

[28] Hippson. *Rid-appen Equilab blev Årets pryl 2016!*. https://www.hippson.se/artikelarkivet/prylkollen/rid-appen-equilab-blev-arets-pryl.htm (Accessed 2017-05-09).

[29] ICD: Analyze the future. *Smartphone OS Market Share, 2016 Q2*. http://www.idc.com/prodserv/smartphone-os-market-share.jsp (Accessed 2016-11-10).

[30] Android Developers. *Sensors Overview*. https://developer.android.com/guide/topics/sensors/sensors_overview.html (Accessed 2016-11-01).

[31] Gustafsson, F. *Statistical Sensor Fusion* Ed. 1:1, Lund: Studentlitteratur AB, 2010.

[32] Android Developers. *Motion Sensors*. https://developer.android.com/guide/topics/sensors/sensors_motion.html (Accessed 2017-03-27).

[33] Grahm L., Jubrink H-G., Lauber A. *Modern Industriell Mätteknik, Givare* Ed.5:2, Lund: Studentlitteratur, 2007.

[34] Hammack B., Ryan P., Ziech N. *Eight Amazing Engineering Stories: Using the Elements to Create Extraordinary Technologies*. Articulate Noise Books 1st. ed. 2012.

[35] YouTube. *How MEMS Accelerometer Gyroscope Magnetometer Work & Arduino*. https://www.youtube.com/watch?v=eqZgxR6eRjo (Accessed 2017-03-29).

[36] Virtual Reality Society. *Understanding Sensors: Magnetometers, Accelerometers and Gyroscope*. https://www.vrs.org.uk/virtual-reality-gear/motion-tracking/sensors.html (Accessed 2017-03-27).

[37] Android Developers. *Position Sensors*. https://developer.android.com/guide/topics/sensors/sensors_position.html (Accessed 2017-03-21).

[38] Bosch. *BMI160 Small, low power inertial measurement unit*. http://www.mouser.com/ds/2/783/BST-BMI160-DS000-07-786474.pdf (Accessed 2016-11-07).

[39] Bosch. *BMM150 Geomagnetic Sensor*. https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMM150-DS001-01.pdf (Accessed 2017-03-08).

[40] Google Play. *Sensors Multitool*. https://play.google.com/store/apps/details?id=com.wered.sensorsmultitool (Accessed 2017-05-10).

[41] Google Play. *AndroSensor*. https://play.google.com/store/apps/details?id=com.fivasim.androsensor\&hl=sv (Accessed 2017-03-05).

[42] fivasim. *AndroSensor* http://www.fivasim.com/androsensor.html#ch1 (Accessed 2017-03-05).

[43] Brolin K., Wass J. *Explicit finite element methods for equestrian applications*. Procedia Engineering, conf. of ISEA. vol. 11, no. 147, pp.275 - 280, 2016.

[44] Attal F., Boubezoul A., Oukhellou L., Cheifetz N. and Espié S. *The Powered Two Wheelers fall detection using Multivariate CUmulative SUM (MCUSUM) control charts*. Intelligent Transportation Systems (ITSC), pp. 1280-1285, 2014.

[45] Figueiredo I.N., Leal C., Pinto L., Bolito J. and Lemos A. *Exploring smartphone sensors for fall and emergency detection*. Universidade de Coimbra, 2015.

[46] Tözeren, A. *Human Body Dynamics: Classical Mechanics and Human Movement* New York: Springer-Verlag, 2000.

[47] Shoaib M., Bosch A., Durmaz I. O., Scholten H.,Havinga P.J.M., *Fusion of Smartphone Motion Sensors for Physical Activity Recognition* Sensors vol.14, no.6 pp. 10146-10176. 2014.

[48] Sparr, G. *Linjär algebra*. Edition 2:21. Lund: Studentlitteratur, 1994.

[49] Hwang S-Y., Ryu, M-H., Yang Y-S., Lee N-B. *Fall Detection with Three-Axis Accelerometer and Magnetometer in a Smartphone.* National University, Korea.

[50] Dai J., Yang Z., *Mobile Phone-Based Pervasive Fall Detection* Personal and Ubiquitous Computing, 2010.

[51] Oracle. *Class TimerTask*. `https://docs.oracle.com/javase/7/docs/api/java/util/TimerTask.html` (Accessed 2017-05-11).

A | Appendix

This Appendix shows examples of sensor data for all three one dimensional signal representations of both normal horseback riding activity and simulated fall events separately.

The first three figures illustrates examples of data during normal riding activities. All three one dimensional signal representations for the same period of time are shown. Figure A.1 shows the acceleration magnitude, figure A.2 shows vertical linear acceleration and third figure, A.3, shows the angular velocity magnitude.

The following three figures shows examples of eight different simulated fall situations. The acceleration magnitude is shown in figure A.4, the vertical linear acceleration in figure A.5. Last the angular velocity magnitude is shown in figure A.6.

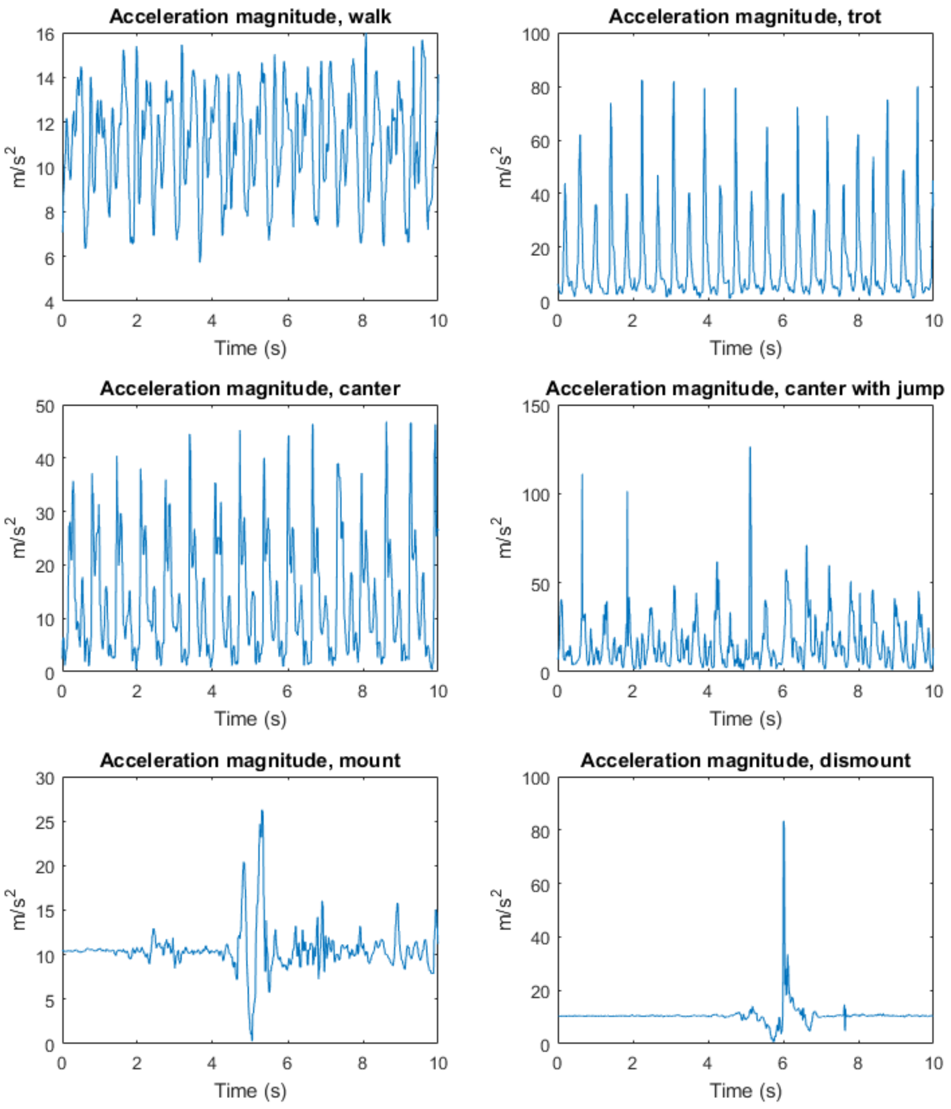


Figure A.1: The acceleration magnitude captured during different types of normal riding activity.

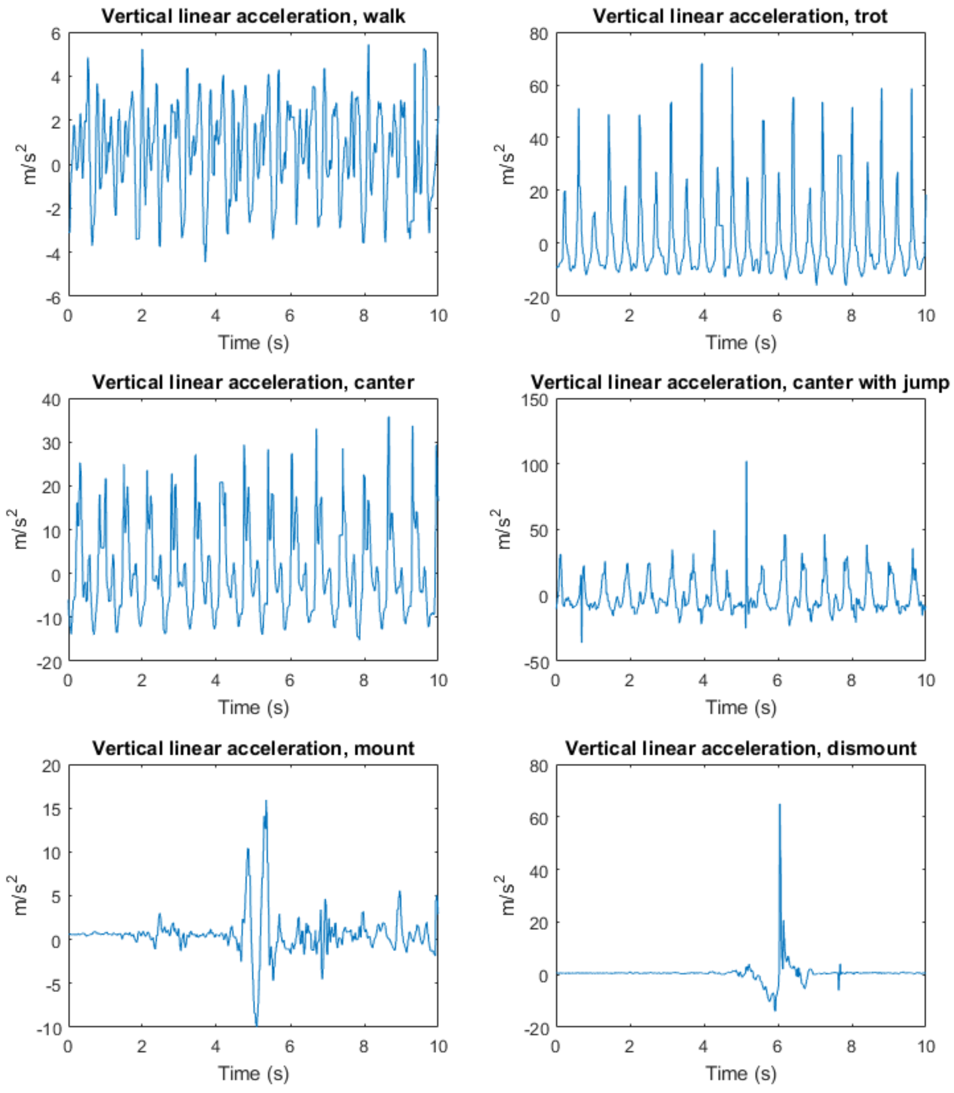


Figure A.2: The vertical linear acceleration during different types of normal riding activity.

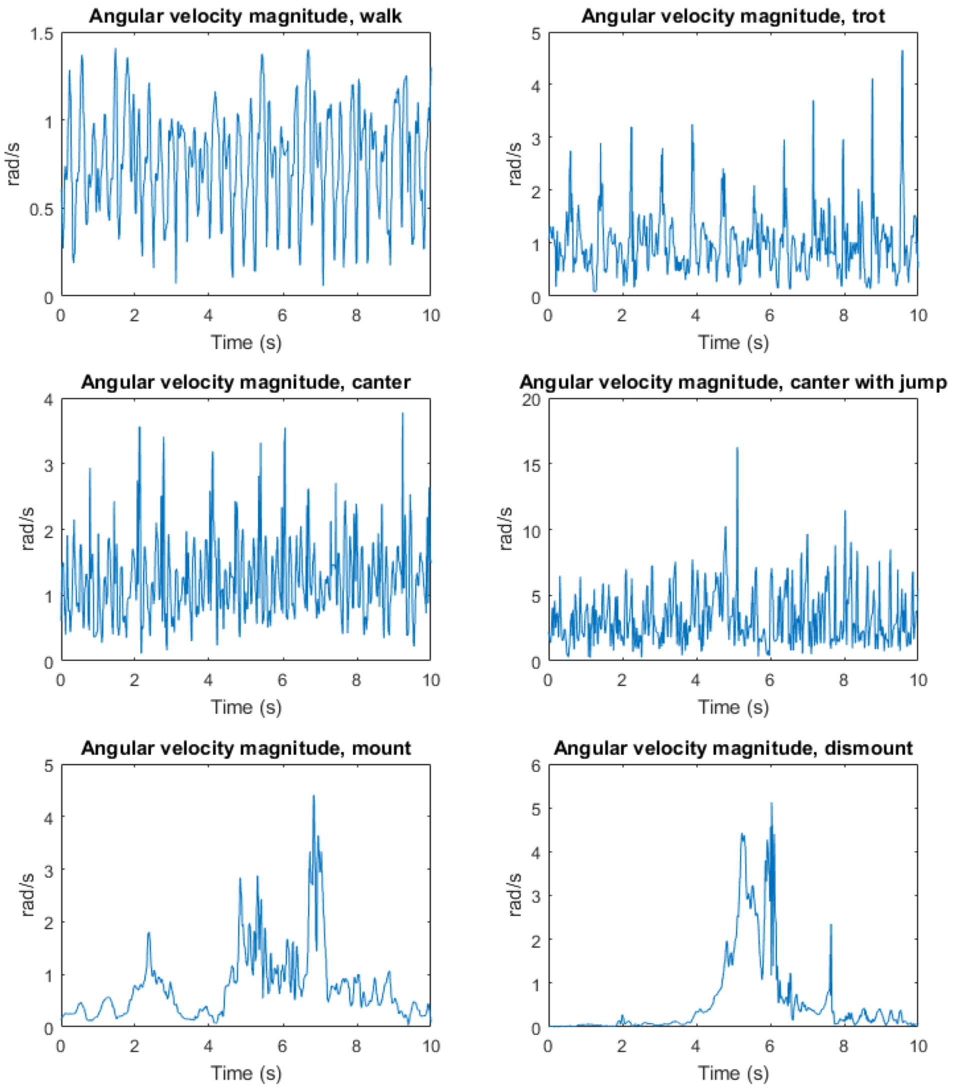


Figure A.3: The angular velocity magnitude measured using the gyroscope during different types of normal riding activity.

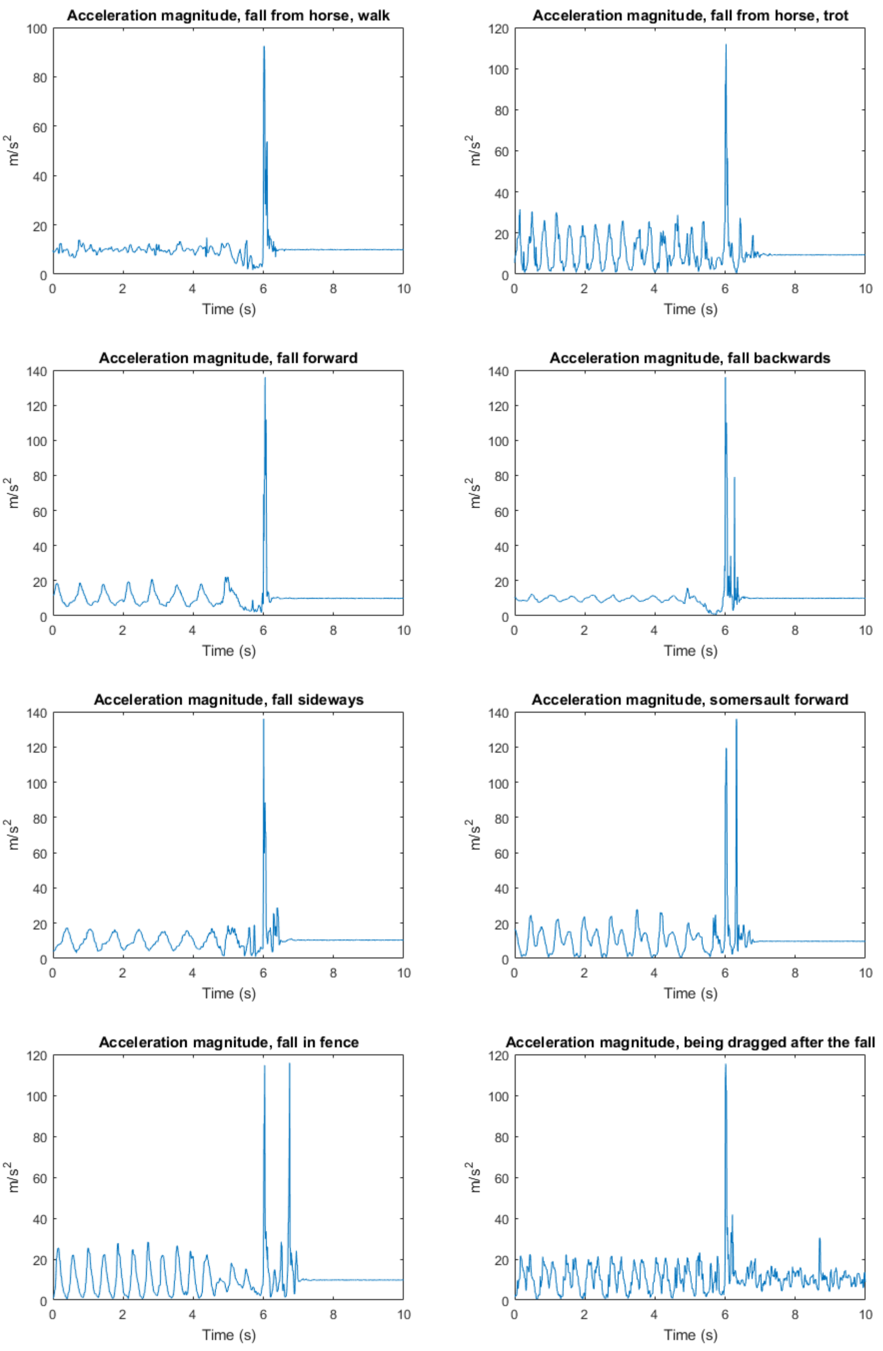


Figure A.4: The acceleration magnitude captured by the accelerometer during eight types of falls.

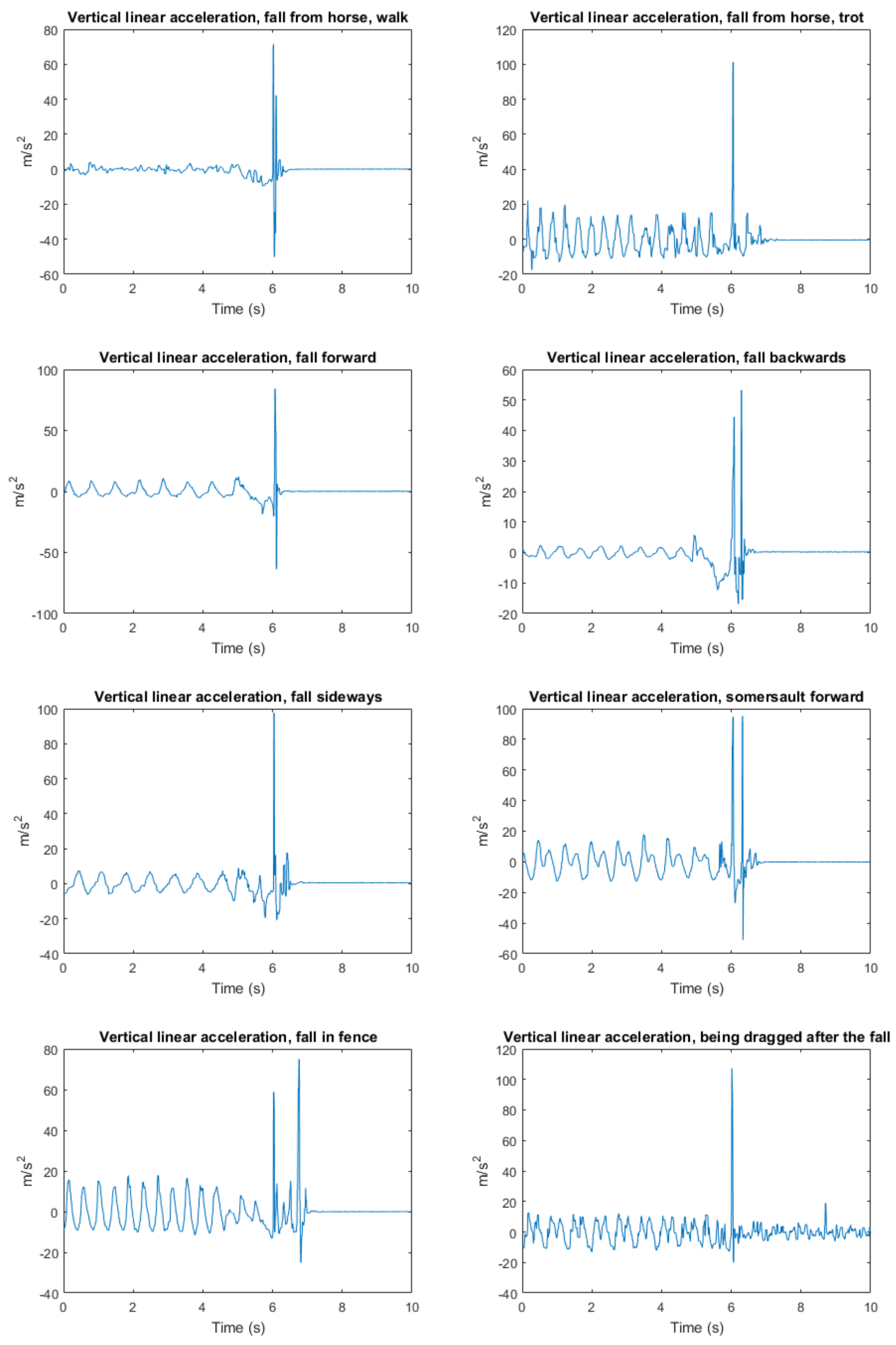


Figure A.5: The vertical linear acceleration during eight types of falls.

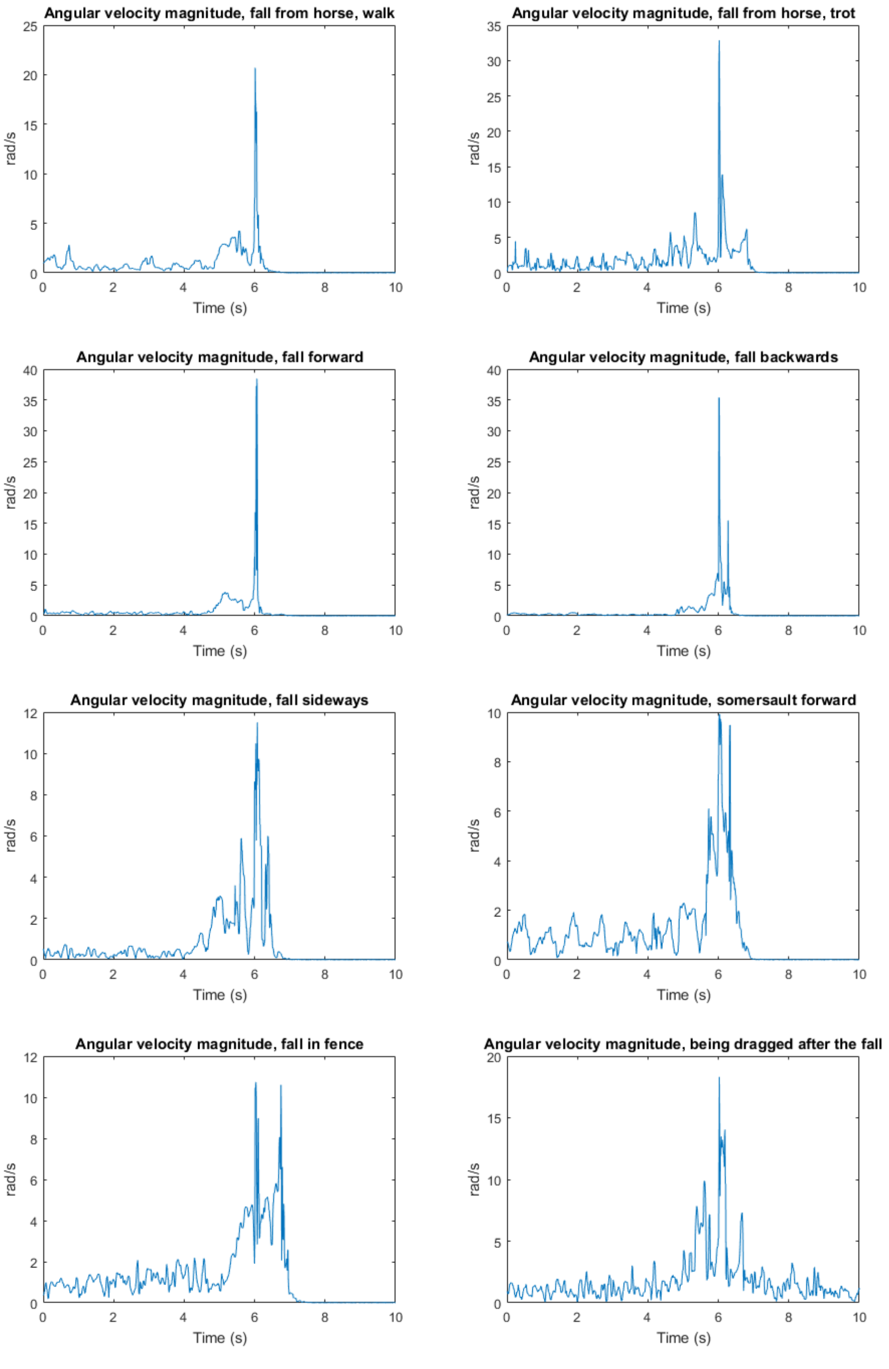


Figure A.6: The angular velocity magnitude measured using the gyroscope during eight different types of falls.

B | Appendix

This Appendix shows all histograms produced for choosing features and thresholds. In all figures does the blue bins represent normal riding activity and the red bins represents simulated fall situations. Worth to notice in the following figures is that the plots shows the probability of a value and not the actual number of corresponding events. The blue bins represents a total of 9618 values while the red bins corresponds to 41 values. Even though the blue bins are really small, and sometimes barely visible, they can contain more events than a red bin representing the same values.

Figure B.1, B.2 and B.3 presents the peak features for the three different signals respectively. Then is the features calculated on the time after a trigger event shown in figure B.4, B.5 and B.6. Last but not least is the orientation change feature values presented in figure B.7.

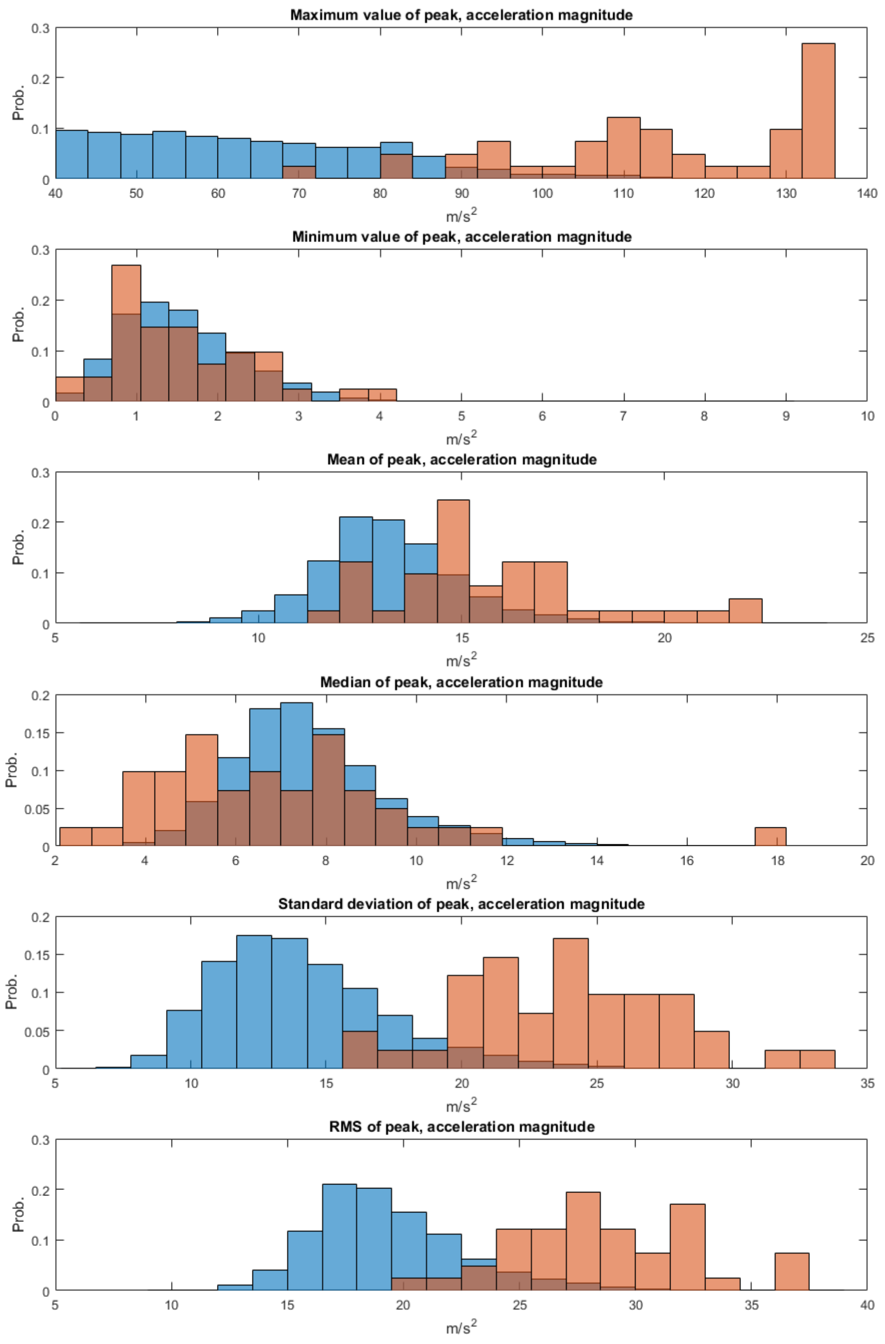


Figure B.1: Peak features of acceleration magnitude.

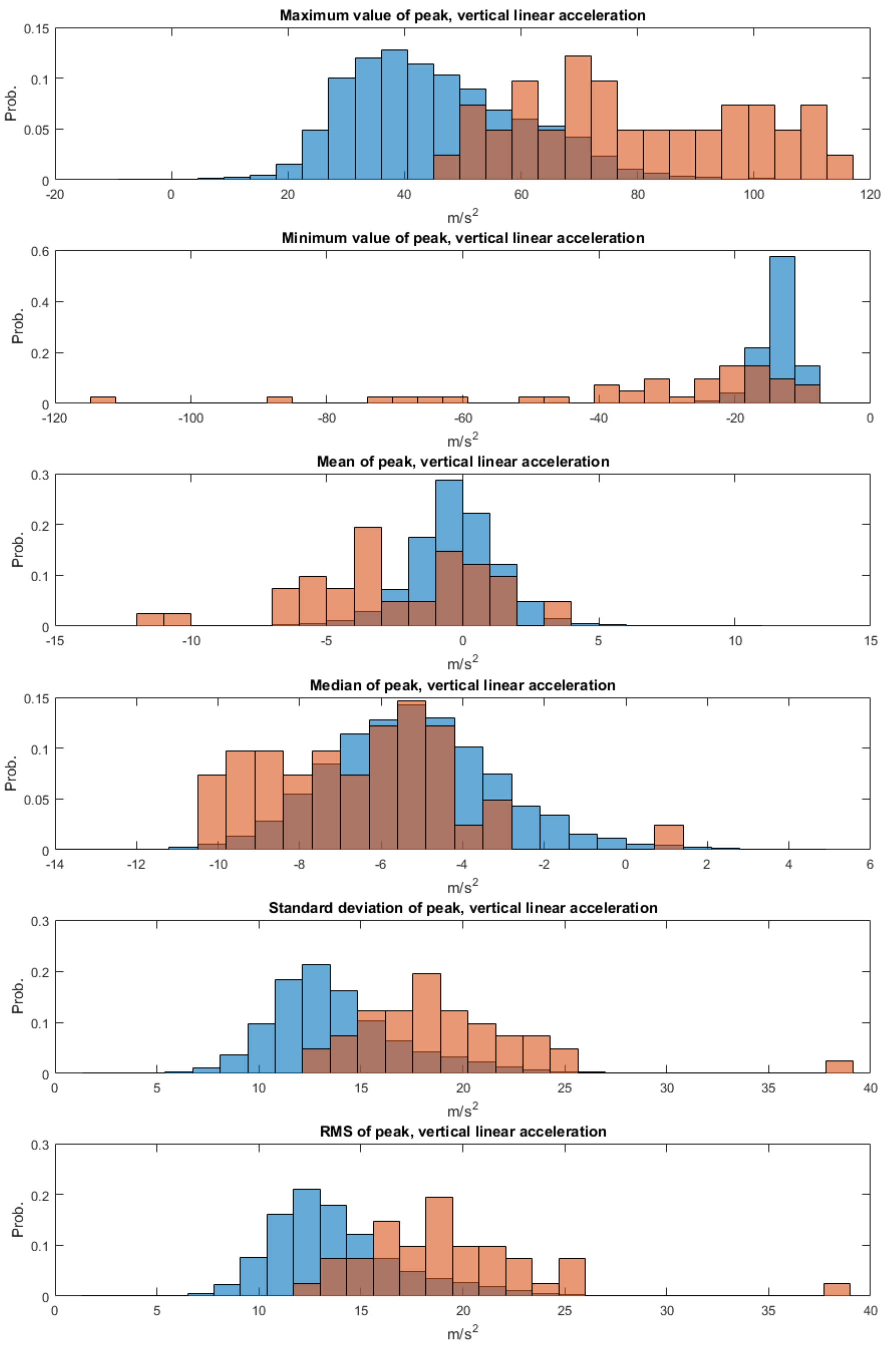


Figure B.2: Peak features of vertical linear acceleration.

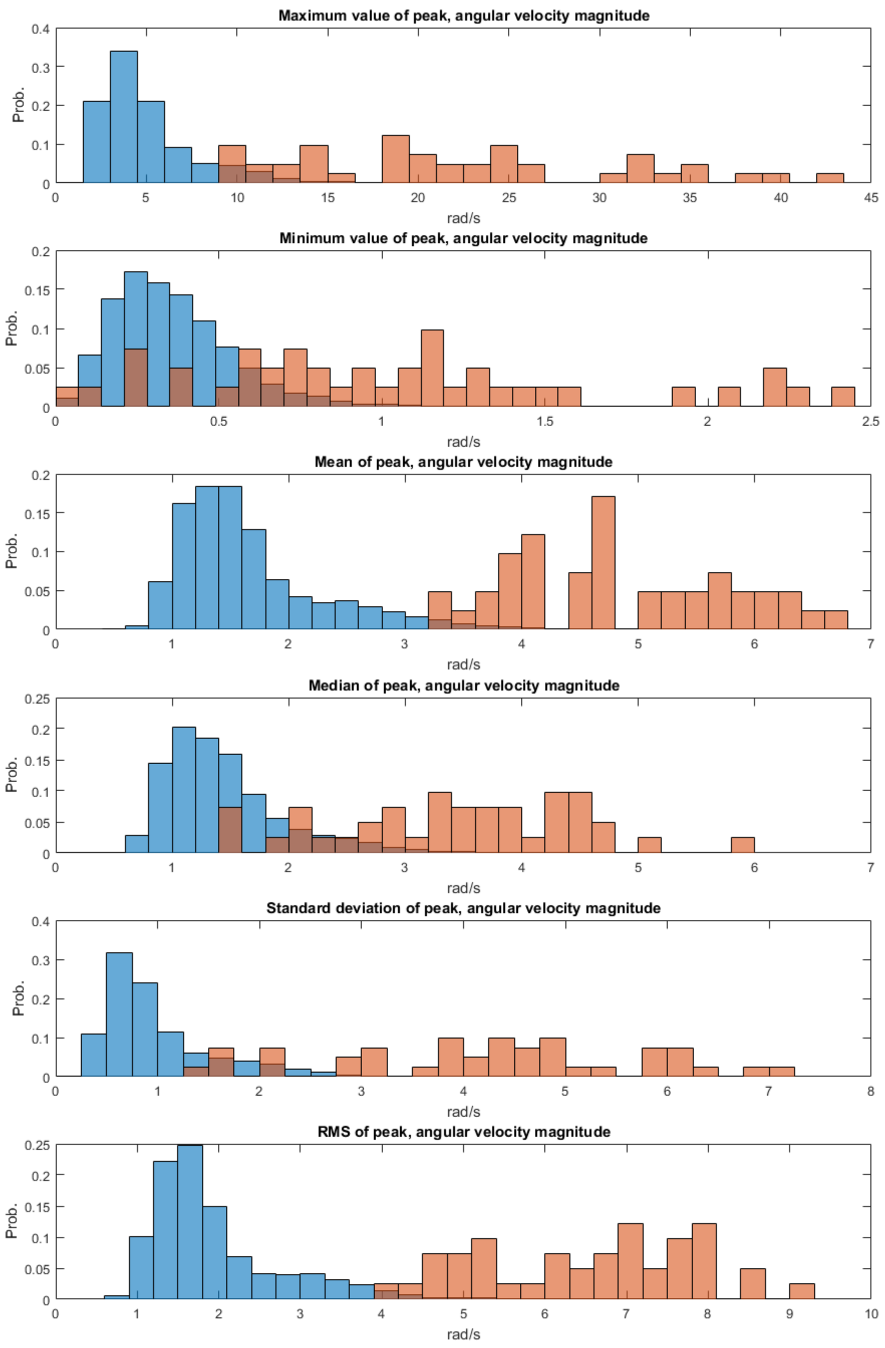


Figure B.3: Peak features of angular velocity magnitude.

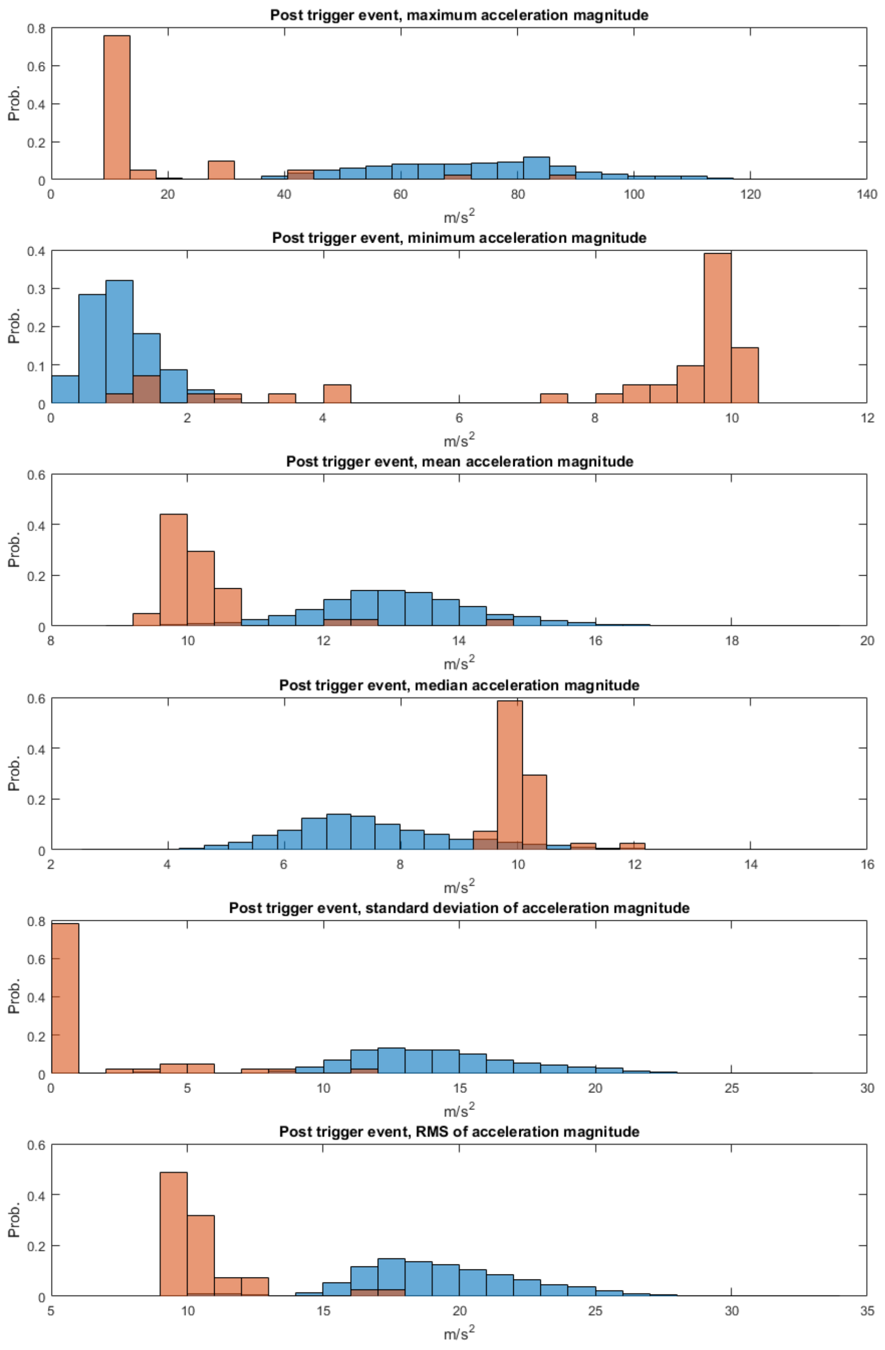


Figure B.4: The acceleration magnitude features calculated on data from after the trigger event.

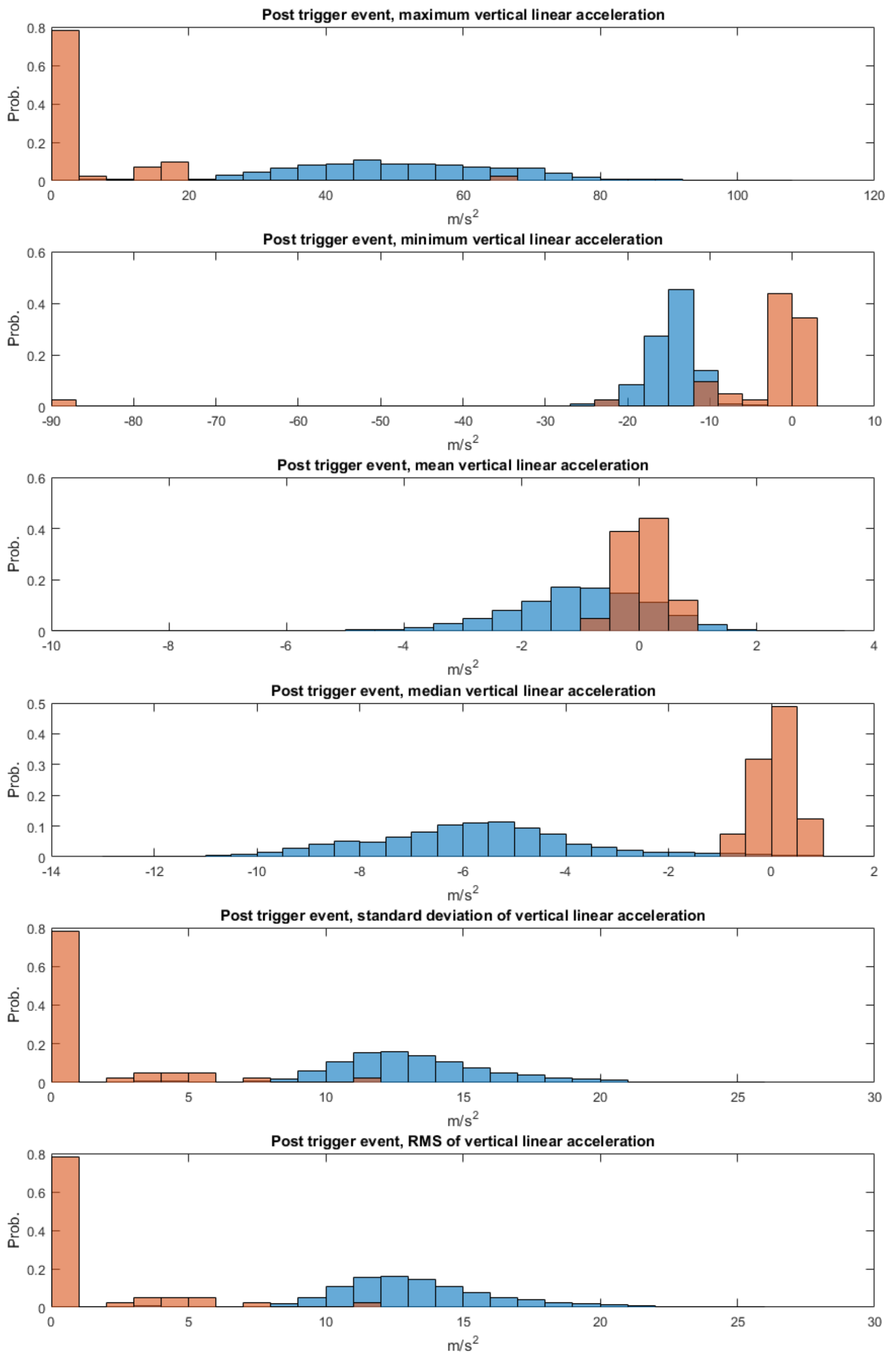


Figure B.5: The vertical linear acceleration features calculated on data from after the trigger event.

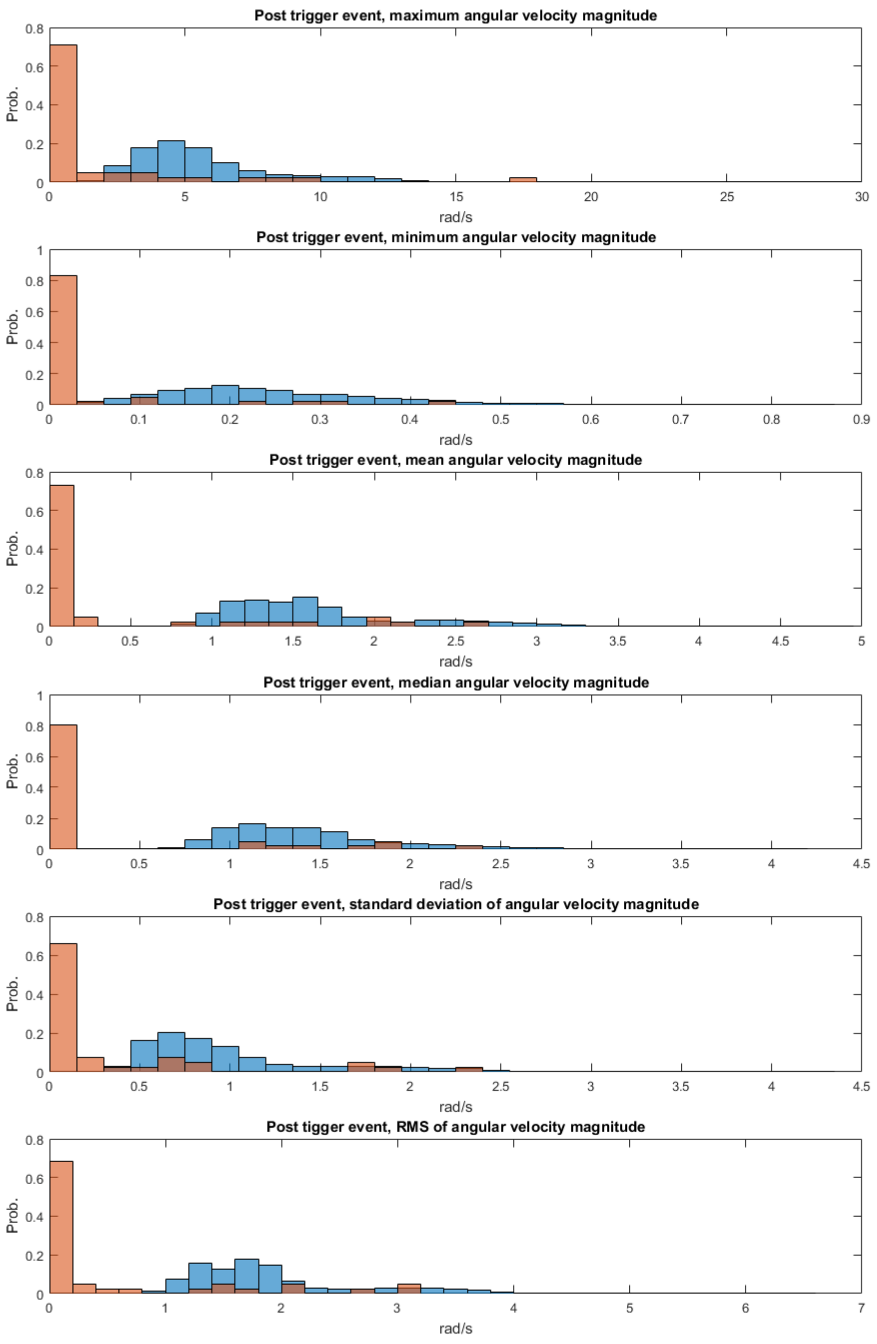


Figure B.6: The angular velocity features calculated on data from after the trigger event.

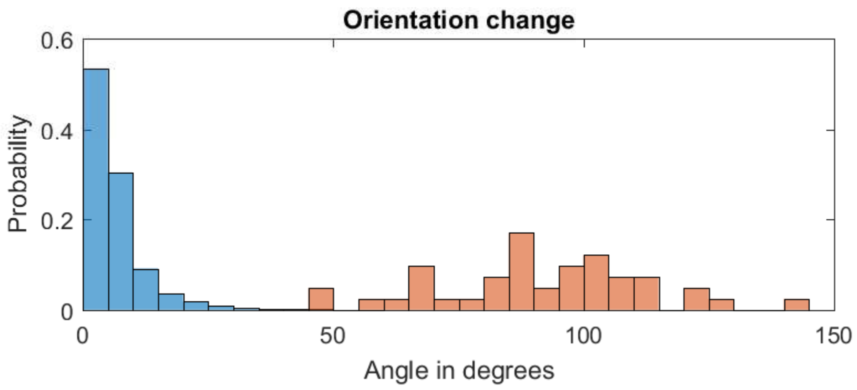


Figure B.7: Values of the feature representing the orientation change of the device during a trigger event.