UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Madis Kariler

Road Surface Quality Detection Using Accelerometer Data

Bachelor's Thesis (9 ECTS)

Supervisor: Amnir Hadachi, PhD

Tartu 2017

Road Surface Quality Detection Using Accelerometer Data

Abstract

Smartphones and tablets have become an integral part of modern society. Increasing popularity, computing power and availability of different sensors have opened new ways to utilize these smart devices. In addition to smartphones and tablets, another important aspect of modern society is the road network. Since roads are used by a large number of people every day, deterioration remains a problem in many places. The aim of this thesis is to provide an algorithm for road surface quality detection and an Android application for accelerometer data collection. The algorithm uses accelerometer data, collected from different roads, in order to train a classifier for road surface quality detection. The accelerometer data is collected with the aforementioned Android application, which is developed as a result of this thesis. The overall result of this thesis is a proof of concept solution, that can be used to detect and classify the surface quality of roads into three road quality categories.

Keywords: *SVM, Machine learning, GPS data, Accelerometer data, Android, Road quality, Smartphone*

CERCS: P170 Computer science, numerical analysis, systems, control

Sõiduteekatte kvaliteedi tuvastamine aktselerомeetri andmeid kasutades

Lühikokkuvõte

Nutiseadmed on muutunud igapäevaelu lahutamatuks osaks. Suurenev populaarsus, arvutusvõimsus ja erinevate sensorite olemasolu on avanud uusi võimalusi nutiseadmete kasutamiseks. Lisaks nutiseadmetele on tänapäeva ühiskonnas tähtis roll ka teede võrgustikul. Kuna sõiduteid kasutab igapäevaselt suur hulk inimesi, siis on teede lagunemine paljudes kohtades endiselt probleemiks. Käesoleva töö eesmärgiks on pakkuda algoritm sõiduteekatte kvaliteedi tuvastamiseks ning Androidi rakendus aktselerомeetri andmete kogumiseks. Algoritm kasutab kogutud aktselerомeetri andmeid erinevatelt sõiduteedelt, eesmärgiga treenida klassifitseerija teekatte kvaliteedi tuvastamiseks. Aktselerомeetri andmeid kogutakse eelnevalt mainitud Androidi rakendusega, mis valmib käesoleva töö raames. Töö lõpptulemuseks on *proof of concept* lahendus, mis võimaldab tuvastada ning klassifitseerida sõiduteid teekatte kvaliteedi järgi kolme defineeritud kategooriasse.

Võtmesõnad: *SVM, Masinõppimine, GPS-andmed, Aktselerомeetri andmed, Android, Teekatte kvaliteet, Nutitelefon*

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Table of contents

Abstract **2**

Lühikokkuvõte **3**

List of abbreviations and acronyms **6**

Chapter 1: Introduction **7**

1.1 General View 7

1.2 Motivation and Problem Statement 7

1.3 Objectives 8

1.4 Limitations 9

1.5 Road Map 9

Chapter 2: State of the art **10**

2.1 Introduction 10

2.2 Data collection 10

2.3 Machine learning 11

2.4 Conclusion 13

Chapter 3: Methodology **15**

3.1 Introduction 15

3.2 Road Surface Data Collector application 15

3.2.1 User interface 17

3.2.2 Data recording 18

3.2.3 Data output 19

3.2.4 Linear acceleration transformation process 21

3.3 Road surface quality detection algorithm 24

3.3.1 Data preprocessing 25
3.3.2 Feature extraction 26
3.3.2.1 Time domain features 27
3.3.2.2 Frequency domain features 28
3.3.3 Supervised learning 29
3.4 Conclusion 29
Chapter 4: Results and analysis 31
4.1 Introduction 31
4.2 Data collection 32
4.2.1 Collection setup 32
4.2.2 Chosen roads 34
4.3 Performance comparison of feature sets 36
4.4 Additional testing 40
4.5 Conclusion 41
Chapter 5: Conclusion and perspectives 42
5.1 Conclusion 42
5.2 Future work 43
Bibliography 44
Appendix 47
License 47

List of abbreviations and acronyms

AOSP

Android Open Source Project

CSV

Comma-separated values

DFT

Discrete Fourier transform

FFT

Fast Fourier transform

GPS

Global Position System

IDE

Integrated development environment

KNN

K-Nearest Neighbor

RBF

Radial basis function

SVM

Support Vector Machine

Chapter 1: Introduction

1.1 General View

Smartphones and tablets have become an important aspect of modern society. In recent years, the smartphone market has seen substantial increases in the number of shipments all over the world [1]. Furthermore, mobile internet use has also seen steady growth in the last few years and has recently surpassed desktop internet usage for the first time worldwide [2]. The popularity of mobile devices, coupled with their increasing computing power, means that smartphones are performing an increasingly wide range of functions. Applications on mobile platforms such as Google Android and Apple iOS, can also take advantage of devices' internal sensors, which opens up new ways to utilize these smart devices. Sensors can be leveraged for different tasks, such as detecting the orientation of the device or for more complex problems, such as human activity detection [3].

Road infrastructure is another important part of modern society. A large number of people frequently use roads, whether it be in a personal car, on a bike, on foot, or in public transport. Many public services and businesses also depend on the road network in one way or another. As the motorization rate continues to grow [4], a well-maintained and safe road network is beneficial for everybody involved.

1.2 Motivation and Problem Statement

Roads require frequent oversight to find and fix anomalies as soon as possible to keep them from deteriorating further. In addition to normal wear and tear, factors such as poor construction quality, heavy traffic, poor drainage, weak subgrade[1] and large variations in temperature can contribute to the creation of potholes, cracks and other anomalies, which significantly lower the overall road quality and ride comfort of asphalt pavement roads [5]. Poor road quality also causes mechanical strain for vehicles, increasing the need for repairs. Furthermore, it can distract

[1] Subgrade - soil that provides support to the pavement from below [5].

drivers, as they might try to dodge road anomalies, which can lead to dangerous situations in traffic. However, building new roads and keeping existing roads in satisfactory shape can be very expensive [6]. Inspecting roads manually or by using special dedicated hardware can also be time-consuming. Therefore, developing a more accessible and efficient road quality detection solution, that can highlight problematic areas, would be beneficial to both the road users and the administrators.

1.3 Objectives

The aim of this thesis is to develop an Android application for acceleration data collection and 2 to design and implement an algorithm for road surface quality detection that can use acceleration data to estimate the quality of roads. The idea behind this approach is that acceleration data manifests certain different characteristics, depending on the roughness of road quality that the device was experiencing, while being driven in a car. These characteristics could then be learned and used to predict and classify the quality of road segments into three separate categories.

The road surface quality detection part of this thesis consists of converting the collected acceleration data into usable data, extracting features from the data and finally, training and testing a Support Vector Machine classifier. In total, six features are calculated based on the acceleration data: mean, variance, sum of the absolute values of maximum and minimum amplitudes, mean of absolute values, energy (frequency domain) and entropy (frequency domain).

An Android smartphone, running a data collection application (Road Surface Data Collector from now on), and a car are used to collect acceleration data of different roads. The acceleration data is collected in 1 second length time based windows in three, progressively worse categories: *smooth*, *bumpy* and *rough*, respectively, in and around the city of Tartu. The collected data files consist of timestamps, acceleration data and corresponding GPS-coordinates.

As a result of these two components, a proof of concept solution is provided by this thesis, which could be a basis for a future software solution, that consists of an Android data collection application and a server-side detection algorithm. The software solution could be used by both the road users and the administrators to record acceleration data and visualize the quality of

different roads to either plan inspections and maintenance or determine routes that provide a smooth driving experience.

1.4 Limitations

Due to time and financial limitations, a single car and smartphone are used during the Android application development and data collection. Specifically, a Volkswagen Jetta (2014 model) and a Huawei P9 smartphone.

Another limitation is the aspect of speed. Research into speed and its influence on the capability of correctly classifying roads, is not included in this thesis. Due to this limitation, all of the acceleration data is collected at a speed of around 30 km/h.

Another thing to note, is that during the development and testing of the road quality detection algorithm and the Road Surface Data Collector application, no server-side solution is used. From the smartphone, the collected data is retrieved manually and the detection algorithm itself is run on a personal computer.

1.5 Road Map

Chapter 2 provides an overview of some of the previous work on the topic of road quality detection. Technologies and techniques, that were chosen for this thesis, are also discussed.

Chapter 3 describes the implementation of the Road Surface Data Collector application and also covers various aspects of the road quality detection algorithm.

Chapter 4 gives insight into the data collection process as well as the testing strategy. Results of the testing are also provided.

Chapter 5 concludes this thesis and discusses future work.

Chapter 2: State of the art

2.1 Introduction

This chapter provides an overview of some of the related work on the subject of road quality detection using accelerometer data. The overview is divided into two subchapters. Subchapter 2.2 focuses on the data collection setups and techniques, while subchapter 2.3 delves into the machine learning techniques, that have been used for road quality and anomaly detection. The results, that those techniques yielded, are also described in this subchapter.
Subchapter 2.4 concludes this chapter by offering perspective on the techniques and technologies, that were chosen for this thesis.

2.2 Data collection

The data collection process is a critical step in this topic. Different methods have been used in previous work, however, all work include the collection of accelerometer data, in one way or another. This subchapter aims to provide an overview of the techniques used to collect this data.
Tonde et al. [7] used an Android smartphone with a fixed orientation to collect the acceleration data. Collection was done at a frequency rate of 100 Hz. From the smartphone, data was uploaded to a desktop computer. Data files were then manually checked to only select those road sections, that provide complete data sets. Overall, four categories of road quality were distinguished.
Seraj et al. [8] also used an Android smartphone for data collection, specifically, a Samsung Galaxy S2. The device was fixed to the windshield of the car and two different recording frequencies were used - 47 Hz and 93 Hz. Data was collected using five different cars in Netherlands and Albania and consisted of accelerometer, GPS and gyroscope sensor readings. Data labelling was done manually by recording the collection process in audio and video.
Alessandroni et al. [9] mainly focused on the architectural aspect of their road surface monitoring system. Their solution features an Android application which gathers data from an

accelerometer and GPS. Both sensors are run at the highest available frequency. Since GPS operating frequency was around 1 Hz, roughly one sample of data was collected every second. An experiment was also carried out, where the data collection application was installed on two Motorola Moto G smartphones, which were then put inside two public buses.

Instead of basing their solution on car-based transport, Hoffmann et al. [10] focused on bicycles and cycling routes by mounting a smartphone, which was used to collect the data, to the handle of a bicycle. In order to prove, that their solution also works on older smartphones, a Nokia 5800, released in 2008 [11], was used. Data was recorded at a frequency of around 37 Hz and consisted of accelerometer sensor readings and GPS data. In the direct road surface classification approach, three different road surface types were distinguished.

Astarita et al. [12] used several Android smartphones and a single tablet in order to collect data of different road anomalies. Three different mounting solutions in the car were tested: completely fixed, partially fixed (vertical axis free) and completely free. Furthermore, they describe a process of reorienting the vertical acceleration using two of the three Euler angles, specifically the roll angle α and the pitch angle β. The reorientation technique is based on the fact that when the vehicle and the smartphone are stationary, the only acceleration registered by the accelerometer is the acceleration due to gravity.

Rather than using smartphones, Eriksson et al. [13] used embedded computers with acceleration and GPS sensors. Raw data was collected 380 times per second and included time, location, speed, heading and acceleration readings. Different mounting points for the accelerometer sensor were tested and it was concluded, that the cleanest signal is produced, when the sensor is attached to the dashboard of the car. Other tested attachment points were the right side of the windshield and the embedded computer itself. A set of data samples was collected, which contained examples of 7 different defined event classes. Data labelling was done by a person sitting in a car and registering each road anomaly manually in real-time.

2.3 Machine learning

Detection algorithms are the most differing aspect of the previous work on this topic. Some papers have described machine learning approaches to road quality classification, however,

others have opted for different solutions. This subchapter aims to provide an overview of the different feature sets and techniques used in machine learning approaches.

For classification, Seraj et al. [8] used support vector machines with RBF kernel. Features were extracted from time domain, frequency domain and wavelet decomposition. Frequency domain representation of the signal was computed using a FFT algorithm with a Hamming window function and the signal decomposition was done using Stationary Wavelet Transform [14]. Time domain features included: standard deviation, mean, variance, root mean square, peak to peak, mean of absolute values, zero-crossing rate, correlation between all axes, tilt angles, signal magnitude area and waveform length. Extracted features in frequency domain included: mean frequency, energy of the frequency bands and median frequency. Features extracted from wavelet decomposition included: standard deviation, absolute mean, variance and energy for every level of detail. It is also mentioned, that experimentations were done with raw signal and demodulated signal data, in order to see which feature sets provide the best detection results. After feature extraction, a 2-step classification process was described. In the first step, all the windows were processed and those that contained road anomalies were taken into the second step. In the second step, another classifier attempted to classify the type of road anomaly that was present in each window. Training data consisted of 3066 windows and a sliding window with an overlap of 66%, was used. 10-fold cross-validation was performed to train and test the classifier and the results revealed that using demodulated features resulted in a more accurate anomaly classification than training on features from the raw data. Overall accuracy of correctly detecting severe road anomalies was concluded to be around 90%.

Hoffmann et al. [10] also uses a machine learning for classification. Two types of road surface classification processes were carried out: direct road surface classification and bump detection based classification. Direct road surface classification consisted of classifying a road surface as one of three categories: *smooth*, *bumpy* or *rough*. Features were extracted from GPS and accelerometer data and included: speed, inclination, acceleration mean, acceleration variance and acceleration standard deviation. In the direct road classification approach, roads were divided into sets of segments. For each segment that needs to be classified, previous segments' features are also taken into account. A feature set optimization was also carried out, with the purpose of

finding those features which increase classification accuracy the most. It was found, that speed and inclination confused the classifier or didn't contribute to the learning algorithm. Two machine learning algorithms were used for classification: KNN and Naive Bayesian Classifier. 10-fold cross-validation was performed, in order to test the classifiers on evaluation data. It was noted that while Naive Bayes and KNN algorithms performed mostly similar, a slight increase was seen with the KNN. However, the overall accuracy of 78% proved to be not satisfactory. In bump detection based classification approach, the classifier had to distinguish between a smooth road and a road that has a bump or some other similar anomaly. Here, classification accuracy was a lot better, with an accuracy of around 98%.

Eriksson et al. [13] used a signal processing and machine learning approach to detect and classify several specific road anomalies. The paper describes the issue of having other unwanted anomalies manifest in the accelerometer signal, for instance abrupt turning or stopping. Different characteristics, such as speed, and filters, such as a high-pass filter, were used to reject such events. The training of the learning algorithm is based on the peak *X* axis and *Z* axis acceleration values and the instantaneous velocity of the car. To help remove false positives and increase the overall accuracy of correct anomaly detection, the paper describes a process where an anomaly is reported only when several other detectors have also detected an anomaly in the same spot. Overall, the paper concludes, that by using training data, which had been carefully examined, the described pothole detection system achieved a false positive rate of 0.2% in controlled experiments.

2.4 Conclusion

This thesis aims to build on some of the aspects from previous work, as well as combine others. The approach of dividing road surface quality into three distinct categories for classification, was attempted by Hoffmann et al. [10] with fairly modest results. This work explores this approach from a perspective of car-based transport.

It has been demonstrated by previous work, that a supervised learning approach to road surface quality and anomaly detection can be successful. This technique is also applied in this thesis. A SVM with RBF kernel is used, which has been successfully used previously by Seraj et al. [8].

Regarding the data collection: similarly to some previous works, an Android smartphone and a custom developed data collection application (Road Surface Data Collector) are used. An additional feature is implemented into Road Surface Data Collector, that allows the device to be in any orientation, while collecting data. The calculation of Euler angles, with the intention of reorienting the acceleration vector, has been previously mentioned by Astarita et al. [12]. With Road Surface Data Collector, the transformation is achieved with the use of a rotation matrix, based on the output of gravity and magnetic field sensors.

The next chapter delves into the methodology of the Road Surface Data Collector application and the road surface detection algorithm.

Chapter 3: Methodology

3.1 Introduction

This chapter provides an overview of the design, architecture and implementation of the Road Surface Data Collector application as well as the road surface quality detection algorithm. The chapter is divided into two main subchapters. Subchapters 3.2 focuses on the Road Surface Data Collector application, while subchapter 3.3 focuses on the road surface quality detection algorithm.

3.2 Road Surface Data Collector application

Road Surface Data Collector is an Android application, programmed in Java[2] programming language and designed to run on smartphones or tablets with Android version 4.0.3[3] or greater. This version of Android was chosen, because according to official data gathered from Google Play Store[4] visits, it covers a large percentage of the active Android devices while still providing sufficient features for this application's purpose [16]. However, an important thing to note, is that the version of Road Surface Data Collector used and described in this thesis was built specifically for the purpose of this thesis and was tested and used on a single smartphone and therefore serves only as a proof of concept and not as final software.

Road Surface Data Collector relies heavily on the Android sensor framework, by using the gravity and linear acceleration sensors, which are accessible from the framework. Depending on the device model and manufacturer, these sensors can be either software-based or hardware-based. Software-based sensors are derived from one or several hardware sensors while hardware-based sensors get their data directly from the hardware sensor. Motion sensors like the gravity sensor and linear acceleration sensor, can provide developers with information that can be very useful for certain specific tasks. A gravity sensor measures acceleration that is due to

[2] http://www.oracle.com/technetwork/java/index.html
[3] Android version 4.0.3, Codename "Ice Cream Sandwich", API level 15 [15]
[4] https://play.google.com/store

gravity while excluding all other, which can be used to calculate the device's orientation in relation to Earth. A linear acceleration sensor provides the opposite effect by providing acceleration data, from which the force of gravity has been excluded. This can be used for tasks such as movement detection. [17, 18]

In the case of gravity and linear acceleration sensors, they usually require the device to have a gyroscope sensor and can therefore be unavailable unless the device has one, which is the case with the AOSP[5] implementation of the gravity and linear acceleration sensors. However, it also has to be noted that Android smartphone and tablet manufacturers can provide their own sensor implementations and therefore the availability and configuration for the gravity, linear acceleration and other sensors can differ across different devices. [18]

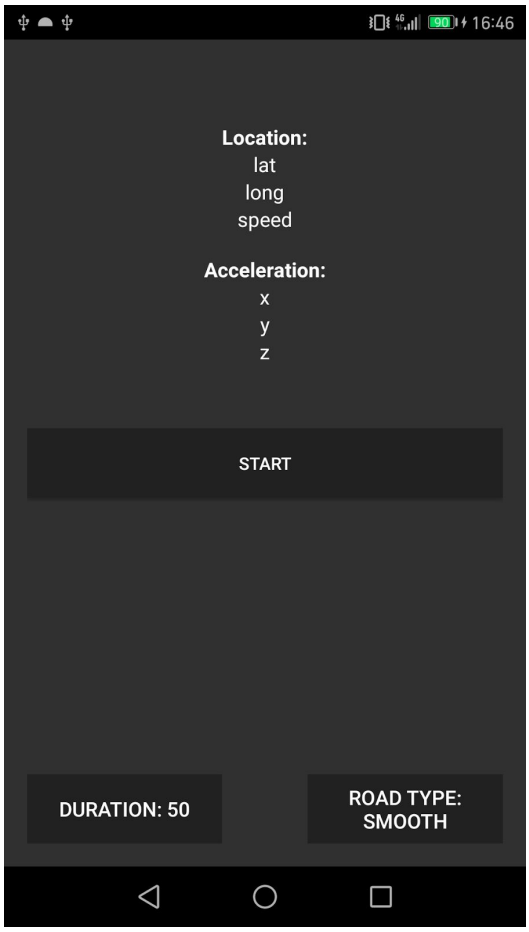


Figure 1. *The main view of Road Surface Data Collector.*

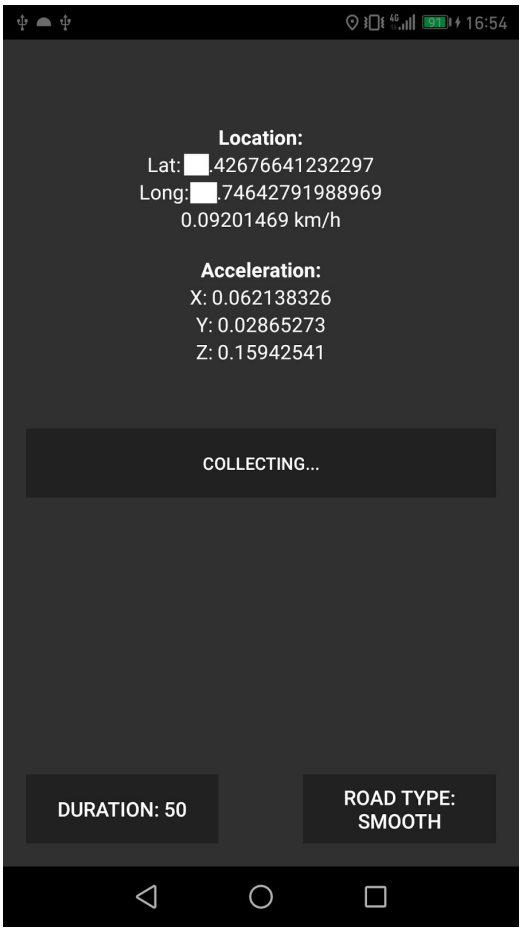


Figure 2. *The main view, while collecting data.*

[5] https://source.android.com/

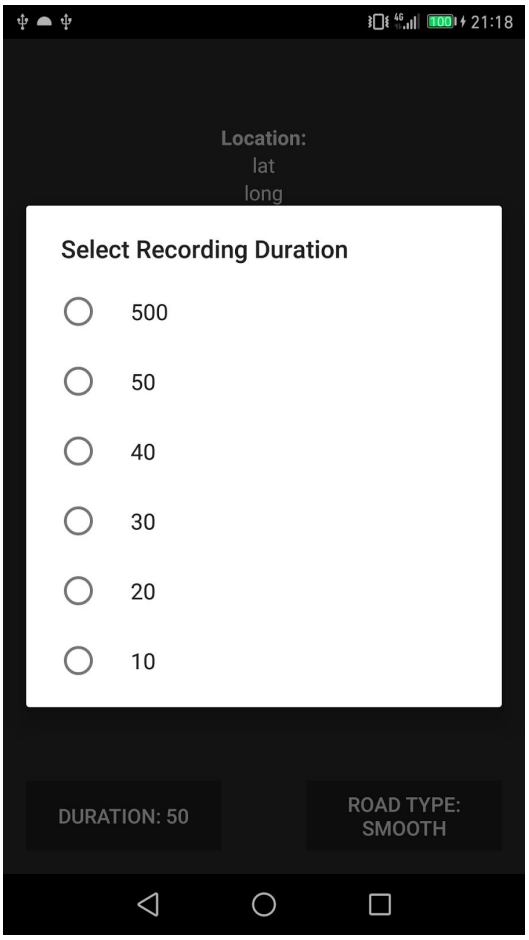


Figure 3. *Selection window for recording duration.*

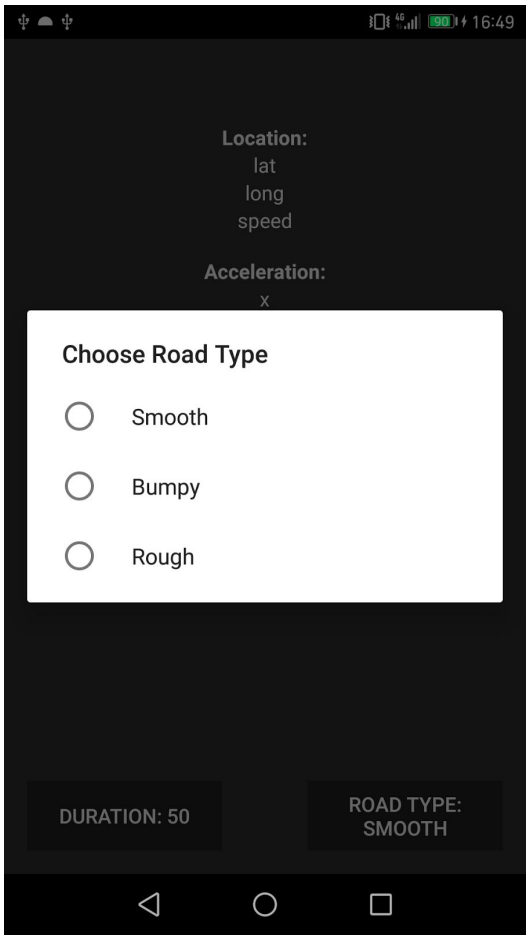


Figure 4. *Selection window for road type.*

3.2.1 User interface

The Road Surface Data Collector features a simplistic user interface. ***Figure 1*** illustrates the main view of the application in an idle state while ***Figure 2*** displays the same view in a state where data is being actively recorded. ***Figure 3*** and ***Figure 4*** demonstrate the selection windows for the recording duration and road surface type, respectively.

If data is not being actively recorded, then a new recording can be initiated by pressing the "Start" button located in the centre of the main view, as viewed in ***Figure 1***. When data collection has been started, information such as the current latitude, longitude and transformed acceleration values (see subchapter 3.2.4) are also visible in the main view, mainly for testing

purposes, as illustrated in ***Figure 2***. From the duration selection window (***Figure 3***), the user can select between different recording durations in seconds. From the road surface type selection window, the user can select the type of road surface that is going to be driven over, while recording (***Figure 4***). This feature is used to record labelled acceleration data of different road surface types which would then be used by the road surface quality detection algorithm.

3.2.2 Data recording

In addition to the gravity and linear acceleration sensors mentioned previously, the application also uses data from the geomagnetic field sensor. Data from the gravity and geomagnetic field sensors is used to transform the raw linear acceleration values from the device coordinate system into world's coordinate system. This process is explained with greater detail in subchapter 3.2.4.

When Android receives an update from a sensor, the system creates a new *SensorEvent*, which is a Java object, that represents a sensor event and holds, among other information, the values received from the sensor [19]. To initiate and monitor *SensorEvent*s, the Android *SensorManager*[6] class provides four different intervals:

1. *SENSOR_DELAY_FASTEST;*
2. *SENSOR_DELAY_UI;*
3. *SENSOR_DELAY_GAME;*
4. *SENSOR_DELAY_NORMAL.*

However, these intervals can be inconsistent and therefore, for the Road Surface Data Collector, a slightly different solution is used. In order to bring more consistency to data recording, all of the sensors are run at their fastest interval (*SENSOR_DELAY_FASTEST*) and every time a listener[7] receives a new *SensorEvent*, the previous value from that sensor is overwritten. Meanwhile, these saved values are sampled at a frequency rate of around 10 Hz. Running the sensors at their fastest interval reveals, that when tested on a Huawei P9 smartphone with Android version 7.0, the interval between consecutive *SensorEvent*s from each sensor is low enough, that while the sampling occurs at 10 Hz, the time difference

[6] https://developer.android.com/reference/android/hardware/SensorManager.html
[7] A type of class that gets notified when a certain, predetermined type of event or state change has happened. Further action can then coordinated.

$$difference = SamplingTimestamp - LastSensorEventTimestamp \quad (1)$$

ranges roughly from 9–30 ms. In addition to sampling the sensor values, GPS data is also gathered, specifically latitude and longitude. GPS data, however, is gathered once every second.

3.2.3 Data output

When data is being gathered, it is simultaneously written to a CSV format output file, which is saved to device's primary external storage. An output file consists of the following data:

1. Timestamps in milliseconds since January 1, 1970 00:00:00.0 UTC;
2. Raw linear acceleration values along the device's X, Y and Z axes (for testing purposes);
3. Linear acceleration values transformed to world coordinate system;
4. Longitude and latitude, acquired from the GPS;
5. Road type (labelled data).

The data is collected and outputted in such a way, that time based windows[8] of data are formed. Since GPS data is collected once per second and acceleration data roughly once every 100 ms, the GPS data collection points act as boundaries around the acceleration data. Each window contains 11 data points in total and has a start point and an end point, represented by the GPS coordinates. All 11 points of acceleration data, gathered during the 1 second long time-frame between two GPS location sampling points, are considered to be in the same window. This structure enables to divide roads into segments and therefore each road segment can be classified individually. Since data is being continuously recorded, two neighboring road segments (windows) always share a single point of acceleration and GPS data because of overlapping start and end points. This structure can be visualized on a map, as illustrated in ***Figure 5***.

[8] An abstract block of data, that contains all the data points from a specific time frame.

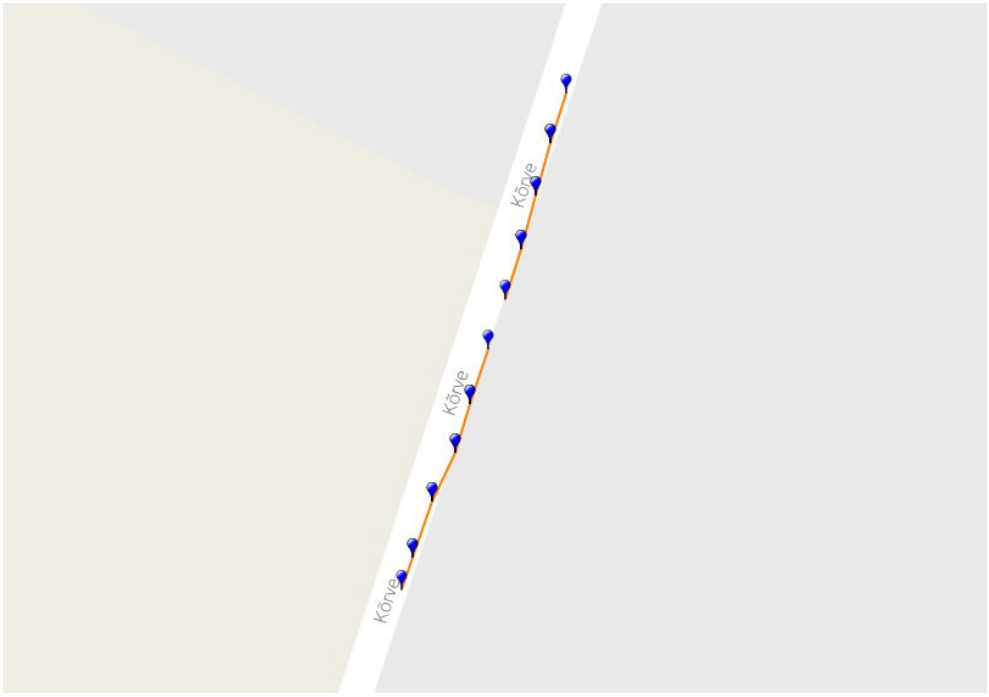


Figure 5. *Visualizing data windows on a map. The difference in segment lengths is caused by GPS inaccuracy.*

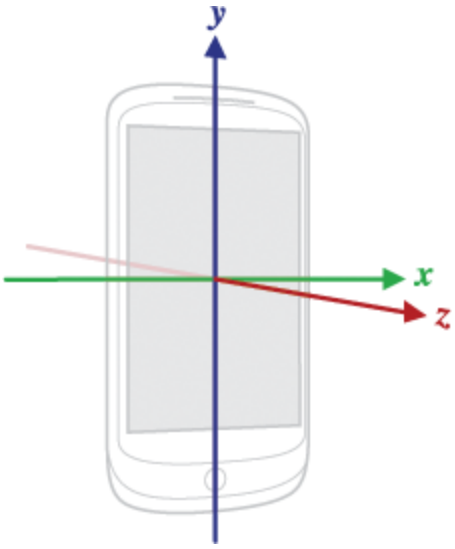


Figure 6. *Device coordinate system* [19].

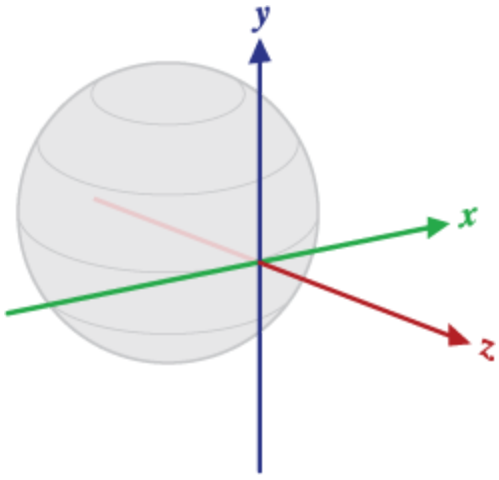


Figure 7. *World (Earth) coordinate system* [18].

3.2.4 Linear acceleration transformation process

Definition 1. Rotation matrix *is a type of matrix that is used to execute a rotation. A rotation matrix can provide a rotation around a single axis or multiple axes, when rotation matrices for different axes have been multiplied together.* [20]

The most important, of all the acceleration data outputted by the Road Surface Data Collector, is the linear acceleration data along the Z axis of the world coordinate system (illustrated in ***Figure 7***). In the literature (see Chapter 2) it has been demonstrated, that acceleration data, which is recorded on the axis that is perpendicular to the ground plane, can represent the surface of a road. However, having to carefully orient the phone to line up one of the device's axes (illustrated in ***Figure 6***) with the Z axis of the world coordinate system, can be tedious. Transforming the linear acceleration values from the device coordinate system into world coordinate system allows the device to be in any orientation while collecting data. This grants more freedom in real world scenarios.

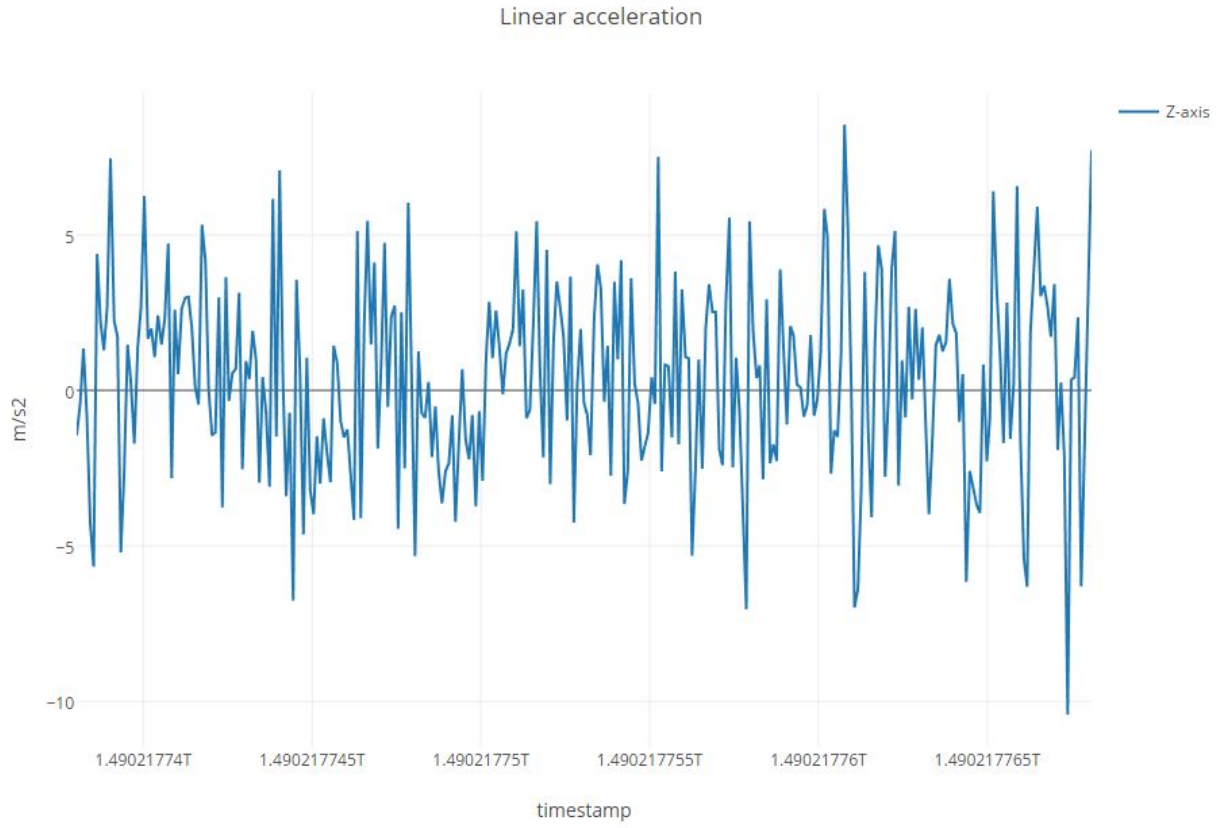
In order to achieve this transformation, the Road Surface Data Collector makes use of the Android sensor framework, in particular the *SensorManager* class, that provides methods to calculate the rotation matrix (***Definition 1***), which can then be used to transform a vector from device coordinate system into world's coordinate system.

First, data from the gravity, linear acceleration and geomagnetic field sensors is gathered and stored. By using this data, the *SensorManager* class static method *getRotationMatrix*[9] is used to calculate the rotation matrix R, which contains rotations around all axes, and the inclination matrix I, which is a rotation around X axis. The method takes 4 arguments:

1. The rotation matrix R (a float array of length 9 or 16);
2. Inclination matrix I (a float array of length 9 or 16);
3. Three dimensional gravity vector (from gravity sensor);
4. Field strength data (from geomagnetic field sensor).

[9] https://developer.android.com/reference/android/hardware/SensorManager.html#getRotationMatrix(float[], float[], float[], float[])

The calculated rotation and inclination matrices are in row-major[10] order and stored in the corresponding float arrays, passed as arguments (R and I, respectively). Matrix R is then inverted and multiplied with the linear acceleration vector v (in device coordinate system), resulting in a vector that is in world coordinate system. The inversion and multiplication are done using methods *invertM* and *multiplyMV*, respectively, from the *android.opengl.Matrix*[11] class. [21]



[10] A structure of storing multidimensional arrays, where each consecutive element of a row is positioned after the previous element.
[11] https://developer.android.com/reference/android/opengl/Matrix.html

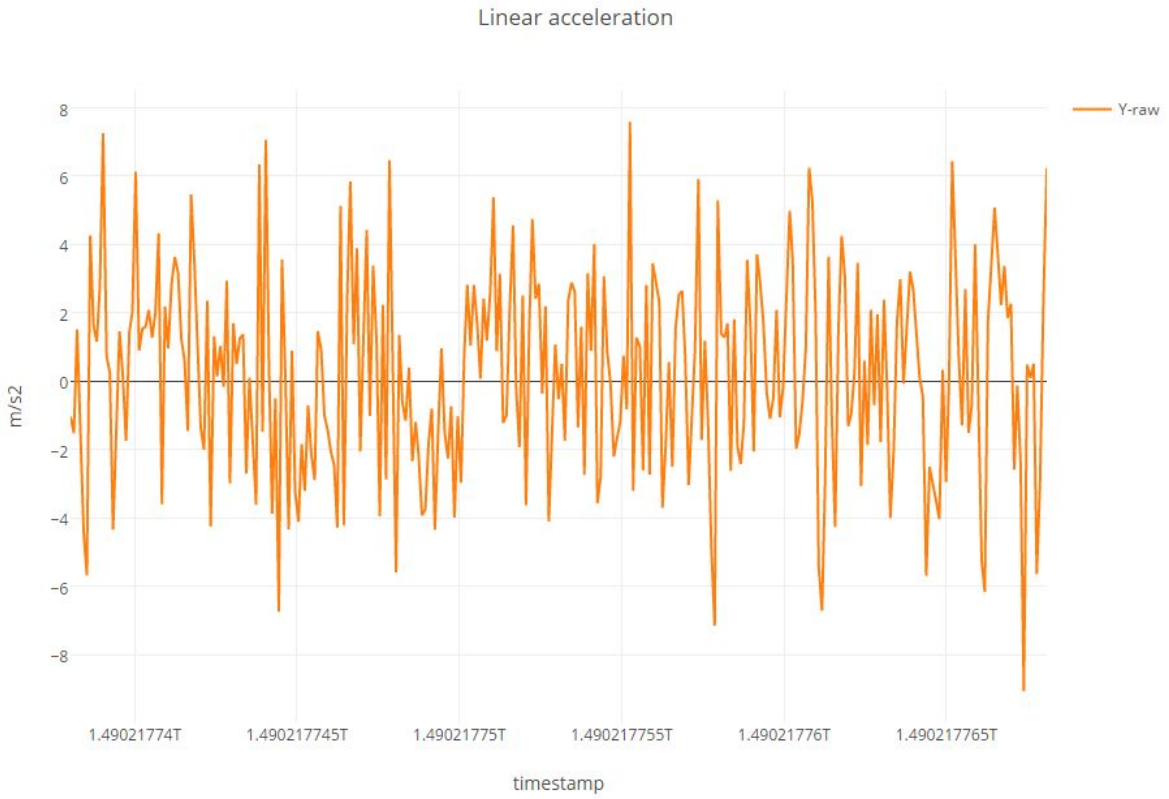


Figure 8. *Linear acceleration sample. Z-axis line (upper image) represents the transformed linear acceleration along the Z axis in world coordinate system and Y-raw line (bottom image) is the raw linear acceleration along the device's Y axis.*

In order to verify that the implementation provides satisfactory results, acceleration data samples were gathered in a way that the Y axis of the device lined up with the Z axis of the world coordinate system (can be observed in ***Figure 6*** and ***Figure 7***). This was achieved by fixing the device to a car in approximately the correct orientation and driving on a road. The reason for approximating the orientation of the device was that the setup, used to fix the device to the car, was not flexible enough to allow fine calibration of the device's angle. However, the data can still provide insight into the performance of the linear acceleration vector transformation. ***Figure 8*** visualizes one of the data samples. Values of both acceleration signals largely follow the same peaks, however some differences can be observed. These differences are likely to be due to the approximation of the device's angle in the car. Furthermore, even in ideal orientation, minor differences can be expected due to random sensor noise.

Alternatively, instead of using the *getRotationMatrix* to calculate the rotation matrix, it is also possible to use *getRotationMatrixFromVector* method from the same class. This method requires data from the rotation vector sensor, which can be either hardware-based or software based. However, this sensor, together with the method *getRotationMatrixFromVector*, generally provides more accurate orientation data than the use of *getRotationMatrix* method together with geomagnetic field and gravity sensors [21]. This is likely due to the sensor making better use of the gyroscope. The implementation of this alternative can otherwise be considered analogous to the previously explained implementation, currently used in Road Surface Data Collector. Nonetheless, because of the satisfactory results, achieved in testing the previously explained solution, this alternative is not implemented within the scope of this thesis. However, it is something to consider in future work for the purpose of increasing the accuracy of the transformation in devices, that provide a rotation vector sensor.

3.3 Road surface quality detection algorithm

Definition 2. Supervised learning *is a machine learning technique that uses learning data, which is made up of pairs of values. Each pair has a set of input variables, which can be denoted as x and a secondary output variable, denoted Y. The idea is, that an algorithm will learn, based on the learning data, to predict the Y variable for new, previously unseen input data.* [22]

The road surface quality detection algorithm implements the technique of supervised learning (***Definition 2***). Three different types of acceleration data are used for the supervised learning process. These types of data represent the different levels of road surface quality. *Smooth* represents the best type of road, while *rough* represents a poor quality road and *bumpy* falls between the two. Chapter 4 goes into greater detail on how the data for each of the road types was collected.

As for the classifier, the OpenCV[12] implementation of a SVM is used. For parameters, the type of SVM formulation is *C_SVC*, which supports n-class classification (where $n > 1$) and the chosen kernel type is RBF [23]. Other parameters are left at their default values.

[12] http://opencv.org/

The algorithm implementation is done in the Python[13] programming language. This language was chosen because of its excellent support in regards to science and research. Several extensive libraries, packages and IDEs are available for Python, which help with the process of creating applications with scientific purposes. In this thesis, NumPy[14], OpenCV library and Plotly Python Library[15] are used. NumPy is used for data structuring aspects and for the implementation of FFT algorithm while OpenCV is used for the implementation of SVM. The Plotly Python library is used to create visualizations of machine learning results.

The detection algorithm itself consists of processing the data into usable objects, extracting features from the data and training a SVM for classification. The following subchapters provide an overview of these aspects.

3.3.1 Data preprocessing

First, the chosen data files are read in and divided into Python dictionary objects. Since each dictionary object represents a single window of data, features are also extracted (see subchapter 3.3.2) from each window and added back to their corresponding dictionary object. The use of dictionary objects provides a simple way of extracting data, when necessary, by querying the windows (dictionaries) with the appropriate keys. In total, three lists of data windows are created, one for each road type and the information, contained in every dictionary (data window), is as follows:

1. Window starting point as latitude and longitude;
2. Window end point as latitude and longitude;
3. List of linear acceleration values in device's coordinate system;
4. List of linear acceleration values in Earth's coordinate system;
5. List of timestamps in milliseconds since January 1, 1970 00:00:00.0 UTC;
6. Road type (in the case of labelled data).
7. Extracted features.

[13] https://www.python.org/
[14] http://www.numpy.org/
[15] https://plot.ly/python/

After the lists are created, training and testing data structures can be compiled and used in the machine learning stage (see subchapter 3.3.3).

3.3.2 Feature extraction

Definition 3. *In machine learning,* ***feature extraction*** *is a process of calculating or determining certain attributes (**features**) from given input data* [22]. *In the case of numerical data, a generic example would be the calculation of the mean value of all the numbers in the given data set.*

In order to effectively use the SVM classifier, a feature extraction process (***Definition 3***) is required. Within the scope of this thesis, extracted features can be divided into two categories: time domain features and frequency domain features. All features are extracted from the Z axis linear acceleration data (in world coordinate system, see ***Figure 7***) and calculated separately for every data window. The following is a list of all the features that are extracted:

1. Mean (time domain);
2. Mean (over absolute values; time domain);
3. Variance (time domain);
4. Sum of the absolute values of maximum amplitude and minimum amplitude (time domain);
5. Energy (frequency domain);
6. Entropy (frequency domain).

Features, that are described in this chapter, were chosen as a result of analysing related work on the topic of road quality detection (see Chapter 2) as well as on the topic of user activity detection, using accelerometer data. For example, Preece et al. [24] and Bao and Intille [25] mention the use of frequency domain energy and entropy for activity recognition. Other features mainly focus on the statistical analysis of the acceleration signal.

However, in order to calculate frequency domain entropy, a conversion from time domain to frequency domain has to be performed first. A FFT algorithm can be used for this purpose. The current paragraph is based on explanations provided by Press et al. [26], regarding the FFT and DFT algorithms. A fast Fourier Transform (FFT) algorithm calculates the DFT of a finite

sequence of complex numbers $x_0, x_1, ..., x_{N-1}$. The output of DFT is another sequence of complex numbers $X_0, X_1, ..., X_{N-1}$, with the same length N. DFT itself can be defined as

$$X_k = \sum_{m=0}^{N-1} x_m \cdot e^{\frac{2i\pi km}{N}}, \; k = 0, 1, ..., N-1 \tag{2}$$

and its inverse as

$$x_m = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{\frac{-2i\pi km}{N}}, \; m = 0, 1, ..., N-1 \tag{3}$$

where N is the length of the input sequence. While DFT can be calculated directly, FFT algorithms are mostly used in implementations, because of their reduced computation complexity of around $O(N \log_2 N)$ as opposed to $O(N^2)$, when DFT is calculated directly. The lower complexity is achieved by optimizations in the DFT calculation process.

In this thesis, the NumPy FFT function[16] is used, for which the DFT and its inverse are defined similarly to equations (2) and (3), with the only difference being the opposite signs of exponents [27].

3.3.2.1 Time domain features

The list of real numbers, from which all the time domain features are calculated from, can be defined as $A = (a_0, a_1, ..., a_{10})$, where $a_i \; (i = 0, 1, ..., 10)$ is a recorded linear acceleration value. All the time domain features with corresponding equations, that are used for calculation, can be listed as follows:

1. Arithmetic mean, calculated as

$$\bar{a} = \frac{1}{N} \sum_{i=0}^{N-1} a_i \tag{4}$$

   where N is the length of A;

2. Arithmetic mean, calculated with absolute values using the equation

$$\bar{a}_{abs} = \frac{1}{N} \sum_{i=0}^{N-1} |a_i| \tag{5}$$

   where N is the length of A;

[16] https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fft.html

3. Variance, defined as

$$\sigma^2 = \frac{\sum_{i=0}^{N-1} (a_i - \overline{a})^2}{N} \quad (6)$$

where N is the length of A and $\overline{a}$ is arithmetic mean, calculated by using equation (4);

4. Sum of the absolute values of minimum amplitude and maximum amplitude, which can be defined as

$$sum_{minmax} = |\min A| + |\max A| \quad (7)$$

where A is the previously defined list of linear acceleration values.

3.3.2.2 Frequency domain features

By using a FFT algorithm on a list of linear acceleration values $A = (a_0, a_1, ..., a_{10})$ (linear acceleration signal) the frequency domain representation of the signal is gained. The result is a list of complex numbers $X = (x_0, x_1, ..., x_{10})$. Based on this data, the following features are calculated:

1. Energy in frequency domain can be defined as the sum of the squared magnitudes of elements $x_0, x_1, ..., x_{10}$, divided by the length of X (see equation (8)) [25]. Furthermore, Parseval's theorem dictates, that the energy in frequency domain can be considered identical to the energy in time domain [28]. As described by Press et al. [26], in its discrete form, Parseval's theorem can be defined as a relation

$$\sum_{i=0}^{N-1} |a_i|^2 = \frac{1}{N} \sum_{j=0}^{N-1} |x_j|^2 \quad (8)$$

where N is the length of lists A and X (equal in length), a_i is an element in the original time domain list A and x_j is an element in X, which is the DFT of the original list A. In the solution provided by this thesis, the energy feature is calculated in frequency domain and therefore, the right side formula of equation (8) is used;

2. Frequency domain entropy calculation is described by Zhang et al. [29] and can be epitomized in three main steps:

   2.1. Calculate power spectral density (power spectrum)

$$\widehat{P}_i = \frac{1}{N}|x_i|^2,\ i = 0, 1, ..., N-1 \tag{9}$$

where x_i is an element in X and N is the length of X;

2.2. Find power spectral density distribution function

$$p_i = \frac{\widehat{P}_i}{\sum_{j=0}^{N-1} \widehat{P}_j},\ i = 0, 1, ..., N-1 \tag{10}$$

where N is the length of X;

2.3. Calculate entropy

$$H = -\sum_{i=0}^{N-1} p_i \ln p_i \tag{11}$$

where N is the length of X.

3.3.3 Supervised learning

A separate learning and testing function was implemented, which, as arguments, takes in training data and testing data windows. Both of these data structures include the extracted features, described in the previous subchapter 3.3.2, and the road surface types, for each window. For both the training and testing data, the calculated features and desired outputs of each window are then added into separate input and output data structures, representing the *x* and *Y*, as described in ***Definition 2***.

After all the necessary data structures have been created, the training data input and output structures are used to train a SVM classifier. The performance of the classifier is then tested by using it to predict the type of road surface quality for each of the windows in the test data input structure. Upon finishing, the method returns the predicted values as well as the correct labels. These values can then be used to calculate the accuracy of the classifier.

3.4 Conclusion

This chapter provided insight into the methodology of this work by explaining various aspects of the Road Surface Data Collector Android application as well as the road quality detection

algorithm implementation. The next chapter explains the data collection process and the results achieved by evaluation the detection on the collected data.

Chapter 4: Results and analysis

4.1 Introduction

This chapter aims to give an overview of the various aspects of data collection, as well as the process of selecting the best performing feature sets. Furthermore, the results of a chosen feature set are analysed further. This chapter is divided into three main subchapters: subchapter 4.2 focuses on the aspects of data collection, subchapter 4.3 delves into the comparison of different feature sets and subchapter 4.4 provides further performance analysis for a selected feature set.

Figure 9. *Photo of a data collection setup. Since data was collected across different sessions, the positioning of the phone varied slightly each time. However, the general location was always in the centre, roughly above the dashboard.*

4.2 Data collection

4.2.1 Collection setup

The data collection setup consisted of a smartphone running the Road Surface Data Collector application while held in place, in a car, by a smartphone mount. This setup can be observed in ***Figure 9***. The smartphone car mount, visible in ***Figure 9***, was further fixed in place by a piece of double-sided tape, that was put between the the bottom of the mount and the car dashboard. This minimized the vertical movement of the mount, which is important for keeping the acceleration data free of noise. Furthermore, since that specific smartphone mount featured a flexible arm (middle part) it was bent in a way that the mount's arm would be under tension and would apply force towards the dashboard, further decreasing the probability of any vertical movement. However, any other movement constraints were not introduced, in order to keep setup reasonably easy to reproduce in a real world scenario.

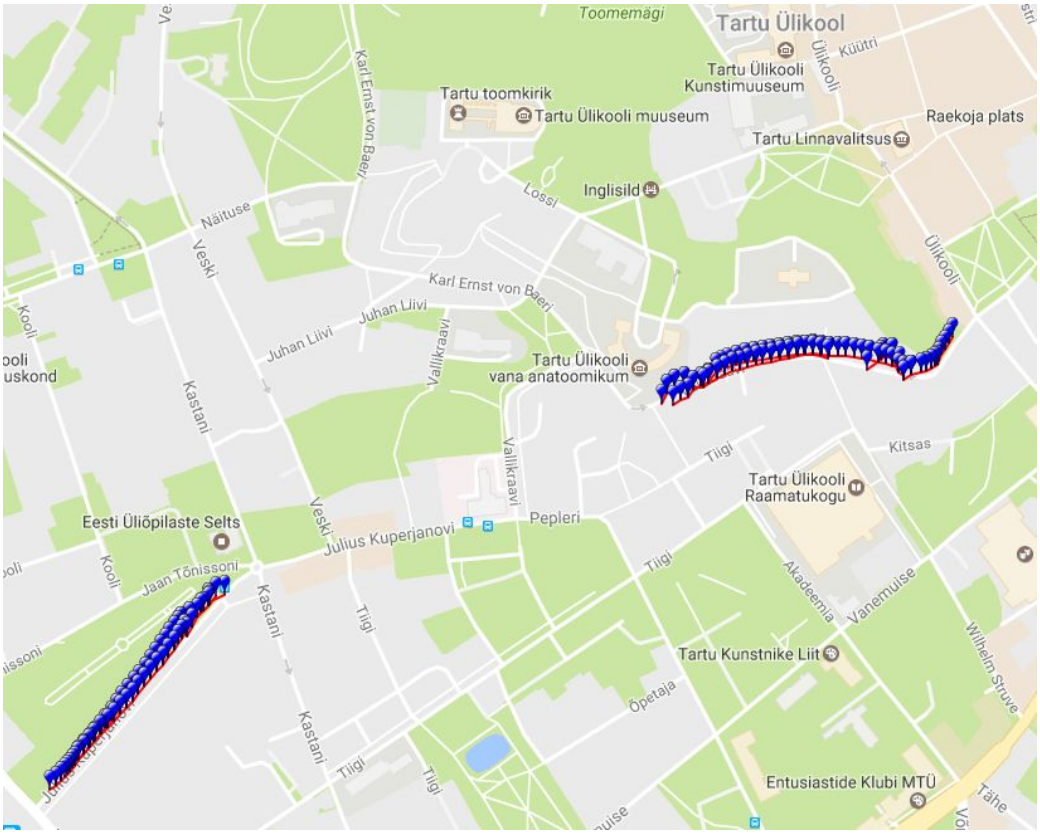


Figure 10. *Areas, where data collection was performed. The lower image illustrates an area just outside of Tartu while the upper image illustrates the downtown area of Tartu. Blue markers represent the start and end points for road segments (data windows). Colors of the segments range from green to red, describing the type of surface quality it was sampled as: green - smooth, orange - bumpy, red - rough.*

4.2.2 Chosen roads

In order to decide on which roads to record acceleration data, for each road surface quality type (*rough*, *bumpy* and *smooth*), prior driving experience and analysis was taken into account. Different roads were driven on and recorded, with the specific intent of finding suitable candidates for the three surface types. The recorded acceleration signals were manually analysed for peak values[17]. A factor, that also had to be taken into account, was that a chosen road section would, ideally, have to be fairly consistent and long enough, so that enough data could be gathered.

It was decided, that for the *smooth* surface type, a flat asphalt road, as seen in ***Figure 11***, can provide a satisfactory example. For the *rough* surface quality type, two cobbled roads were chosen. One of the roads can be observed in ***Figure 12***. These cobbled roads were chosen because they can provide fairly consistent acceleration data with high peak values and at the same time, are easily accessible in the city of Tartu. ***Figure 13*** illustrates a type of gravel road, that was chosen for the *bumpy* surface type. This road was chosen, because it features fairly shallow potholes with moderate frequency, which, when driven over, provide acceleration data with peak values that generally fall between *smooth* and *rough* surface types. All of the data was recorded at speed of around 30 km/h. The data collection routes can be seen in ***Figure 10***.

Figure 11. *A flat asphalt road, used as a base road for the smooth road surface type.*

[17] In the context of this thesis, a peak value is considered high, when it is larger than or around 4 m/s^2 and low, when it is around 1 m/s^2 or lower. Peak value itself can be defined as the largest value in a finite sequence of acceleration values (absolute values).

Figure 12. *A cobbled road, used as a basis for the rough road surface type.*

Figure 13. *A gravel road with potholes, used as a basis for the bumpy road surface type.*

While roads visible in ***Figure 11*** and ***Figure 12*** provide fairly consistent data, the road that was chosen for the *bumpy* surface type, can often contain either high or low peak values. For this reason, the data for *bumpy* surface type was manually processed after collection, in order to remove data windows, that contained such peak values. Furthermore, a few data windows, that contained too low peak values, were removed from *rough* surface data and some windows, which contained too high peak values, were removed from *smooth* data. In total, 126 *rough*, 125 *bumpy* and 142 *smooth* data windows were counted after processing.

It is important to acknowledge, that while these roads were chosen for the purpose of this thesis, some cars are likely to perform differently to others, when driven over potholes or on cobbled

streets. This can be due to factors such as differences in suspension setups and rim or wheel sizes. As a whole, this is an aspect, that likely needs addressing in a future work.

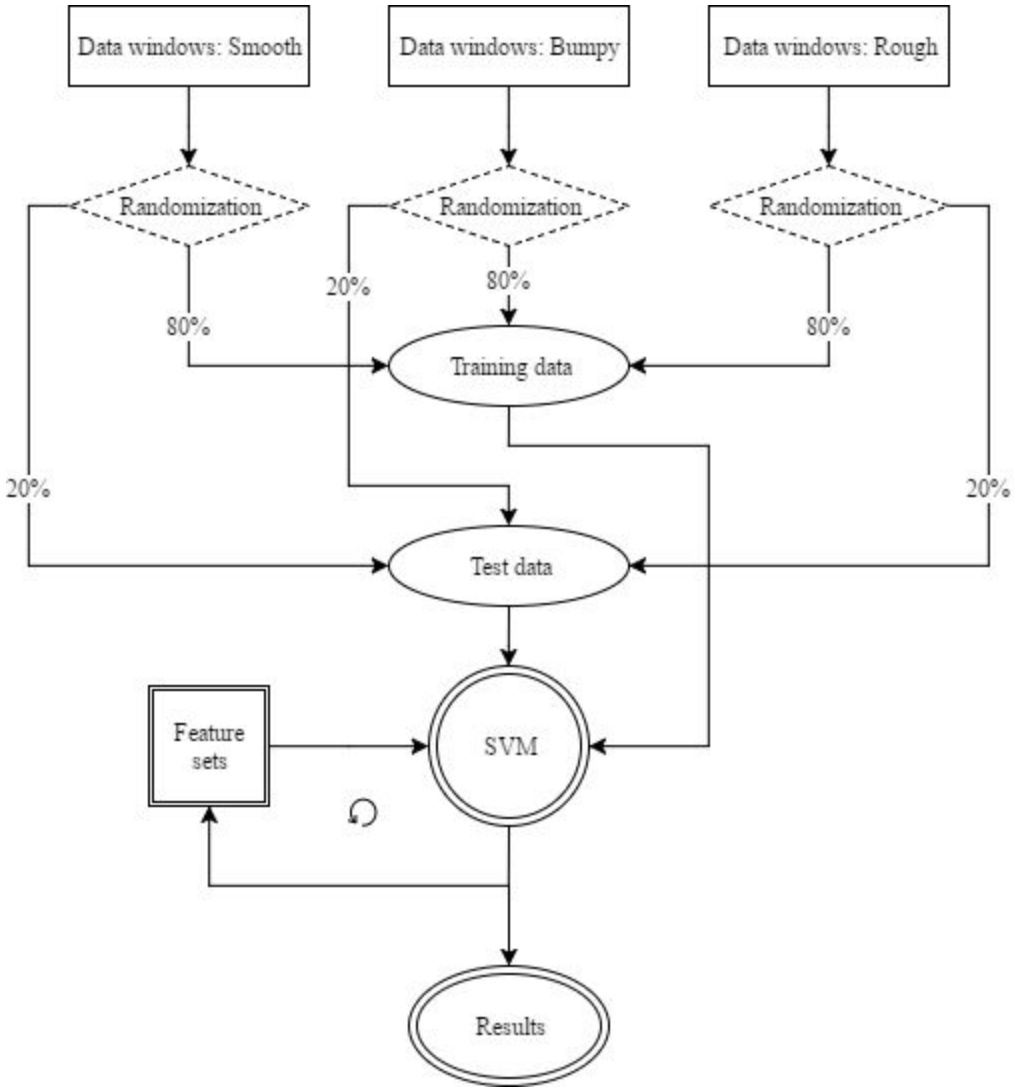


Figure 14. *A diagram, describing a single evaluation cycle.*

4.3 Performance comparison of feature sets

When deciding which features to extract, there is often no clear way of knowing beforehand, how different features will impact the performance of a classifier. One solution is to create subsets of features. If every subset of the complete feature set is tested, a much better

performance overview can be created. This overview can then be used to select feature sets, that provide the best performance.

In this thesis, the complete feature set consists of 6 features. Combinatorics were used to create subsets of this feature set. The performance of the detection algorithm was then tested with 500 evaluation cycles. During every evaluation cycle, the data structures, that contained the data windows, were randomized and divided into separate training and testing structures, with 80% as training and 20% as testing data. Using this data, the performance of every feature set was tested by training and testing a SVM classifier. A single evaluation cycle is illustrated in ***Figure 14***. ***Figure 15*** illustrates the classifier's average accuracy for every feature set and road surface quality type, across the 500 evaluation cycles. ***Figure 16*** presents the standard deviations of classifier accuracy over the same evaluation cycles. It must be noted, that since this evaluation process is based on the randomness of the training and testing data, slight variations in results can be observed, when doing multiple evaluations.

Based on ***Figure 15*** and ***Figure 16***, it can be concluded, that the used dataset is predictable at a satisfactory level with most of the feature sets, when tested with the aforementioned technique. However, some differences in accuracy and standard deviation, with *bumpy* and *rough* surface types in particular, can be observed. Upon analysing the results, several feature sets were found to provide satisfactory results, one of which is (*mean*, *variance*, *mean_abs*, *sum_minmax*, *entropy_frqdm* (*frequency domain entrop*y)). This set of features, among others, provides satisfactory classifier accuracy for every road surface type, while also keeping the standard deviation of accuracy, for both *bumpy* and *rough* surface types, lower than most other feature sets.

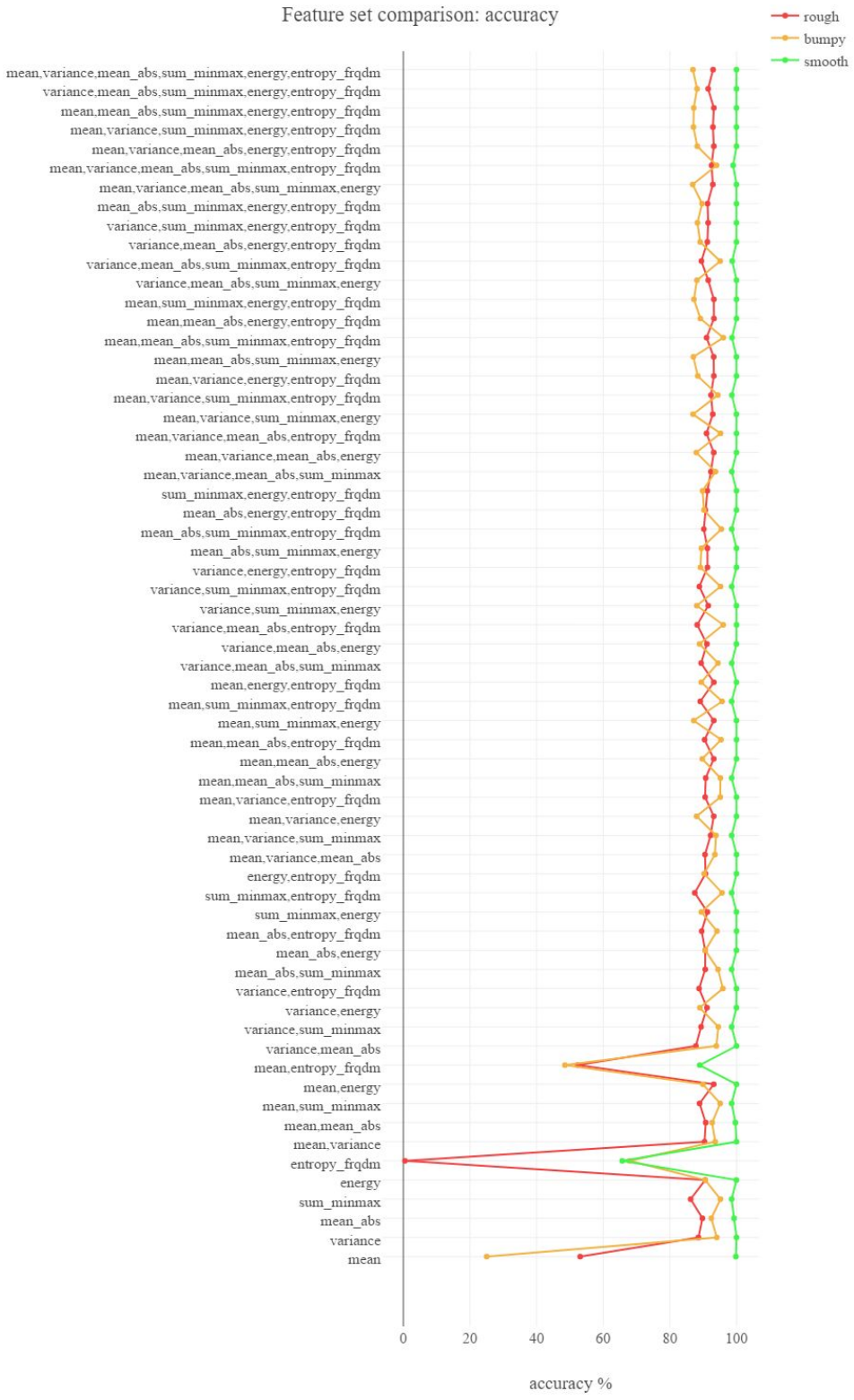


Figure 15. *A graph, describing the classifier's average accuracy over 500 evaluation cycles for every road surface type and feature set.*

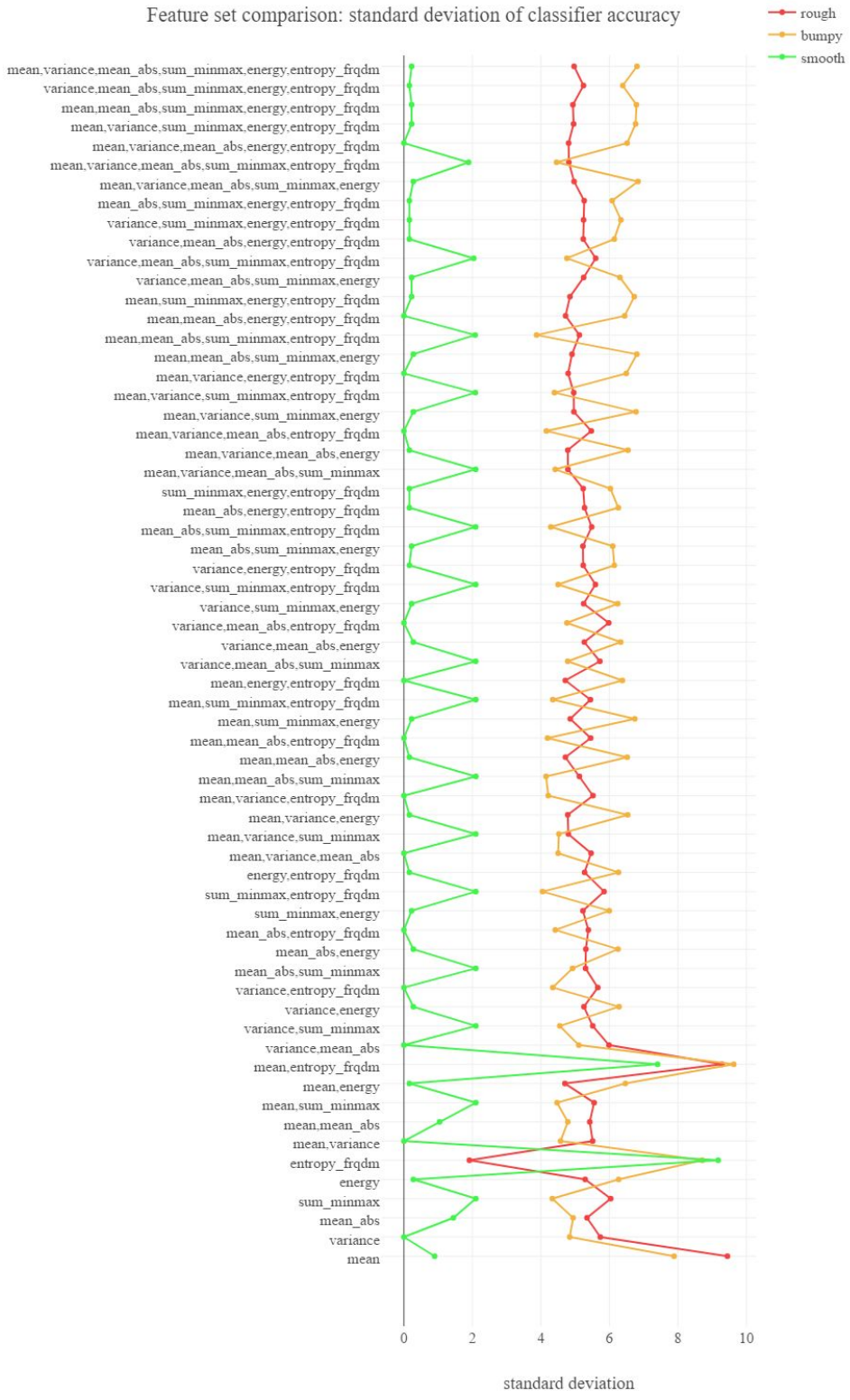


Figure 16. *A graph, describing the standard deviations of classifier accuracy over 500 evaluation cycles for every road surface type and feature set.*

4.4 Additional testing

Table 1. *Summed confusion matrix for feature set (mean, variance, mean_abs, sum_minmax, entropy_frqdm (frequency domain entropy)), describing the precision[18] and recall[19] metrics for each road type.*

	Rough (predicted)	Bumpy (predicted)	Smooth (predicted)	Total	Recall
Rough (actual)	**4623**	357	20	5000	92.46%
Bumpy (actual)	247	**4697**	56	5000	93.94%
Smooth (actual)	0	45	**5555**	5600	99.20%
Total	4870	5099	5631		
Precision	94.93%	92.12%	98.65%		

In subchapter 4.3, it was determined that classifier provides satisfactory performance with several feature sets. The feature set (*mean*, *variance*, *mean_abs*, *sum_minmax*, *entropy_frqdm* (*frequency domain entropy*)) was chosen to be analysed further.

A separate evaluation was performed for the selected feature set, with the aim of gaining additional information about the performance. The evaluation consisted of 200 randomized evaluation cycles, with 80% as training and 20% as testing data. This evaluation process can be considered similar to the one used to compile data for ***Figure 15*** and ***Figure 16***, with the only difference being the number of cycles performed. The results of this evaluation can be seen in ***Table 1***. The accuracy (recall) figures for *rough*, *bumpy* and *smooth* road types were 92.46%, 93.94% and 99.20%, respectively. For the purpose of this thesis, these accuracy values can be

[18] The precision metric indicates, how much of all the instance, where the classifier predicted a certain road type, were relevant.

[19] The recall metric indicates, what percentage of all the instances of a certain road type, were classified correctly.

considered satisfactory. Since this evaluation is also based on randomness, minor variations in accuracy can be expected, when doing multiple evaluations.

4.5 Conclusion

This chapter gave an overview of the data collection setup and also provided insight into the logic, that was used to choose roads for the collection process. Furthermore, figures and explanations were given for the evaluation process of the road surface quality detection algorithm. A confusion matrix was composed from the additional evaluation data of a chosen feature set.

The next chapter provides a conclusion to this thesis and also discusses future work.

Chapter 5: Conclusion and perspectives

5.1 Conclusion

As a result of this thesis, an Android application (Road Surface Data Collector) for linear acceleration data collection, was developed. The application makes use of the Android Sensor framework in order to gain access to magnetometer, gravity and linear accelerometer sensors. Based on the input from a magnetometer and a gravity sensor, a transformation process, that allows the phone to be in any orientation during data collection, was described and implemented. The application was used to collect acceleration data on different roads in and around the city of Tartu in 1 second long segments. Three types of road surfaces were distinguished: *smooth*, *bumpy* and *rough*. Each type being worse in quality than the previous.

As a secondary component, an algorithm for road surface quality detection was developed. It implements the idea of supervised learning in order to predict the quality of road segments based on features, calculated from the acceleration data. Six features, in total, were calculated: mean, variance, sum of the absolute values of maximum and minimum amplitudes, mean (absolute), energy (frequency domain) and entropy (frequency domain).

The detection algorithm was evaluated on a dataset that contained 393 data windows in total. Testing was done by randomizing the dataset and compiling train and test data with an 80%, 20% ratio respectively. The data was used to train and test a SVM classifier. A set of features, that provided good classifier accuracy in the initial feature set comparison, was chosen to be tested further. During the additional testing, the classifier demonstrated an accuracy of 92.46% in correctly classifying *rough* road segments with the chosen feature set. For *bumpy* and *smooth* road surface types, the accuracy values were 93.94% and 99.20%, respectively.

Compared to other works on this topic, Hoffmann et al. [10] attempted a similar approach of classifying road surface quality into three categories. However, with their data and implementation, they did not manage to get the accuracy of the detection to a satisfactory level, with the best overall accuracy for the 3-class classification being around 78%.

5.2 Future work

The main problem, that needs to be addressed in a future work, is speed. There needs to be an algorithm in place to filter out the unwanted effects of speed, before the solution, provided in this thesis, can be effectively used in a real world scenario.

Another aspect that needs to be explored further, is the possibility of using more than one type of car. Different cars have to tested and their acceleration signals compared, in order to gain more knowledge on the magnitude of their differences. If more tests are done, there is a possibility that a solution could be found.

If the aforementioned problems are solved, then this thesis' proof of concept solution could be developed into a cloud based application. In such application, data could be received from multiple persons and therefore, the quality of classification would be improved further. Instead of using a single data source to classify a road section, data from several persons can be taken into account, in order to filter out spontaneous events and possibly false data. Furthermore, the possibility of expanding the number of different road surface types, can also be explored.

Bibliography

1. Bajpai P. The Evolution of Smartphone Markets: Where Growth Is Going. 2016. http://www.nasdaq.com/article/the-evolution-of-smartphone-markets-where-growth-is-going-cm619105 (10.05.2017).
2. StatCounter press release. Mobile and tablet internet usage exceeds desktop for first time worldwide. 2016. http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide (10.05.2017).
3. Anguita D., Ghio A., Oneto L., Parra X., Reyes-Ortiz J.L. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. *International Workshop on Ambient Assisted Living*, 2012, pp. 216–223. Springer Berlin Heidelberg.
4. Statista statistics on worldwide vehicle usage. https://www.statista.com/statistics/281134/number-of-vehicles-in-use-worldwide/ (10.05.2017).
5. Adlinge S.S, Gupta A.K. Pavement deterioration and its causes. *International Journal of Innovative Research and Development*, 2013, vol. 2, no. 4, pp. 437–450.
6. OECD statistics page on transport infrastructure investment and maintenance spending. Used variable: ”Road infrastructure investment.” https://stats.oecd.org/Index.aspx?DataSetCode=ITF_INV-MTN_DATA (10.05.2017).
7. Tonde V.P., Jadhav A., Shinde S., Dhoka A., Bablade S. Road quality and ghats complexity analysis using android sensors. *International Journal of Advanced Research in Computer and Communication Engineering*, 2015, vol. 4, no. 3, pp. 101–104.
8. Seraj F., van der Zwaag B.J., Dilo A., Luarasi T., Havinga P. Roads: A road pavement monitoring system for anomaly detection using smart phones. *International Workshop on Modeling Social Media*, 2014, pp. 128–146. Springer International Publishing.
9. Alessandroni G., Klopfenstein L.C., Delpriori S., Dromedari M., Luchetti G., Paolini B.D., Seraghiti A., Lattanzi E., Freschi V., Carini A., Bogliolo A. SmartRoadSense: Collaborative Road Surface Condition Monitoring. *Proc. of UBICOMM-2014. IARIA.,* 2014, pp. 210–215.

10. Hoffmann M., Mock M., May M. Road-quality classification and bump detection with bicycle-mounted smartphones. *Proceedings of the 3rd International Conference on Ubiquitous Data Mining-Volume 1088*, 2013, vol. 1088, pp. 39–43. CEUR-WS.org.
11. GSMArena information page. http://www.gsmarena.com/nokia_5800_xpressmusic-2537.php (10.05.2017).
12. Astarita V., Caruso M.V., Danieli G., Festa D.C., Giofrè V.P., Iuele T., Vaiana R. A mobile application for road surface quality control: UNIquALroad. *Procedia-Social and Behavioral Sciences*, 2012, vol. 54, pp. 1135–1144.
13. Eriksson J., Girod L., Hull B., Newton R., Madden S., Balakrishnan H. The pothole patrol: using a mobile sensor network for road surface monitoring. *Proceedings of the 6th international conference on Mobile systems, applications, and services*, 2008, pp. 29–39. ACM.
14. Nason G.P., Silverman B.W. The stationary wavelet transform and some statistical applications. *Wavelets and statistics*, 1995, pp. 281–299. Springer New York.
15. List of Android code names, versions and API levels. https://source.android.com/source/build-numbers (10.05.2017).
16. Android Developer platform overview page. https://developer.android.com/about/dashboards/index.html (10.05.2017).
17. Android Developer overview page on sensors. https://developer.android.com/guide/topics/sensors/sensors_overview.html (10.05.2017).
18. Android Developer overview page on motion sensors. https://developer.android.com/guide/topics/sensors/sensors_motion.html (10.05.2017).
19. Android Developer page for *SensorEvent* class. https://developer.android.com/reference/android/hardware/SensorEvent.html (10.05.2017).
20. Slabaugh G.G. Computing Euler angles from a rotation matrix. *Technical Report*, 1999. http://www.staff.city.ac.uk/~sbbh653/publications/euler.pdf (11.05.2017).
21. Android Developer page for *SensorManager* class. https://developer.android.com/reference/android/hardware/SensorManager.html (10.05.2017).

22. Mohri M., Rostamizadeh A. and Talwalkar A. *Foundations of Machine Learning*. MIT press, 2012.
23. OpenCV API documentation for support vector machines. http://docs.opencv.org/2.4/modules/ml/doc/support_vector_machines.html (10.05.2017).
24. Preece S.J., Goulermas J.Y., Kenney L.P., Howard D. A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. *IEEE Transactions on Biomedical Engineering*, 2009, vol. 56, no. 3, pp. 871–879.
25. Bao L., Intille S.S. Activity recognition from user-annotated acceleration data. *International Conference on Pervasive Computing*, 2004, pp. 1–17. Springer Berlin Heidelberg.
26. Press W.H., Teukolsky S.A., Vetterling W.T., and Flannery, B.P. *Numerical Recipes: The Art of Scientific Computing*. Cambridge, UK: Cambridge Univ. Press, 2007.
27. NumPy documentation page for DFT. https://docs.scipy.org/doc/numpy/reference/routines.fft.html (10.05.2017).
28. Agrawal R., Faloutsos C., Swami A. Efficient similarity search in sequence databases. *Foundations of data organization and algorithms*, 1993, pp. 69–84. Springer, Berlin, Heidelberg.
29. Zhang A., Yang B., Huang L. Feature extraction of EEG signals using power spectral entropy. *BioMedical Engineering and Informatics, 2008. BMEI 2008. International Conference on*, 2008, vol. 2, pp. 435–439. IEEE.

Appendix

License

Non-exclusive licence to reproduce thesis and make thesis public

I, Madis Kariler,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

    1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

    1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Road Surface Quality Detection Using Accelerometer Data,

supervised by Amnir Hadachi, PhD

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 11.05.2017