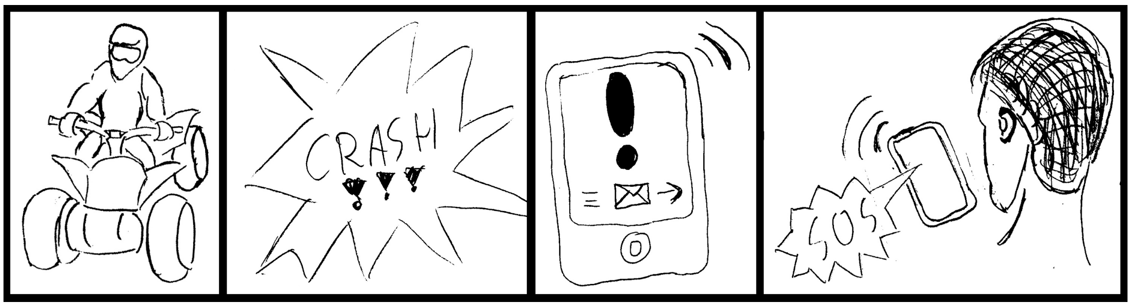
CHALMERS
UNIVERSITY OF TECHNOLOGY



Smartphone Based Automatic Incident Detection Algorithm and Crash Notification System for All-Terrain Vehicle Drivers

Using Smartphones to Automatically Notify Emergency Contacts of Accidents

Master's thesis in Systems, Control and Mechatronics

Gabriel Matuszczyk
Rasmus Åberg

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2016

Master's thesis 2016:73

Smartphone Based Automatic Incident Detection Algorithm and Crash Notification System for All-Terrain Vehicle Drivers

Using Smartphones to Automatically Notify Emergency Contacts of Accidents

Gabriel Matuszczyk
Rasmus Åberg

Department of Signals and Systems
Division of Automation and Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2016

Smartphone Based Automatic Incident Detection Algorithm and Crash Notification System for All-Terrain Vehicle Drivers
Using Smartphones to Automatically Notify Emergency Contacts of Accidents
Gabriel Matuszczyk
Rasmus Åberg

Supervisor: Leif Sandsjö, University of Borås/MedTech West
Examiner: Stefan Candefjord, Department of Signals and Systems/MedTech West

Master's Thesis 2016:73
Department of Signals and Systems
Division of Automation and Mechatronics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2016

Acknowledgements

As our work has progressed during this spring, certain people, without personal gain, have provided invaluable help. Kjell and Kristina Åberg, Tomasz and Josefa Matuszczyk, and Andréa Bergqvist helped us log data we've used throughout this project. They have given us a dataset with more variation.

We would like to thank the volunteer ATV drivers who, out of interest in our field, provided a previous project with a large dataset used also in our research.

Our examiner Stefan Candefjord and supervisor Leif Sandsjö have given us guidance and feedback from day one until the very end, inspiring us to find new solutions to difficult problems, while allowing us to work autonomously.

Gabriel Matuszczyk and Rasmus Åberg, Gothenburg, June 2016

Abstract

All-Terrain Vehicle (*ATV*) drivers face a different sort of danger than posed by most other means of travel. An ATV is mainly designed to travel forests and unpaved areas. It is a versatile vehicle often used only by one person at a time; thus, if an accident occurs far out in the wilderness help is hard to come by, especially if the driver is incapacitated. As a result of the prevalence of GPS-enabled smartphones, an application for them implementing an accurate Incident Detection Algorithm (*IDA*) could save even an unconscious driver, via an automatic message to an In Case of Emergency (*ICE*) contact. This thesis investigates the possibility of designing such an application, as well as the feasibility of running it in real time on a smartphone.

A dataset containing 55 hours of logged normal ATV driving motion data was available, of which approximately 21 hours were of high quality. Two logs were omitted due to their content of abnormally erratic motion data. Machine learning methods (specifically One Class Support Vector Machines (*OC-SVM*)) were used in order to create an IDA that can satisfactorily identify several types of accidents. Motion data was collected containing 20 abnormalities in the form of a test person falling and rolling in several directions, in order to simulate a number of crash scenarios. Together with Accident Confirmation Criteria (*ACC*) to cancel false positives, and a decision tree within the IDA, few false alarms are raised while alarms do occur for all incidents simulated in the crash dataset. Overall, the trained OC-SVM classified normal driving with a precision of 99.39%, and correctly identified all 20 simulated accidents; thus, the OC-SVM obtained an estimated F_1-score of 99.69%.

A design for a smartphone application to enable this automatic alarm is proposed in the form of a flow chart. Investigations of required functionality support the claim that a smartphone is capable of running the IDA in real time and with low battery consumption.

With the limited amount of normal driving data, and only simulated crash data, further investigations must be performed in order to ensure no overfitting has taken place. The next step in the development would be for a test group consisting of regular ATV drivers to evaluate the performance of the IDA in real life situations. It is the authors' opinion that with additional trials and tweaking of parameters, a well-functioning smartphone application could be released to the public and potentially serve as a life saver, perhaps even for other vehicles, in cases where the driver is otherwise helpless.

Keywords: Machine learning, Support vector machines (SVM), Incident detection algorithm (IDA), All-terrain Vehicle (ATV), Smartphone, eSafety.

Contents

Glossary

ACC - Accident Confirmation Criteria to prevent false alarms, several confirmation criteria has to be fulfilled in order for an accident to be taken seriously. These criteria are defined as ACC throughout this report.

Algorithm evaluations of an input are carried out, and depending on the input's content a certain output is reached. An algorithm follows predefined rules and steps to determine the output and can often run autonomously on a computer.

Android the operating system developed by Google for smartphones, implemented by several different smartphone manufacturers.

ATV - All-Terrain Vehicle also known as *quadricycle*, *four-wheeler* or *quad bike*. A vehicle with a high center of gravity and high ground clearance, the latter making it easier to drive in rough terrain.

Feature extraction the process of selecting parts of data to use for training a classifier, in order to decrease the number of variables (features) that the classifier must consider. For example, if several parts of the data are correlated, it could be possible to merge them into one feature in order to increase classifier performance while retaining classification accuracy.

GPS - Global Positioning System with a receiver (built-in in most smartphones) signals from GPS satellites can be used to obtain coordinates for longitude and latitude.

ICE - In Case of Emergency common term to identify who to contact for example in a person's contact list if that person has been part of an accident or similar.

IDA - Incident Detection Algorithm used to describe the whole process of the decision making process. Starts with analysing data, make a decision regarding if an anomaly (incident) has occurred, if so, trigger an alarm and cancel alarm if new data indicates it was a false alarm. Worth noting is that the IDA doesn't notify an ICE, it simply determines whether an incident has occurred, and whether the incident should trigger an alarm or be classified as a false alarm.

IDE - Integrated Development Environment a computer program used to program other software programs, often include libraries, debugger, compiler etc.

Incident when the IDA is running a lot of data is collected from different sensors, for normal driving all sensor values should be within a *normal driving space*. If a sensor's value exits this space an *anomaly* occurs, these are the *incidents* that require further investigation by the IDA.

iOS the operating system used by, and developed for, Apple's smartphone model; iPhone.

OC-SVM - One Class Support Vector Machine like SVM, it is a machine learning method. However it uses only one class to define the limitation in space.

RC - Rejection Criteria similar to ACC, but instead of confirming an accident, they confirm false alarms. If a rejection criterion is fulfilled within a certain time limit, then the incident is discarded as a false alarm.

SDK - Software Development Kit a set of tools to develop software programs for specific hardware and/or operating systems, often available in the IDE and chosen before a new project is started.

SVM - Support Vector Machine a machine learning method, which uses samples of each class to create a decision boundary between the classes, which maximises the distances between outermost samples from each class and the boundary. often in a higher dimension than that of the samples.

User smartphone interaction refers to the common case where the smartphone is interacted with in such a way that the motion sensors detect erratic movements. Such an event could be the driver placing it in a pocket or altering its position in a pocket. These scenarios create sensor readings which can be quite similar to incident sensor readings.

VRU - Vulnerable Road User a group of people using roads by means of travel that offer little to no protection, most commonly thought of is pedestrians and cyclists.

1 Introduction

Vulnerable Road Users (*VRUs*) include almost everyone not traveling by car, truck or bus on common roads, most commonly thought of is people travelling on foot or biking on or near the road. Two other vulnerable road users are motorcyclists and All-terrain vehicle (*ATV*) drivers. Compared to the protective capacity of modern cars, all subgroups of VRUs are at a significant disadvantage. Lately, the focus of traffic safety has shifted from passively protecting car users in a crash, to actively avoid crashes altogether, and also to develop new safety equipment for VRUs. One such piece of safety equipment is *Hövding*, a modern helmet for cyclist that inflate when an accident is imminent. It is essentially an airbag for cyclists which envelops the head and generally provides better protection compared to conventional helmets [1].

ATV drivers face another problem than just being vulnerable compared to car drivers; their vehicle is by design very versatile and is used in unpaved areas for both work and pleasure. If an accident occurs in a seldom travelled area (such as a forest) the driver has to manage the situation by him or her self, since the likelihood of a passerby appearing generally is small. Even on the assumption that the driver is able to contact emergency services on his or her own, relating the location can be problematic since forests rarely possess any road signs or discernible landmarks.

Machine learning algorithms, with their relatively high computational performance requirements, have greatly increased in popularity, fuelled by increases in computer performances and decreases in hardware prices. New areas where these algorithms are imagined to work well are constantly explored. This report documents precisely such an exploration: a machine learning algorithm is trained using smartphone sensor data, to detect anomalies in order to detect incidents while driving ATVs. The algorithm, running within a smartphone application, should trigger a message (preferably containing GPS coordinates) to an emergency contact if an anomaly (i.e. an incident) is detected.

1.1 Background

The Specialty Vehicle Institute of America (*SVIA*) defines an ATV as a motorised off-highway vehicle designed to travel on four low-pressure tires, having a seat designed to be straddled by the operator and handlebars for steering control; furthermore, the SVIA defines two subgroups, Types I and II, where Type I ATV's are designed to carry only the driver and Type II ATV's can carry both the driver and one passenger, situated behind the driver [2].

In Table 1 accident data on Swedish roads for 2015 is presented [3]. ATVs may be registered under different categories depending on their use and performance, or not registered at all if they are used in an enclosed area. Of the three possible categories in Table 1 (*Motorcycle*, *Moped rider* and *Other*) that ATVs fall under, two of them, by percent, has the highest death and severe accident rate. For all road user groups the goal is of course to avoid accidents altogether and keep the severity at a minimum if unavoidable, meaning improvement is needed.

Table 1: *Statistics of accident severity for different groups of road users in Sweden during 2015 [3].*

Severity	**Car**	**Motor-cycle**	**Moped rider**	**Cyclist**	**Pedestrian**	**Other**	**Total**
Light	12850	663	756	1604	1153	172	17198
Severe	1534	248	110	241	271	41	2445
Deadly	159	44	5	17	28	6	259
Total	14543	955	871	1862	1452	219	19902
Percent of accident per severity level and road user group [%]							
Light	88.36	69.42	86.80	86.14	79.41	78.54	86.41
Severe	10.55	25.97	12.63	12.94	18.66	18.72	12.29
Deadly	1.09	4.61	0.57	0.91	1.93	2.74	1.30

From a report published by *the Swedish Transport Administration* [4] some interesting data can be found. During the years 2000-2003 around 3000 new ATVs were registered per year, for the years 2011-2012 this number had risen to 11000 and at the beginning of 2013 just over 91000 ATVs were registered (this figure does not include unregistered ATVs or ATVs registered as tractors, so total ATVs in use is likely higher). In the same report accident data presented estimates that 7000 persons visited emergency rooms between 2007 and 2010 for ATV related injuries. Other data for the period 2001-2012 is presented in Table 2. For the *on road*-row in the table is it worth mentioning that 90 % of fatal accidents were single vehicle accidents and that for 20 % of the cases where the ATV overturned, the deceased driver was still beneath the ATV when found.

Table 2: *Deadly ATV accidents during 2001-2012 and how many involved an overturned ATV [4].*

Location	**Killed**	**Overturned ATV [%]**
On road	42	70
Off road	27	60

Due to the high rate of single-vehicle accidents combined with drivers often traveling alone, an automatic safety system could help lower the death toll. What this automatic system should be able to do is to detect incidents using an algorithm without any direct input from the user. Commonly this kind of system is known as an *Incident detection algorithm*, or *IDA*; IDA's use collected data (from for example accelerometer sensors) to automatically evaluate if an incident has occurred. Smartphones have become such common gadgets, and their increasing quality regarding sensor readings and computational power could provide a very attractive platform to run such an IDA on, both for the developers and for users.

Smartphones have become everyday objects, between 76 and 89 percent of Swedes aged 16-54 have utilised a smartphone outside of their homes [5]. Since smartphones are so widespread and apps are easily distributed through official channels, a successful application may be an important contribution towards providing medical assistance to victims sooner.

Similar IDA's have been produced before, which found that the quality of the built in sensors in common smartphones is good enough to detect accidents while riding a bike as well as for horse riders. For the development of the bike IDA [6], simulations were made with a smartphone on a crash dummy and a bike to collect data, the dummy was mounted and the bike pushed in order to gain speed and crashed into objects to simulate an accident. For the horse riders [7], similar simulations were obtained by means of researchers allowing themselves to be thrown off a mechanical bull. Data of this type can be impractical or expensive to collect in many other incident detection applications, such as ATV accidents.

The United States Consumer Product Safety Commission (*CPSC*) found after analysing 71 400 ATV-related injuries, that in 68.5% of the cases the driver was the only rider [8]. Thus, if severe accidents happen, the driver is alone and help won't arrive until someone misses the driver or he or she is found by a passerby. If the driver is able to contact emergency services, conveying an exact location can be difficult for several reasons. The driver may not know where he or she is, cellular services may be poor, which would make triangulation hard, and can also cause a phone call to cut off. If the driver has a GPS signal, problems may arise since there are several similar standards to convey coordinates. A short text based message including a specific standard of GPS-coordinates (which generally has much better coverage [9] than cellular services [10]), could be delivered in its entirety to an in case of emergency (*ICE*) contact and provide useful information for a search and rescue team. In short, there is a need for an IDA which can autonomously detect an incident, and quickly relay all necessary information to another person; the problem, and potential application of an IDA to solve it, is illustrated in Figure 1.

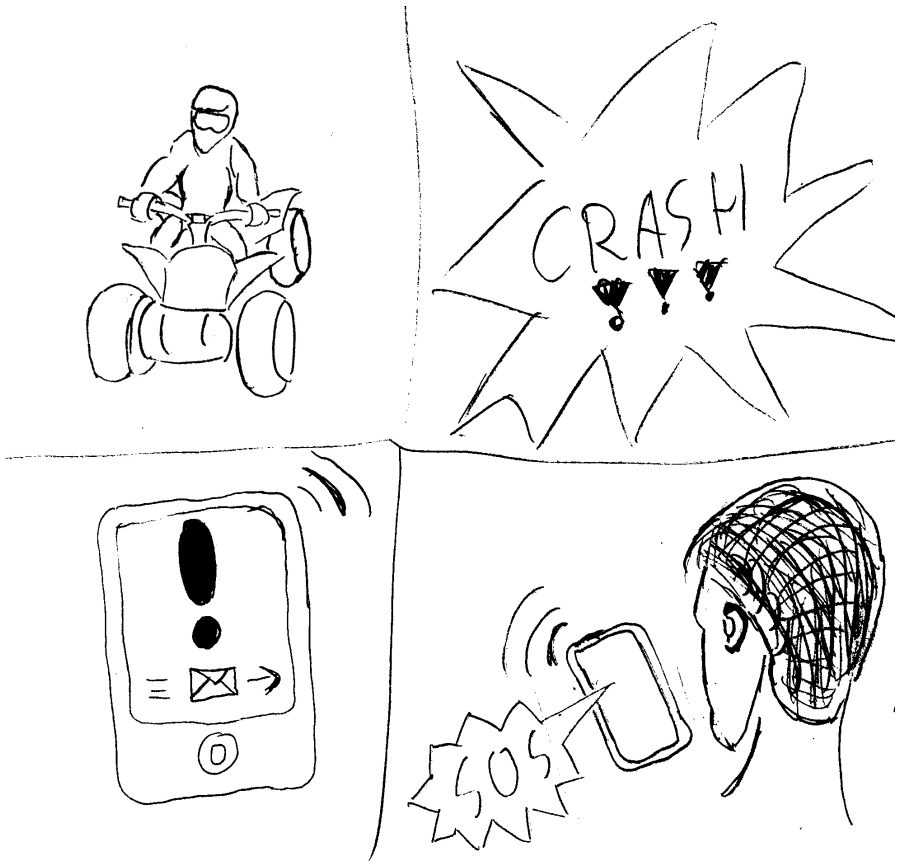


Figure 1: *A dangerous situation, for which the risk could be reduced by use of an autonomous IDA to facilitate expeditious search and rescue. The series of images depict an ATV driver during a normal drive, who is suddenly subjected to a crash; however, the driver's smartphone is running the proposed IDA which detects the crash and notifies an ICE contact.*

1.2 Purpose

The primary long-term purpose of the project is to save lives, and to minimise the effects of injuries in ATV-related accidents, by offering a means of obtaining fast emergency medical attention in cases where the driver cannot him- or herself summon such aid. Another purpose is to evaluate what is actually possible to identify using the sensors in today's smartphones, how well the information captured represents the real world and if it is possible to filter out disturbances well enough that the obtained information is useful. General research on sensor accuracy and fields of use could uncover opportunities not previously thought of.

1.3 Aim

The primary aims are:

1. to create an IDA which can identify accidents and crashes.
2. Optimise the IDA in order to obtain a high F_1-score and
3. release it as an application through official distribution channels, such as the Google and Apple online application stores.

No other hardware than the smartphone's stock sensors should be used as input for the IDA. The application should create some sort of distress signal, such as a text message, in case an incident or crash has occurred.

1.4 Scope and Limitations

Some uncertainty surrounds the collected volunteer data (see Section 3.2). In some cases volunteers seem to have forgotten about the running logging after dismounting the ATV, and inadvertently logged other data, such as walking. Evaluation will be conducted and as much as possible of incorrectly logged data will be removed. Although there is no way of knowing with full certainty that the remaining data is only ATV driving, it will be assumed that it is.

No real life evaluation will be conducted since it would either require the expensive employment of professional stunt men, or be unreasonably dangerous for the test person. Simulation of crashes and general unusual data (such as just dropping the phone from a low height) will however be collected and evaluated with the IDA to see how it classifies these scenarios.

1.5 Feasibility Studies of Machine Learning Methods and Smartphone Performance

Machine learning methods have already been used extensively in smartphone applications. Specifically of interest to this project, a report from 2012 documents the use of *multi-class support vector machines* for smartphone-based human activity recognition using accelerometer data [11], and another report from 2014 describes the use of machine learning to filter out smartphone magnetometer disturbances in *Pedestrian Dead Reconning* for indoor localisation [12]. Furthermore, two more reports documented the use of machine learning in smartphones: the first report, from 2012, for detection of *Freeze of Gait* prevalence in the everyday lives of patients suffering from advanced Parkinson's Disease with more than 95 % accuracy and specificity [13]. The second report, from 2009, documents the implementation of machine learning in a smartphone application, which detects whether an individual falls, as well as said individuals response to the fall, and subsequently sends an SMS to one or more pre-specified social contacts [14]. Many other smartphone-based applications using machine learning have been reported as well, even fish species recognition using computer vision and multi-class support vector machines in smartphones as a step towards allowing Chinese fish farmers to diagnose fish disease using their smartphones [15]. The work of David M. J. Tax, 2001, shows that it is possible to use SVM's to differentiate between two classes even though only one of the classes can be sampled [16]; something that had also already been confirmed by Schölkopf et al in 1999 [17].

2 Theory

To understand some of the decisions made later on in the report, some theoretical background is necessary. Also included in this section is explanations to some terminology and differences for methods.

2.1 Machine Learning Methods for Classification

The general idea of Machine Learning is to use a computer's capability to repeatedly and quickly calculate (complex) equations, and with the guidance of an optimisation equation find the best parameters and/or function for the problem evaluated. Machine learning's true strength lies within computers ability to do the same calculation with slightly altered values back-to-back without any fault. They can therefore be used in applications where a lot of data exists and an abstract solution is prominent.

Machine learning can be used for other purposes than classification [18] but in this thesis it is a case of *accident* or *no accident*. Several different methods exists for classifications, each with its different strengths and weaknesses depending on preconditions and goals set by the developer/researcher [19]. In the present study, a lot of data exists of only one class (no *accident* data exists, see Section 3.2.5 for simulated crash data), the data consist of twelve sensor values in each sampling instance and no need for fast training exist. With these preconditions *Support Vector Machine (*SVM*) for anomaly detection* is a fitting method [20]. Furthermore, in a 2003 study examining the possibility to identify traffic incidents in an arterial network found that SVM classifiers offered a lower misclassification rate, higher correct detection rate, lower false alarm rate and slightly faster detection time than Multi-Layer Feed forward neural network and probabilistic neural network models [21].

2.2 Evaluation of Algorithm Performance

When evaluating the performance of a binary (i.e two-class) classification algorithm such as an SVM classifier, it is necessary to consider both accuracy in detecting datapoints belonging to a target class, as well as accuracy in detecting datapoints belonging to the outlier class. This is especially important when the numbers of available datapoints from each respective class differ greatly. For example, if 90 datapoints are available from the target class, and 10 datapoints are available from the outlier class, and a classifier classifies all datapoints as belonging to the target class, the resulting accuracy would be 90% even though all outlier datapoints were misclassified. One well-known performance measure for binary classification is the F_1-score.

The F_1-score is based on two parameters; namely, *precision* and *recall*. The precision of a classifier is given by the number of correct positive classifications, i.e target class datapoints classified correctly, divided by the total number of positive classifications, i.e all datapoints classified as belonging to the target class. The recall of a classifier is obtained identically but concerns negative classifications, i.e outlier class datapoints classified correctly out of the total number of datapoints classified as belonging to the outlier class. The F_1-score is a weighted average of the precision and recall, as defined in Equation (1), which reaches its best value at 1.

$$F_1 = 2 \cdot \frac{(precision) \cdot (recall)}{(precision) + (recall)} \tag{1}$$

In the example described in the previous paragraph, the classifier precision would be 100 %; however, since the classifier misclassified all outlier datapoints, the recall would be 0 % and the F_1-score, calculated by Equation (1), would also be 0 %.

2.3 Large-Margin Binary Classification Using Support Vector Machines

In order to understand SVM's, consider the data set created by Ronald Fisher with the objective of solving the taxonomic problem of distinguishing iris flower species *Iris Setosa* from *Iris Versicolor* and *Iris Virginica* by observing four parameters: sepal length, sepal width, petal length and petal width [22]. While Fisher used traditional statistical methods and manual calculations as an attempt to find coefficients which could be used to classify the species, using SVM's it is possible to automatically find a classifier which accurately identifies the species of iris flower.

2.3.1 Understanding the Power of Support Vector Machines

Considering only sepal length and sepal width as parameters and plotting each observed iris flower as a data point based on the two parameters, an SVM will find the straight line which separates the data points of two classes with the largest possible margin of separation. It can be readily seen in Figure 2 that Iris Setosa data points are indeed linearly separable from Iris Versicolor in Fisher's dataset, with great classification accuracy; however, the same cannot be said for the randomly generated data in Figure 3a. An adjustment is necessary to distinguish between classes that are not linearly separable. This is handled by SVM's by mapping non-linearly separable data into a higher dimension where it is possible to obtain, instead of the optimal separating line, the optimal separating hyperplane [23][24]. From the higher dimension this hyperplane is mapped back to the original dimension to obtain a non-linear separator *even though* only linear methods have been used. This method is called the *Kernel trick*, where *Kernel* refers to the function used to map the data points to the higher dimension (worth noting is that during training of the SVM a non-linear kernel is used, but when the training is finished only linear methods are used to classify data). The non-linear separator shown in Figure 3b is the result of classification using the Kernel trick.

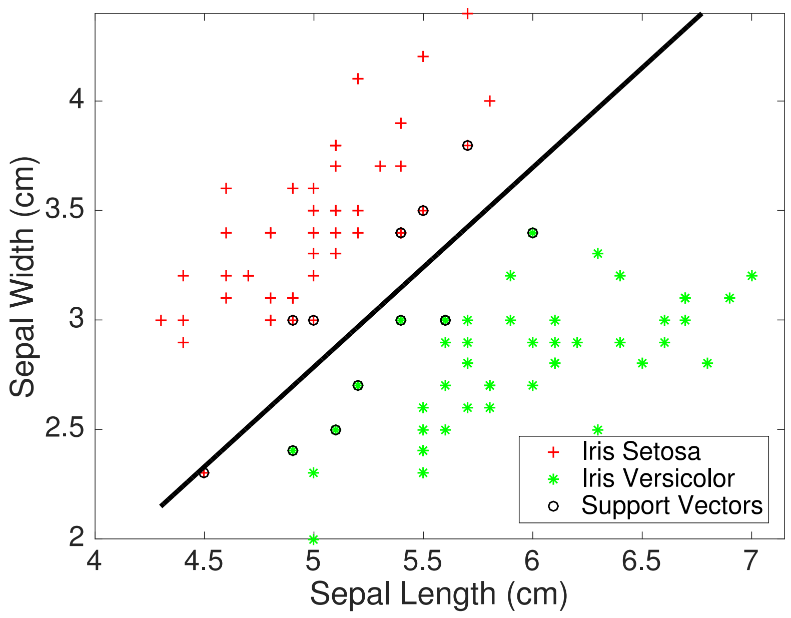
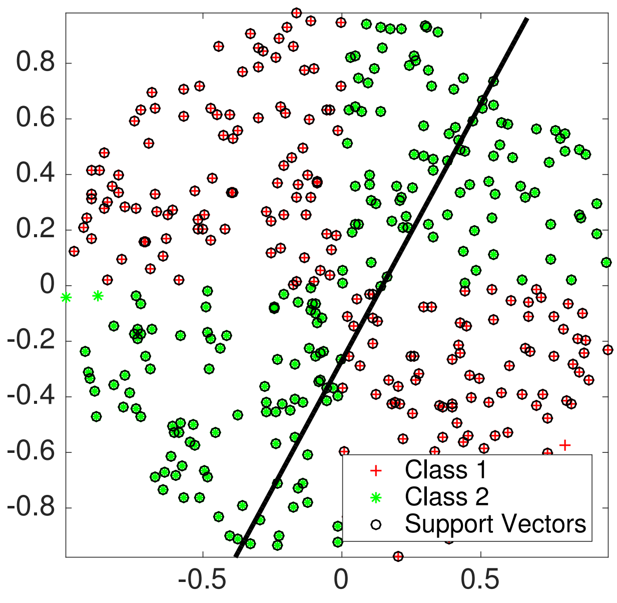
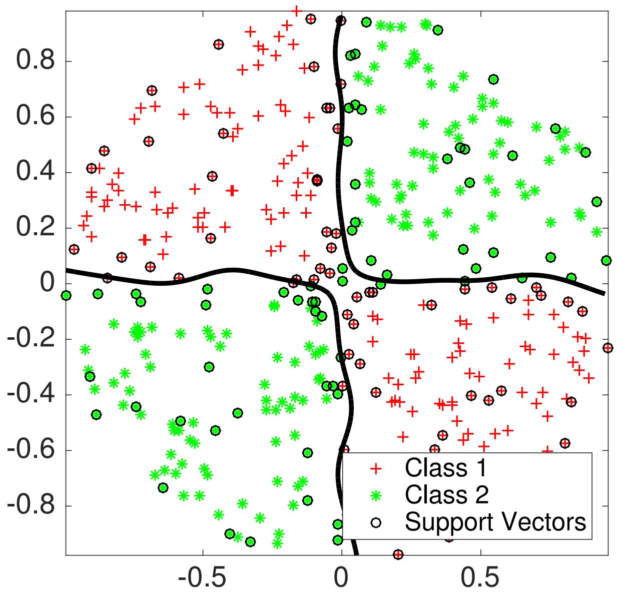


Figure 2: *Classification of Iris Setosa versus Iris Versicolor using an SVM classifier. Generated using Fisher's Iris data set.*



(a) *SVM using Linear Kernel.*

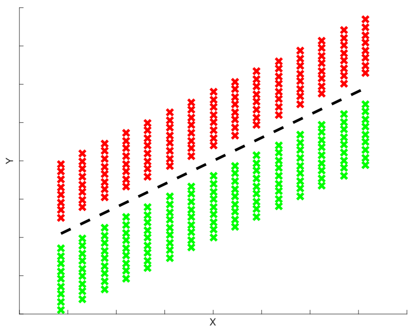


(b) *SVM using Radial Basis Function Kernel.*

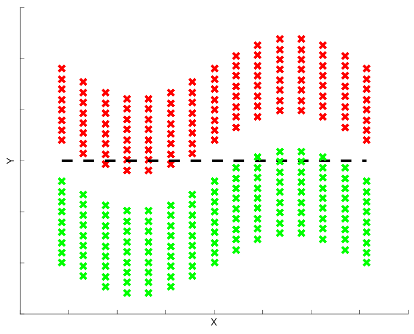
Figure 3: *Visualisation of the results of using the "Kernel trick", in order to separate linearly unseparable data using an SVM.*

2.3.2 Understanding The Kernel Trick

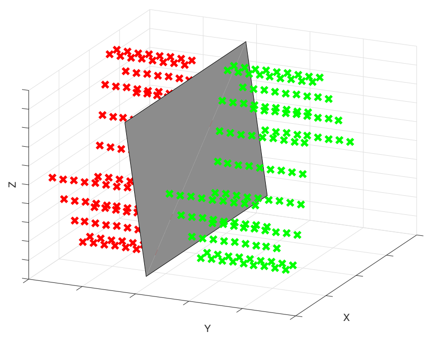
As mentioned in Section 2.3.1, SVM's provide non-linear decision boundaries by mapping non-linearly separable datapoints into a higher dimension where they are linearly separable. A step-by-step visual description of how the Kernel Trick works is displayed in Figure 4. In Figure 4a two datasets containing two-dimensional datapoints are represented in a Cartesian plane. Clearly, the two datasets can easily be separated by a straight line, as illustrated by the dotted line drawn between them. Figure 4b shows a case where the datapoints contained in each dataset are distributed in such a way that it is not possible to separate them with a straight line. The dotted line drawn in the figure would give rise to misclassifications if used as a decision boundary. This is where the Kernel Trick comes in handy: in Figure 4c, the datapoints of both datasets are mapped into a three-dimensional space using the same non-linear function, termed the "Kernel", and it turns out that it is possible to find a plane in 3D which optimally separates the datasets. Using the same Kernel function to map the optimal separating plane back to the original dimension provides the non-linear separator drawn as a dotted line in Figure 4d. Use of the Kernel Trick is not limited to low-dimensional classification problems such as the ones described in this section – it handles much higher dimensions using the same approach.



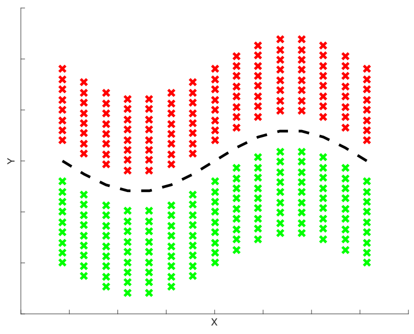
(a) *Linearly separable 2D datasets.*



(b) *Non-linearly separable datasets.*



(c) *Mapping of datapoints into higher dimension (3D).*



(d) *Non-linear decision boundary after mapping back to 2D.*

Figure 4: *Step-by-step description of how SVM's produce non-linear decision boundaries.*

2.4 One-Class Support Vector Machines for Data Description and Anomaly Detection

Binary classification using SVM's usually involves supplying training data from two or more different classes. In Section 2.3, the classes were different species of iris flowers, as well as some randomly generated datapoints, and training data was available from each class; however, one might consider the case where the goal is to obtain an SVM classifier algorithm which can differentiate between two classes, but where there is little or no data describing one of them. An example of this case, as described by Tax and Duin in 2004 [24], is a machine monitoring system in which the current condition of a machine is examined and an alarm is raised when there is an anomaly. Measurements of normal working conditions are usually cheap and easy to obtain; however, measurements of abnormal operation would require the destruction of the machine in all ways that are desirable to detect. The solution proposed by Tax and Duin is to use data from the well-sampled class and define it as the *target class* and thus define any other data-point, which does not classify as part of the target class, as an outlier, or anomaly [24]. This is called *Support Vector Data Description* (*SVDD*) or *One-Class Support Vector Machines* (*OC-SVM*).

2.4.1 Using Samples From One Class to Distinguish Between Two

Reviewing the Fisher Iris problem of Section 2.3, consider the case in which Fisher had only measured a large number of Iris Setosa flowers, but did not have access to any measurements from Iris Versicolor. Using an OC-SVM classifier, it is possible to obtain a defined area in the datapoint space which is isolated and defined as the target class. Any samples which occur outside of this space will be considered outlier data; in this case, target data will be Iris Setosa and any datapoint outside the defined separator will be classified as Iris Versicolor. This is illustrated in Figure 5. For illustrative purposes, the same method has been used to describe one of the classes of randomly generated data of Figure 3; the results of training a one-class SVM on only the positive examples (represented as "+") are displayed in Figure 6.

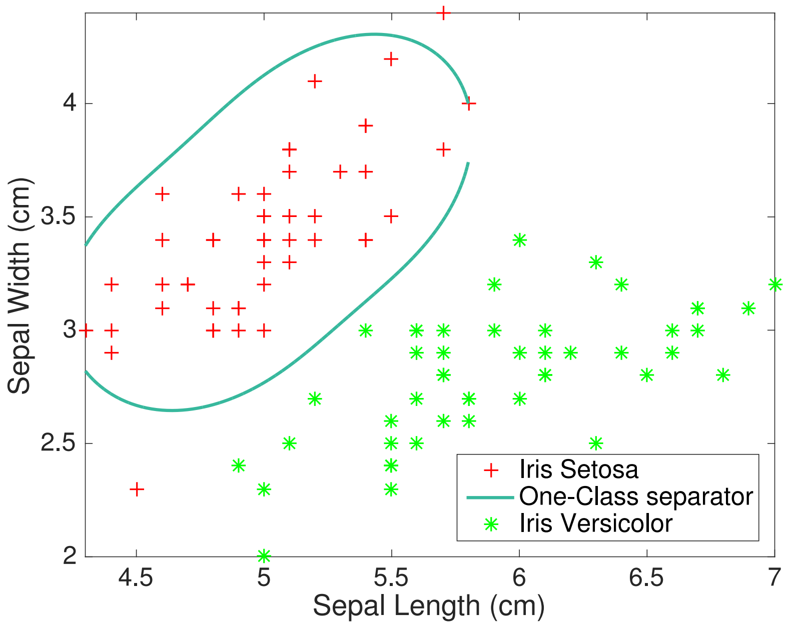


Figure 5: *Using OC-SVM to distinguish between Iris Setosa and Iris Versicolor using only Iris Setosa samples for training. Generated using Fisher's Iris data set.*

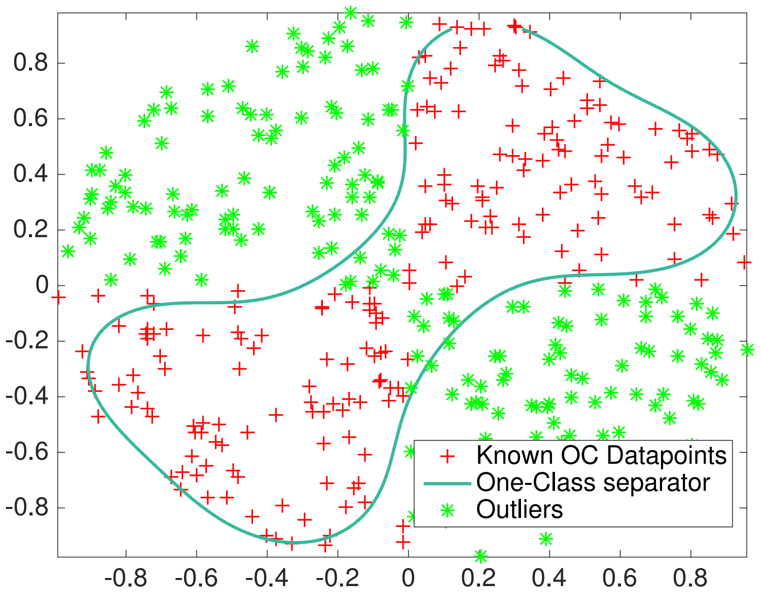
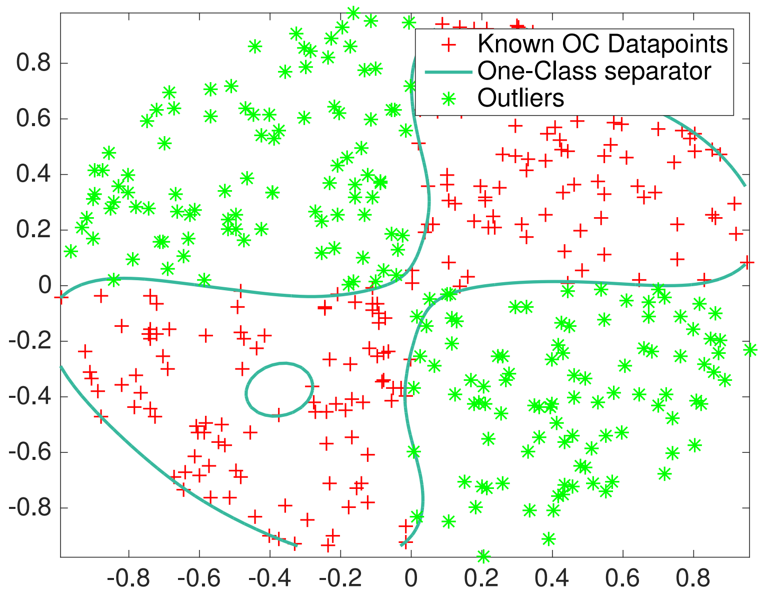


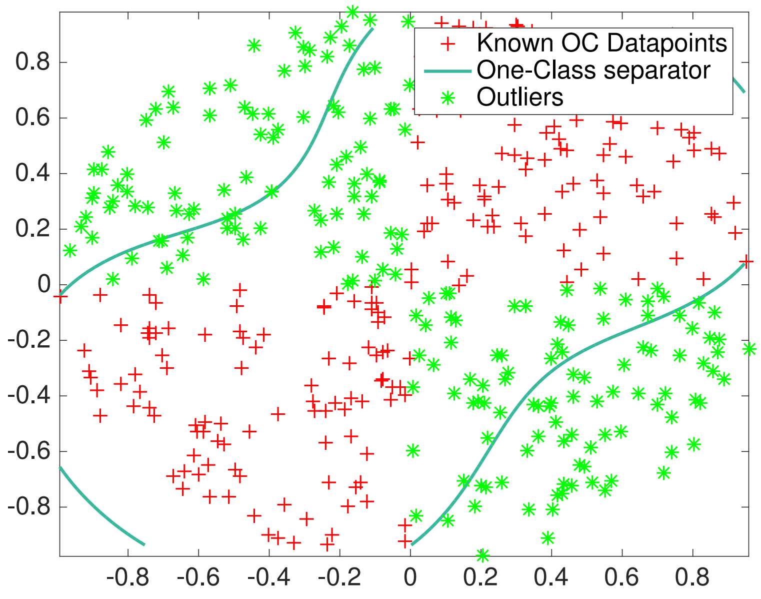
Figure 6: *Using OC-SVM to distinguish between the two randomly generated, linearly unseparable classes.*

2.4.2 Tuning the OC-SVM Decision Boundary Using Regularisation

As shown in Figure 6, the decision boundary does not enclose all known datapoints perfectly. This is due to the OC-SVM having been *regularised* during training. Regularisation is a way of ensuring that the resulting classifier does not *only* recognise all datapoints it has previously seen, but also generalises well to new datapoints from the same class. In OC-SVM training, this is performed by adding a regularisation term, denoted by the Greek letter ν (*nu*), which adjusts the penalty for misclassifications during training. The value of ν can be varied between zero: *non-inclusive*, and one: *inclusive*. By setting a small ν, misclassifications are heavily penalised and the OC-SVM will try to find a decision boundary which includes as many as possible of the training datapoints. If there should be unlikely or highly unusual datapoints in the set, it could be better if the OC-SVM ignored these; increasing ν allows disregarding of some misclassifications and produces a simpler classifier. Figure 7 shows the results of two OC-SVM classifiers. Figure 7a shows a highly overfitted decision boundary as a result of a small ν. On the other hand, Figure 7b shows an underfitted decision boundary, resulting from a ν set too high.



(a) ν *close to zero.*

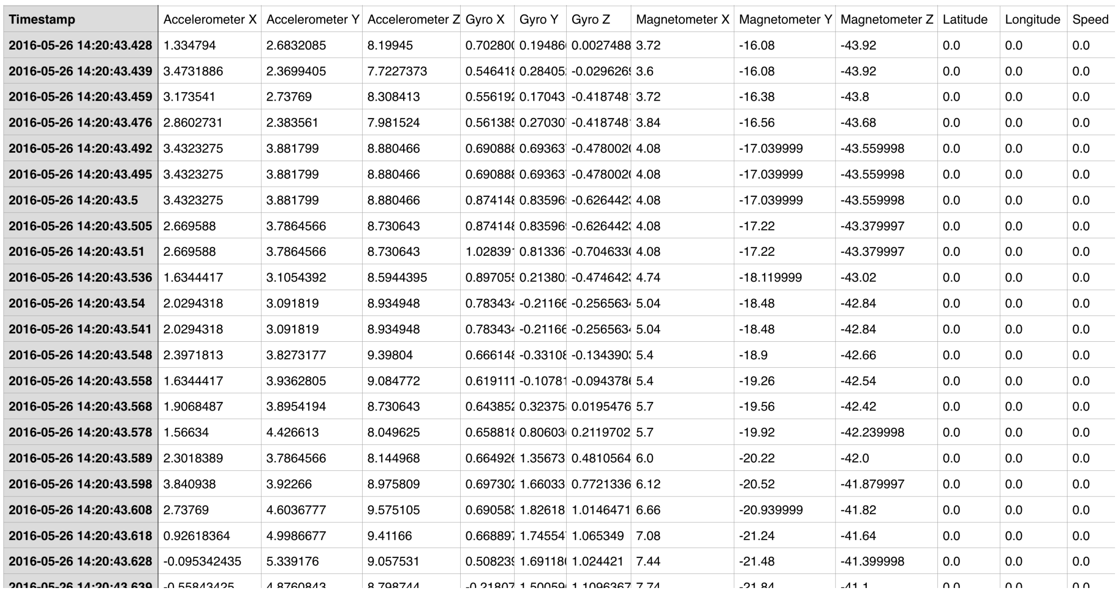


(b) ν *close to one.*

Figure 7: *OC-SVM classification results with different levels of regularisation.*

2.5 Using the Smartphone Application LogYard for Data Collection

A smartphone application called LogYard [25] was used to perform all data logging both during the writing of this thesis, and during a prior project [26] which resulted in data used for this thesis. LogYard works by constantly sampling data from a smartphone's 3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer and GPS location and estimated speed, and saving each sample to the smartphone's system storage. The sampling rate is set to 100 samples per second. Data is stored in the form of Comma-Separated Value (*CSV*) files and each log is denoted by an anonymous numerical user ID. CSV-files can be imported and parsed into spreadsheet form by both MATLAB and most spreadsheet applications; a typical output of a LogYard logging session is illustrated in Figure 8, where the CSV-file in question was parsed by the application *Numbers* [27]. Each file also contains metadata detailing, e.g, device model, sensor ranges and which unit system was used for the measurements. Data logging is started and stopped manually by the user and, thus, the logs invariably contain motion data produced by smartphone interaction in the beginning and end sections.



Timestamp	Accelerometer X	Accelerometer Y	Accelerometer Z	Gyro X	Gyro Y	Gyro Z	Magnetometer X	Magnetometer Y	Magnetometer Z	Latitude	Longitude	Speed
2016-05-26 14:20:43.428	1.334794	2.6832085	8.19945	0.70280	0.19486	0.0027488	3.72	-16.08	-43.92	0.0	0.0	0.0
2016-05-26 14:20:43.439	3.4731886	2.3699405	7.7227373	0.546418	0.28405	-0.029626	3.6	-16.08	-43.92	0.0	0.0	0.0
2016-05-26 14:20:43.459	3.173541	2.73769	8.308413	0.55619	0.17043	-0.418748	3.72	-16.38	-43.8	0.0	0.0	0.0
2016-05-26 14:20:43.476	2.8602731	2.383561	7.981524	0.56138	0.27030	-0.418748	3.84	-16.56	-43.68	0.0	0.0	0.0
2016-05-26 14:20:43.492	3.4323275	3.881799	8.880466	0.690888	0.69363	-0.478002	4.08	-17.039999	-43.559998	0.0	0.0	0.0
2016-05-26 14:20:43.495	3.4323275	3.881799	8.880466	0.690888	0.69363	-0.478002	4.08	-17.039999	-43.559998	0.0	0.0	0.0
2016-05-26 14:20:43.5	3.4323275	3.881799	8.880466	0.874148	0.83596	-0.626442	4.08	-17.039999	-43.559998	0.0	0.0	0.0
2016-05-26 14:20:43.505	2.669588	3.7864566	8.730643	0.874148	0.83596	-0.626442	4.08	-17.22	-43.379997	0.0	0.0	0.0
2016-05-26 14:20:43.51	2.669588	3.7864566	8.730643	1.02839	0.81336	-0.704633	4.08	-17.22	-43.379997	0.0	0.0	0.0
2016-05-26 14:20:43.536	1.6344417	3.1054392	8.5944395	0.89705	0.21380	-0.474642	4.74	-18.119999	-43.02	0.0	0.0	0.0
2016-05-26 14:20:43.54	2.0294318	3.091819	8.934948	0.78343	-0.21166	-0.256563	5.04	-18.48	-42.84	0.0	0.0	0.0
2016-05-26 14:20:43.541	2.0294318	3.091819	8.934948	0.78343	-0.21166	-0.256563	5.04	-18.48	-42.84	0.0	0.0	0.0
2016-05-26 14:20:43.548	2.3971813	3.8273177	9.39804	0.666148	-0.33108	-0.134390	5.4	-18.9	-42.66	0.0	0.0	0.0
2016-05-26 14:20:43.558	1.6344417	3.9362805	9.084772	0.619111	-0.10781	-0.094378	5.4	-19.26	-42.54	0.0	0.0	0.0
2016-05-26 14:20:43.568	1.9068487	3.8954194	8.730643	0.64385	0.32375	0.0195476	5.7	-19.56	-42.42	0.0	0.0	0.0
2016-05-26 14:20:43.578	1.56634	4.426613	8.049625	0.658818	0.80603	0.2119702	5.7	-19.92	-42.239998	0.0	0.0	0.0
2016-05-26 14:20:43.589	2.3018389	3.7864566	8.144968	0.664926	1.35673	0.4810564	6.0	-20.22	-42.0	0.0	0.0	0.0
2016-05-26 14:20:43.598	3.840938	3.92266	8.975809	0.69730	1.66033	0.7721336	6.12	-20.52	-41.879997	0.0	0.0	0.0
2016-05-26 14:20:43.608	2.73769	4.6036777	9.575105	0.69058	1.82618	1.0146471	6.66	-20.939999	-41.82	0.0	0.0	0.0
2016-05-26 14:20:43.618	0.92618364	4.9986677	9.41166	0.66889	1.74554	1.065349	7.08	-21.24	-41.64	0.0	0.0	0.0
2016-05-26 14:20:43.628	-0.095342435	5.339176	9.057531	0.50823	1.69118	1.024421	7.44	-21.48	-41.399998	0.0	0.0	0.0
2016-05-26 14:20:43.639	-0.55843425	4.8760843	8.798744	-0.21807	1.50059	1.1096367	7.74	-21.84	-41.1	0.0	0.0	0.0

Figure 8: *The CSV-file output of a LogYard logging session, parsed into spreadsheet form by the application Numbers.*

2.6 Medical Considerations

Aspects of the human body should be taken into consideration in the development of the algorithm. More specifically, how different accidents will affect the drivers ability to summon help and also expected actions he or she will take.

2.6.1 Patient Response Resulting From Accidents Involving Trauma to the Brain

A study presented in 1998 found that the most common injuries in ATV accidents in the United States were orthopedic, 53.2 % of the drivers were afflicted by (at least) one such injury. Second most common were head injuries, 40.8 % of all drivers hurt their head in some way (scalp wound, concussion etc.) [28]. Both these type of injuries can leave the driver unable to move or otherwise incapable of contacting emergency services. Concussions are particularly dangerous; some of its symptoms are: dizziness, seizures, trouble walking, weakness, numbness, decreased coordination, confusion and slurred speech [29]. These symptoms may not be instant, meaning that the driver may function normally for a short time period after the accident but later on become dizzy and fall or crash again. A specific danger with this kind of injury is that the driver may cancel the alarm raised by the smartphone application and/or inform the ICE that it was a false alarm and later on lose consciousness.

2.6.2 Movability Following a Serious Accident

Following any accident involving high velocity and sudden stops, broken bones are a common complication [30]. If a driver crashes several scenarios are possible, some involving broken bones, for the more extreme versions it is possible the driver can't walk at all, can't manipulate hands and fingers, puncture a lung or the skin. Most of these (extreme version) injuries result in a driver not capable of getting him- or herself the help needed.

2.7 Smartphone Application

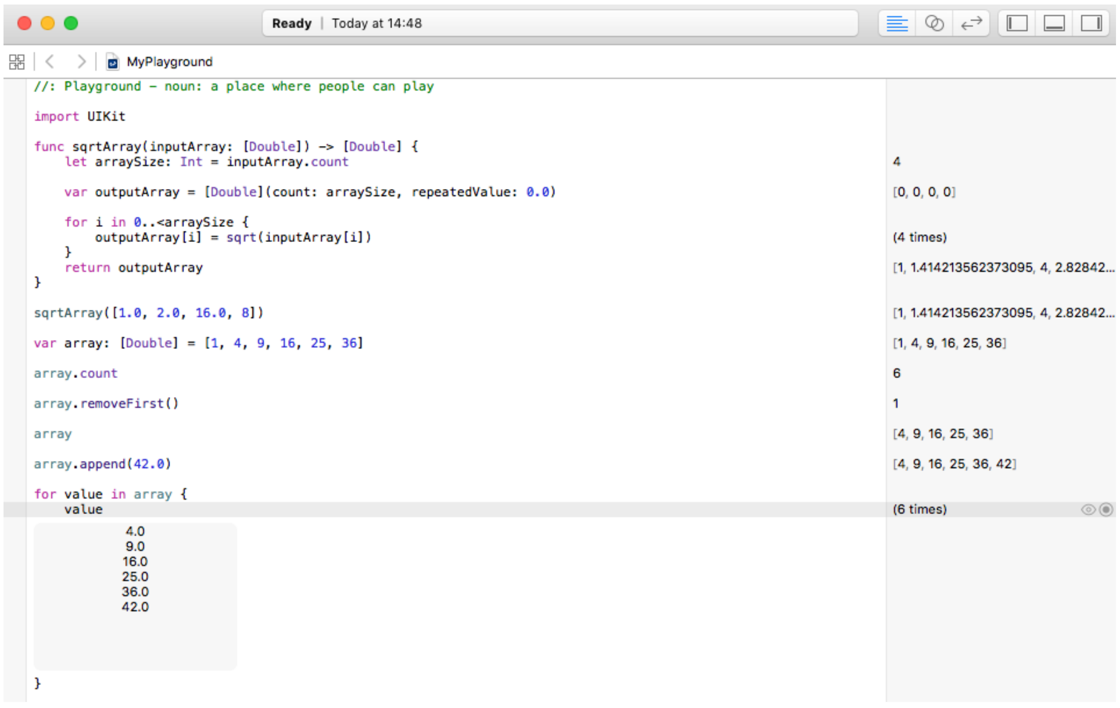
Since one aim is to release an application containing the finished IDA (see Section 1.3) investigations into smartphone application programming is needed as well. Several smartphone Operating Systems (*OS*) exist; Google's Android and Apple's iOS are the two largest smartphone markets [31], and online resources offer good support for novice application programmers and are therefore chosen.

2.8 Smartphone Application Development: Apple iOS

The development of applications for Apple iOS devices is done using Objective-C or *Swift*. Available and relevant functionality incorporated in the software development kit (*SDK*) are mentioned and briefly described in the following sections.

2.8.1 Apple iOS and Swift Programming

While traditionally Objective-C has been used to develop for the Apple range of operating systems, namely iOS, watchOS, tvOS and macOS; however, since 2014, it is also possible to program for said systems in the Apple-developed language called Swift. Swift builds upon Objective-C and C but has many simplifications which improve readability and makes programming easier, in particular for beginners. Swift programming in the Apple-supplied IDE, *Xcode*, supports the use of so called *Playgrounds* which evaluate each line of code written in real time so that it is possible to troubleshoot blocks of code in an isolated and controlled environment to see how it works [32]. A Playground example is illustrated in Figure 9. In the right margin of the playground, the resulting output from evaluating each line of code is shown and reevaluated in real time as the user writes or changes code. While programming in Xcode, an application may be executed on any connected iPhone which will communicate real time energy impact, CPU and memory consumption, and disk and network usage to Xcode so that app performance may be analysed [33].



```
//: Playground - noun: a place where people can play

import UIKit

func sqrtArray(inputArray: [Double]) -> [Double] {
    let arraySize: Int = inputArray.count                                  4

    var outputArray = [Double](count: arraySize, repeatedValue: 0.0)      [0, 0, 0, 0]

    for i in 0..<arraySize {
        outputArray[i] = sqrt(inputArray[i])                               (4 times)
    }
    return outputArray                                                     [1, 1.414213562373095, 4, 2.82842...
}

sqrtArray([1.0, 2.0, 16.0, 8])                                             [1, 1.414213562373095, 4, 2.82842...

var array: [Double] = [1, 4, 9, 16, 25, 36]                                [1, 4, 9, 16, 25, 36]

array.count                                                                6

array.removeFirst()                                                        1

array                                                                      [4, 9, 16, 25, 36]

array.append(42.0)                                                         [4, 9, 16, 25, 36, 42]

for value in array {
    value                                                                  (6 times)
        4.0
        9.0
        16.0
        25.0
        36.0
        42.0
}
```

Figure 9: *An Apple Xcode Playground for running Swift code.*

2.8.2 Using Apple's Accelerate Framework For Optimised Digital Signal Processing

The SDK for Apple's iOS includes an optimised package for *Digital Signal Processing* (DSP) called *vDSP* [34]. This package provides mathematical functions for, e.g, vector and matrix algebra, statistical analysis, and frequency analyses using Fast Fourier Transforms, and is part of an Apple-developed and computationally efficient C-based general framework for advanced mathematics known as *Accelerate* [35].

2.8.3 Built-in Activity Classification Algorithms in iOS

The iOS SDK already includes code for activity classification in a class called *CMMotionActivity* designed to identify the following activities: *stationary*, *walking*, *running*, *automotive*, *cycling* and *unknown*. The activity classifications also come with a three-parted level of confidence associated with the classification. When an activity is detected, a flag for that specific activity is set and the confidence level for the detection can be accessed. Several flags can be set at once, e.g if the user is driving a vehicle, the automotive flag will be set, and if said user stops the vehicle but does not get out, both the automotive and stationary flags will be set [36].

2.9 Smartphone Application Development: Google's Android

Android application development is done in Java, and several IDEs exist supporting Android packages. For this application, Google's *Android Studio* will be used since it is easily set up and developed specifically for Android development.

2.9.1 Theory Behind API-level

Since several different companies produce smartphones with Android as its OS, a system with different *API-levels* is used. When new functionality is introduced (such as communication with smart watches) with a major update of the Android software the API-level is raised. This is done for mainly two reasons:

1. Manufacturers design and produce phones according to demands placed on them by the API-level they want to fulfill, and can use the API-level to determine what hardware is needed, for example a company's flagship model containing all new functionality and some predicted for future updates generally has a higher API-level than the company's budget model.

2. Software developers choose for which API-level they develop their application, i.e. how old phones (and consequently, how many) they want to be able to run their application, a lower API-level will make the application available for more smartphones, but limit functionality for the application.

This means every Android phone has an API-level and every application has one too. If you combine one Android smartphone and one application, they will be a successful combo if the API of the smartphone is equal or greater that of the application. Android users of course doesn't need to do this comparison themselves, but will be notified if they try to install an incompatible application.

2.9.2 Built-in Activity Classification Algorithms in Android

Available via Google is a method for activity classification, which can predict the user's activity. The activities that the method is able to identify are: *in vehicle*, *on bicycle*, *on foot*, *walking*, *running*, *still*, *tilting* and *unknown*. If this method is called different results can be returned, for example the programmer can choose to only obtain the most likely activity in return, or all activities with a corresponding confidence. Activities are not mutually exclusive for several reasons, one being that some are versions of other (*walking* and *running* are a version of *on foot*) and another is that they are possible to combine in real life as well (i.e. to walk around in a bus, which would result in both *on foot* and *in vehicle* having high confidence) [37].

3 Method

Two main areas in this report are MATLAB programming and smartphone application development. MATLAB is used to both train and evaluate an IDA, before implementing it in an application. Since the algorithm will run on smartphones, some MATLAB functionality can't be used, and how data is handled has to be taken into account when both training and evaluating the algorithm, since it is done on collected data in MATLAB but will run in real time on the smartphones.

3.1 Previously Available Dataset

Prior to the start of this thesis, about 55 hours of normal ATV driving data from 20 different ATV drivers had been collected in another project [26]. Using the smartphone application LogYard, detailed in Section 2.5, data was collected by a group of 20 volunteer drivers using ATV's in their work (e.g in farming and forestry), or in their spare time. LogYard constantly samples and saves data from available motion sensors, i.e gyroscopes and accelerometers, as well as orientation and position sensors, i.e magnetometers and GPS. Data is generally sampled at a rate of about 100 samples per second; however, due to the way smartphone CPU time is divided between applications, the sampling rate may fall well below this level. The previous study found that approximately 8 of the 55 hours had to be omitted due to inconsistencies, leaving 47 hours of data to be analysed. Out of these 47 hours of data logs, about 15.5 hours of logs were found to be affected by file corruption or inconsistencies, leaving 31.5 hours of data logs available in total for this thesis.

3.2 Data Evaluation and Classification

LogYard samples twelve different sensors, as mentioned in Section 2.5, however not all sampled sensor values were used for this project. Several different smartphone models has been used to collect normal driving data and, from visually analysing data from different sensors, some conclusions were drawn concerning sensor aptitude for the application;

- Magnetometer sensor quality seems to differ greatly among manufacturers. Quality of sensor values varies with such a degree, that using it as an input to the OC-SVM might cause misclassifications dependent on the smartphone model and therefore aren't used. Also, magnetometers are sensitive to magnetic fields generated even by small nearby electronic devices.

- GPS locations (longitude and latitude coordinates) are important in case an accident has occurred, but otherwise too uncertain. Sometimes the signal is lost completely, other times the estimated position is off by several hundred meters, GPS location is only used to communicate an accident location.
- Estimated speed (derived from the GPS data) is as uncertain as GPS data itself. Due to sometimes being off by hundreds of meters and sometimes losing the signal, improbable speed changes occur in the data; thus, estimated speed by GPS data isn't used as an input the OC-SVM.
- Accelerometer sensors are generally robust and not prone to disturbances, they are therefore quite independent of the smartphone model that the driver has. A lot of information about the drive can be found in the accelerometer values and are one of the most important inputs to the OC-SVM.
- Gyroscope sensors, like accelerometers, are robust enough not to depend on smartphone model; however, they do usually produce a bias term in their signal which must be filtered out to obtain accurate measurements. Due to the nature of a normal drive, only certain events can be found within gyroscope values. However, a roll accident logged by a gyroscope sensor has a quite unique pattern and the sensor is a valuable input to the OC-SVM.

Some feature extraction from accelerometer and gyroscope data is done as well, see Section 3.2.4 for more information about input parameters to the OC-SVM.

Due to the fact that LogYard starts logging immediately after a user manually enables it in the smartphone application, and that it continues up until the point when the user manually stops it, unwanted data is unintentionally included in every log. Unwanted data is composed mainly of motion data resulting from smartphone interaction related to starting and stopping the logging, and of motion data resulting from the user mounting or dismounting their ATV. Since neither of these cases are similar to normal driving, it is undesirable to train a normal driving classifier on these.

Labels for the data don't exist, and it is most likely that other activities besides ATV driving have been logged, which has to be handled somehow. A likely activity to be among the data is walking, since it is necessary to get to and from the ATV in order to drive it.

3.2.1 Removal of Accidentally Logged Walking Data: Preprocessing of Data

Since it was desirable to train on largely pure driving data, it was necessary to attempt to remove data that weren't driving data, such as data created when drivers forgot to turn off data logging after getting off their vehicles. The absence of labels corresponding to the activities logged in the dataset lead to the identification and removal of irrelevant data being left up to the authors.

In order to facilitate removing of data, different preprocessing methods were tried:

Preprocessing method 1 (not used): Although the data mentioned in Section 3.1, was assumed to be of sufficient quality, some evaluation of the data was also required. With the help of MATLAB, data with too low sampling frequency was removed (varying thresholds were tried in the interval $40 - 60$ Hz). Also samples where too many sensors, or one sensor for a too long time, aren't updating their values, are scrapped (here, also, variations were tried, the best seemed to be all sensors locked up or one sensor for 100 ms). A window length was also implemented, meaning that if a continuous area of good data was too short it were also scrapped. Every logged drive were evaluated and data sections not fulfilling the demands were removed, meaning a single logged drive could be split into several datasets, since most drives fit more than just a window length of continuous data. A segment of bad data means that the data before the first bad sample is saved as one interval and the first good sample after the bad segment is the first of a new interval.

Preprocessing method 2 (used): To assist the method, 200 seconds of data were removed from both start and end of each logged drive as it was assumed that the likelihood of the drive having started was high after 200 seconds. This data trimming was done for several reasons; to remove abnormal samples originating from the driver turning on/off logging, removing areas of uncertain activity, such as mounting/dismounting the ATV for example and train on longer datasets. This left 21 hours of data from the 31.5 hours mentioned in Section 3.1, which around 1.5 hours were used for training and the rest for testing.

3.2.2 Removal of Accidentally Logged Walking Data: Removal of Data

Removal method 1 (not used): It was assumed that most or all of the irrelevant data would consist of short periods of walking until the drivers realised that logging was still taking place; therefore, an algorithm was developed to identify and classify walking data only. About 80 minutes of walking data was collected using LogYard with varying positioning of the smartphone (trouser pocket, shirt pocket and in a handbag) and subsequently screened to remove sections of the data with low sampling rates. This dataset was known to contain only walking and was used to develop the classification algorithm. However the classification algorithm proved to remove too much data, leaving only a fraction for training. Examining the results, it seemed small groups or single samples were removed here and there in the datasets, resulting in other criteria failing later on in the evaluation process.

Removal method 2 (not used): A signal to noise ratio (SNR) approach was also tried, with the help of accelerometer data and Fast Fourier Transform (FFT) on this data. Lower frequencies were considered signal, and remaining frequencies noise, since walking typically is a low frequency activity. All frequencies below a certain threshold were summarised and considered the signal, while the remaining also were summarised but considered noise instead. To determine whether the current sample was walking or not, the ratio between signal and noise were calculated by dividing the signal with the noise, and if this ratio reached a certain level it was considered walking. Similar to previous methods, too much data seemed to be removed from the normal driving dataset, most likely due to miss-classification.

Removal method 3 (used): It was assumed that the preprocessing procedure would suffice, since the volunteer drivers had been expressly told to collect driving data [26]. The parameter ν (see Section 2.4.2) was adjusted during OC-SVM training in iterations. In every iteration, the OC-SVM was used to classify test data to see how much of it was considered normal driving. The method is detailed in Section 3.2.4.

3.2.3 Using the Matlab Function *fitcsvm* to Train an OC-SVM

In MATLAB's *Statistics and Machine Learning Toolbox* the function *fitcsvm* could be used to train an SVM from a set of training observations containing different features and a vector with labels for each row (i.e. sample). Training an OC-SVM in lieu of the default two-class SVM involved labeling all observations equally, e.g as belonging to the only available class, and tuning parameters exclusively intended for one-class learning. Barring Kernel selection, i.e selecting and/or modifying the non-linear Kernel function, which is essential in both two-class SVM and OC-SVM training, the main parameter to tune in one-class learning is the level of regularisation in training, here symbolised by the Greek letter ν (nu). A small ν heavily penalises misclassifications during training, i.e observations which do not fall within the class boundaries, which may lead to overfitting. In turn, a large ν produces simpler solutions with possibly better generalisation abilities; however, this may lead to underfitting, i.e that the trained SVM does not properly classify new cases (See Section 2.4.2). Fine tuning the ν parameter is essential for the final SVM to produce satisfactory classifications.

Experimentation with training an OC-SVM on example observation sets generated by a known mathematical function and subsequently using the trained OC-SVM to classify equivalent observation sets with different levels of added noise showed several important points. Firstly, training an OC-SVM on raw data produced much higher misclassification rates than on data which was scaled, either by mean-variance normalisation or by mapping the values of each feature to similar ranges and, secondly, that large differences in the ranges of features undermined the effect of ν-parameter tuning. Also, the experiments confirmed that increasing ν at training time resulted in a larger tolerance for noise in the classifications, as well as increasing the acceptance of observations generated by different mathematical functions. The Kernel used throughout was the Radial Basis Function (RBF) Kernel which is commonly used, and is advocated as a primary alternative by a 2010 report [38].

3.2.4 Training an OC-SVM Classifier on Normal Driving Data

Due to time constraints and the large amount of observations available (around 21 hours), an OC-SVM was trained on a small fraction of the total available data (around 1.5 hours). However, in order to avoid training only on one or a few sets of user data, and to avoid training on breakpoint areas between logs, a script was developed which produced an adequate training set. Concisely, the script first loaded all user data, categorised by user and log number. Then, a short period of time was clipped from the start and end of each log as mentioned in Section 3.2.1. The resulting logs were examined in order to produce feature vectors for each observation, using mean-variance normalisation to obtain adequate value ranges for each feature. Each new observation was added to a larger dataset which would include all observations. Finally, the observations' orders in the large dataset were randomised several times, using a randomisation script built around the Matlab random number generator. A fraction (1.5 hours) of the randomised set was reserved as training data, while the other, much larger (19.5 hours), fraction was saved as a test set to act as observations not before seen by the OC-SVM.

Several different variations to the method were tested, among others: feature selection, historical sample influence on features, as well as the number of historical samples to consider. However, the parameter which governed at training time was the ν-parameter described in Section 3.2.3. Additional parameters were implemented post-training in order to simplify individual run-time adaptation within the later smartphone application. One such parameter is a feature weighting vector which can tweak the run-time sensitivity to certain scenarios.

In the final OC-SVM solution, the input feature vector was comprised of data from accelerometers, and gyroscopes, both "current" and historical. Accelerometer and gyroscope data was preprocessed, separately for each axis of the 3-axis data. An estimated derivative of the gyroscope data is also included in order to let the OC-SVM specifically keep track of quick turns, and a historical measure for each parameter was saved as a new feature which would also go into the OC-SVM feature vector. Five seconds of history is used in an attempt to balance the two goals of catching inconsistencies in the short time leading up to an incident, while not drowning these inconsistencies in large amounts of historical data. A ν value of 0.75 was found to be a rough optimum in that it produced the lowest false alarm rates for the normal driving datalogs while still detecting all simulated accidents described in Section 3.2.5.

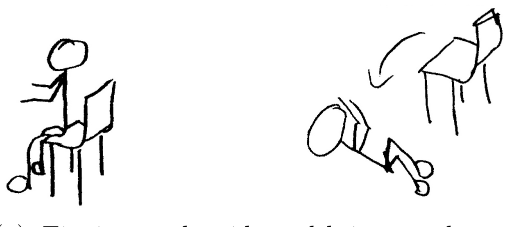
3.2.5 Collection of Simulated Crash Data

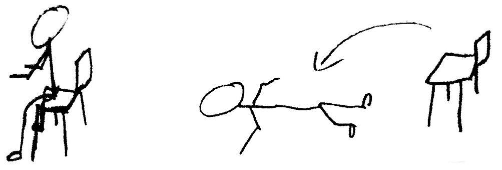
In order to obtain data for evaluation purposes, a number of abnormal movement patterns were realised while collecting sensor data in a smartphone. The initial experiment consisted of placing the smartphone in a rucksack and subsequently subjecting it to impacts, falls and rolling, separated by short periods of rest or carrying the rucksack around. A video was recorded during testing and the different events were later identified and marked in the sensor data by comparing the timestamps in the data with the elapsed times pertaining to each event in the video.

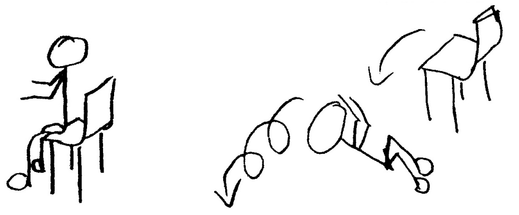
A second experiment was also realised, where an operator performed a number of abnormal movements with two smartphones placed on their person, one in a trouser pocket and one in a chest pocket, both simultaneously collecting sensor data. The goal of this experiment was to collect a larger timeseries of approximately naturalistic data, i.e. having a real person perform movements similar to accidents rather than subjecting a rucksack to extreme movements, in order to evaluate the performance of the IDA. The simulated events were as follows:

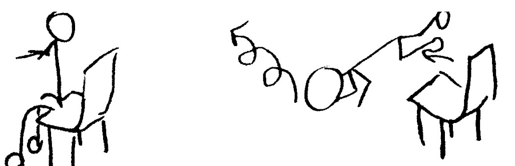
1. Tipping to the side and being caught underneath the vehicle.
2. Being thrown over the handle.
3. Rolling off the vehicle to the side.
4. Rolling off the vehicle in a forward direction.
5. Unintentionally performing a wheelie and falling backwards.

The different motions, apart from the backwards fall, were performed using a chair placed on a lawn, see Figure 10. The acts of falling off towards the sides, as well as being thrown forward, were simulated by an operator throwing their body in either direction from a sitting position in the chair, landing flat on the grass and laying still for a short period of time. Tipping and rolling were simulated by tipping or rolling off of the chair, and by rolling a few times on the grass. The backwards fall was simulated by sitting on a stool placed in front of a mattress. An operator sat on the stool with their back towards the mattress, and also slightly elevated above the mattress, and subsequently let themselves fall backwards onto the mattress. These experiments were also filmed to allow for later event identification and marking in the data. Sadly, data collected by the smartphone in the trouser pocket were deemed poor, due to long sequences where no sensor's value were updated, probably due to a higher priority application running simultaneously.


(a) *Tipping to the side and being caught underneath the vehicle.*


(b) *Being thrown over the handlebar.*


(c) *Rolling off the vehicle to the side.*


(d) *Rolling off the vehicle in a forward direction*

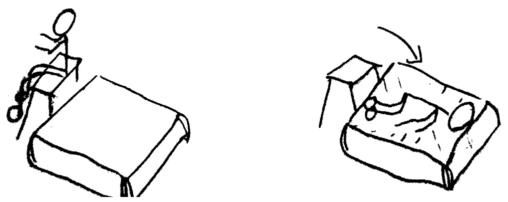

(e) *Unintentionally performing a wheelie and falling backwards.*

Figure 10: *Simulated accidents, how they were performed and which type of accident they simulate.*

3.2.6 Collection of Potentially Problematic Data

An experiment was done with the goal of evaluating the possibility to avoid false alarms during common actions which may seem abnormal to the IDA (i.e. not normal driving). Such actions include mounting and dismounting the ATV, as well as interacting with the smartphone running the IDA. As some form of smartphone interaction is captured per definition in the beginning and end sections of all data, no more smartphone interaction motion data was collected; however, data was collected for the act of mounting and dismounting several times from different directions. As an ATV was not available, a stone wall of similar height and width was used in order to obtain approximately naturalistic data. The motions were performed with two smartphones simultaneously collecting sensor data, one in a trouser pocket and the other in a chest pocket. The experiment was filmed in order to later identify and mark each event in the data. As with the simulated crash data, the trouser pocket data where deemed poor for the same reason.

3.2.7 Summarisation of Data

In Table 3 a summarisation of data used in this project can be seen. All data presented in the table has been subject to vetting, i.e. segments of deficient data has been removed.

Table 3: *Summarisation of good quality data used in this project.*

Type of data	**Quantity**	**Usage**	**Comment**
Normal driving	~21 hours	Train the OC-SVM	Collected by several volunteers.
Simulated accidents	20 accidents	Evaluate IDA performance	20 accidents occurring back-to-back.
Problematic movements	4 mountings & 4 dismounts	Evaluate IDA performance	Simulated using a stone wall.
Walking	~80 minutes	Evaluate IDA performance	Originally collected for a separate algorithm to remove accidentally logged data.

3.3 Notification of Emergency Services in Case of Accident

In Sweden a project started in 2006 that lets people with limited hearing and/or speaking abilities register their mobile phone with the national emergency service provider, SOS Alarm [39], which lets individuals and emergency operators communicate with each other via text message. This service is not available for everybody, since the project is still in an early stage and communicating emergencies via text messages has turned out to take longer time than a regular phone call. For this project, the optimal solution would of course be to send a text message directly to SOS Alarm with last known location and other vital information. However, the infrastructure used to receive text messages is dedicate for those with disabilities and a smartphone application that can automatically send text messages could potentially overload the system.

In order to actually communicate the accident information, the driver must choose an ICE before the application can be used. This ICE will receive a message (note: not necessarily a text message) with the appropriate information together with the task of contacting emergency services and/or finding the driver.

3.4 Smartphone Application Development: General Idea of Program Operations

To easier grasp how an application would work a flow chart is presented here, see Figure 11. Some steps are not included and others require further explanation.

This list explains functionalities of different boxes in Figure 11, the bold words corresponds with the label of one or more boxes in the figure.

Application launched User launches the application, but the IDA isn't launched.

Wait for IDA to be started (by user) It should be possible to start the application without running the IDA, if the user wants to change some settings for example. Is the IDA stopped for some reason (turned off by the user) but the application is still running, this becomes the "active" state.

Is the screen off? *(left column)* An assumption was made that the driver isn't using the smartphone actively while driving, so if some interaction is detected (here in the form of a lit screen) the IDA shouldn't be running.

Wait for d1 sec To reach this box, the user has recently interacted with the smartphone but now stopped, and in order to avoid generating a false alarm from the driver mounting the ATV a delay of *d1* seconds was added.

Is SVM triggered? The first step of the IDA is an OC-SVM which, since it is trained with high sensitivity as a priority, results in some false positive predictions (preferred over false negatives which would mean missing an incident). If the OC-SVM isn't triggered an interaction check is made.

Start timer1 As a positive prediction is made, a timer is started to limit how long it takes to confirm the prediction.

Is ACC triggered? If the ACC[1] are not fulfilled in less than *x1* seconds the IDA starts to check the OC-SVM predictions again.

[1]The ACC will not be described in this report.

Start timer2 After a positive confirmation of the prediction a second timer is started, because the potential alarm might still be rejected.

RC triggered? If the prediction is confirmed there still remains scenarios where an alarm isn't suitable. The RC must be triggered in less than *x2* seconds for the alarm to be canceled.

Wait for d2 sec In order to give the IDA time to collect a sufficient amount of data for the next step a delay is implemented.

Walking detected? Since alarms shouldn't occur if the driver dismounts the ATV and starts walking, a continuous check is made if walking is detected and the IDA is paused until the driver stops walking.

Is the screen off? *(right column)* This check is an extra rejection criteria, if all other predictions indicate an accident, a last check to see if the driver is interacting with the phone is made before continuing.

Notify user, Start timer3 In order to give the driver a chance to cancel a false alarm, the phone starts to notify the driver that it is about to send an alarm. A timer is also started to limit the notification period.

Any user input? If the driver interacts with the application (note: application, not smartphone), the notification is stopped. Depending on the input two different actions are executed.

Ask for feedback In case of a false alarm, the driver is asked to give feedback on the situation in order to further improve the IDA. A possible automatic improvement could be to alter thresholds depending on this feedback.

Send alarm This box is reached if there is no interaction from the driver, or if manually pressed by the driver. A message is transmitted containing vital information (such as last known location, time stamp, and some suggestions of actions) to an ICE.

Post alarm mode After an alarm has been sent, further actions can benefit both the driver and emergency personnel. One possibility is to send messages with fixed intervals to get the attention of the ICE (if the first message failed to do this), also GPS positions may vary and a more accurate position might have been acquired. Another possible action is to make the phone sound to draw the attention of passerby to the site (or, if the phone is dropped by the driver this would indicate where it is).

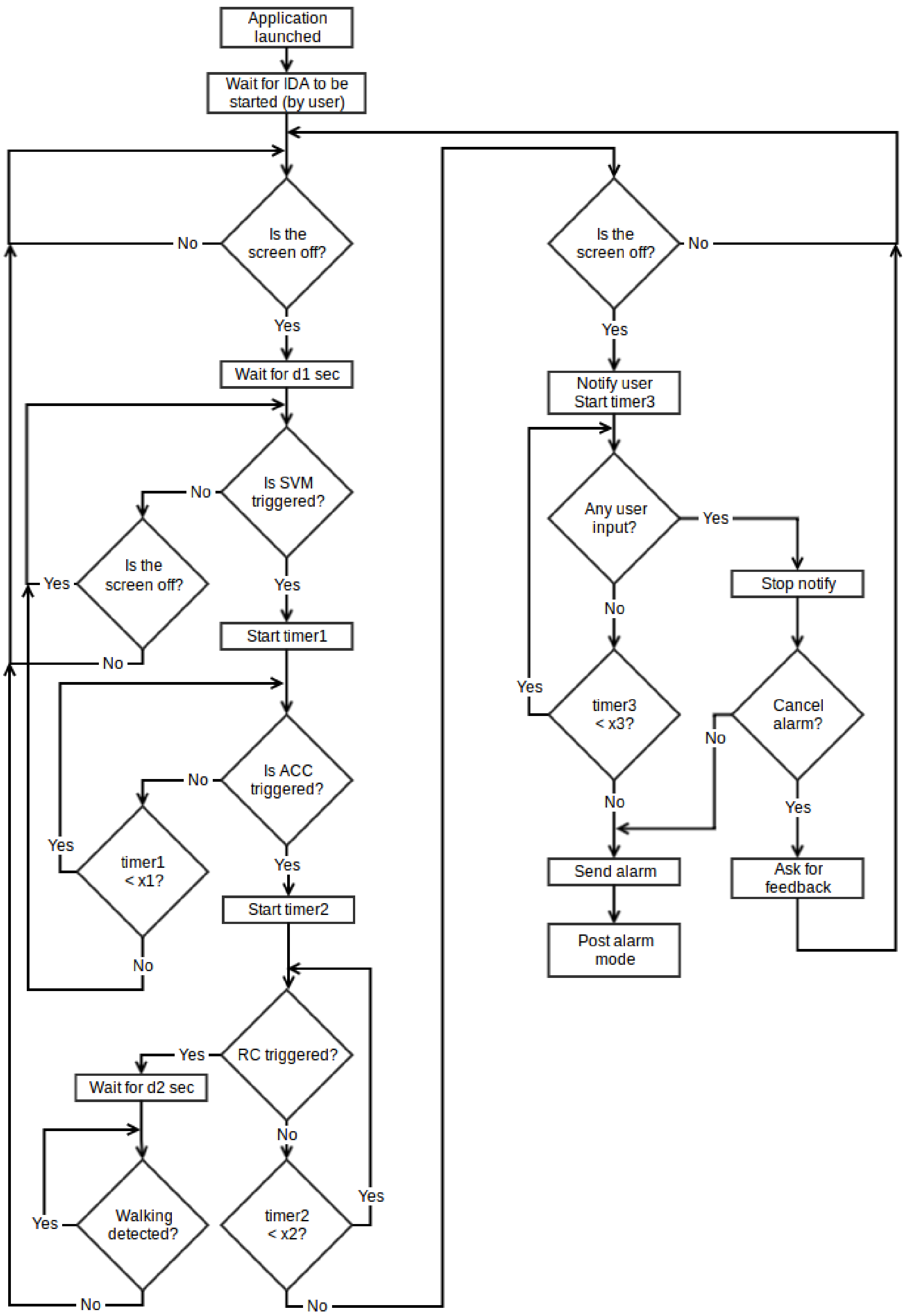


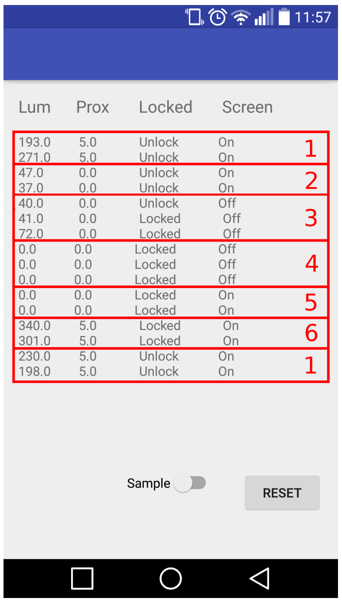
Figure 11: *Flow chart of general smartphone application.*

3.5 User Smartphone Interaction and Undesired Triggering of Alarms

By choosing smartphones for motion data collection (rather than a separate unit) a problem arises due to the multipurpose machine a smartphone is. Regular usage of a smartphone, such as texting, answering phone calls and playing games, creates sensor motion patterns that are closely related to incidents and crashes (for instance, large acceleration under short periods of time).

To counter this implementation issue, a test program (for Andriod) was created to determine which sensors and parameters could be used and how. Seen in Figure 12 is a screen shot of said test program, two sensors (both located on the front of the smartphone) and two parameters set by the OS are sampled and their values presented:

- *Lum* corresponds to the light sensor on the phone and is presented in *lux*.
- *Prox* represents the proximity sensor, which the phone mainly uses to determine where the phone is during a telephone conversation (close vs not close to the head), and are presented in *cm*.
- *Locked* is the first parameter, in Android it is possible to extract if the phone is locked or not (even if no password is needed to access the smartphone, a "lock screen" exists).
- *Screen* is the second parameter and shows whether the screen is lit or not.



Lum	Prox	Locked	Screen	
193.0	5.0	Unlock	On	1
271.0	5.0	Unlock	On	
47.0	0.0	Unlock	On	2
37.0	0.0	Unlock	On	
40.0	0.0	Unlock	Off	3
41.0	0.0	Locked	Off	
72.0	0.0	Locked	Off	
0.0	0.0	Locked	Off	4
0.0	0.0	Locked	Off	
0.0	0.0	Locked	Off	
0.0	0.0	Locked	On	5
0.0	0.0	Locked	On	
340.0	5.0	Locked	On	6
301.0	5.0	Locked	On	
230.0	5.0	Unlock	On	1
198.0	5.0	Unlock	On	

Figure 12: *Test program for different sensors used to determine user action.*

Enclosed in red rectangles are six different scenarios, they are as following:

1. The phone is unlocked and rests in an operator's hand, exposed to normal office light with a lit screen and nothing is close to the proximity sensor.
2. From the previous state the phone is raised up to the head, to mimic the placement during a phone call.
3. In the same position, the screen is manually turned off. A specific setting of this smartphone is that it isn't locked immediately after the screen is turned off.
4. The phone is placed in a trouser pocket.
5. Still in the pocket, the screen is turned on but remaining locked.
6. The smartphone is brought out of the pocket, with the screen lit while still locked.

All these different scenarios create unique sets of sensor and parameter combinations. Other possibilities include, e.g, the same scenarios, but in a dark room. However, for a first version of a commercial application just a few things are necessary, to be able to identify when the smartphone is placed in a pocket and when the user interacts with the phone (which is simplistically represented by a lit and unlocked screen). So with the four sensors and parameters listed, enough information is extractable to handle user smartphone interaction.

3.6 Smartphone Application Development: Apple iOS

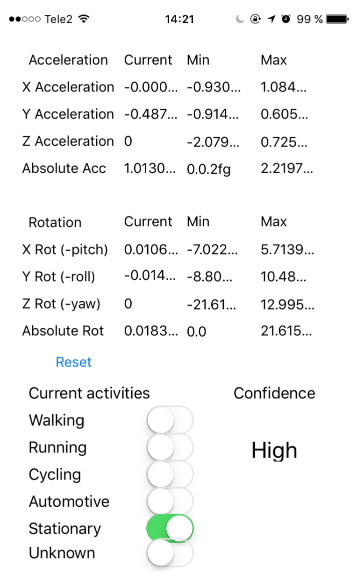
In order to evaluate the feasibility of an SVM-based IDA running in real time on an Apple iPhone, a simple application was designed to incorporate some of the functionality which would likely be necessary or valuable to the final app.

3.6.1 Sensor, Activity Detection and Calculation Performance Testing Application

A simple iOS application was built to sample sensor data at a rate of 100 Hz and display sensor values in real time as well as the results of built-in activity recognition data. The application displayed current, max and min values of each axis of the accelerometer and gyroscope, respectively, as well as for the absolute values calculated at each sampling instance for the accelerometer and gyroscope axes, respectively. A set of switch icons in the application showed which of the following activity flags were set: Walking, Running, Cycling, Automotive, Stationary and Unknown. The built-in level of confidence for the current activity state was also displayed in the application, next to the set of switches. A screen shot taken with the device, an iPhone 6, lying still on a desk is displayed in Figure 13a.

In order to assess the performance of the built-in activity recognition, an operator brought the device onto a tram, travelled for two stops, and then got off. This was documented using screen shots, the first of which, as seen in Figure 13b, depicts activity states when the operator was on the moving tram. The activity recognition estimated, with a high level of confidence, that the operator was in a vehicle. In the subsequent case seen in Figure 13c, the tram was stationary at a tram stop. The activity recognition correctly estimated that the operator had not moved off the tram and was thus inside a stationary vehicle; however, the confidence level was at a lower level. Seconds after the tram started moving again, the screen shot of Figure 13d was captured, which shows clearly that the activity recognition correctly determined that the vehicle was moving again. Finally, when the screen shot of Figure 13e was taken, the operator had just gotten off the tram and had taken a few steps towards the booth at the tram stop, thus leading the activity recognition to estimate that the operator was walking, albeit with initially low confidence.

Initial testing showed that an iPhone 6 had no problem running the application and updating the sensor values in real time; in fact, historical activity recognition results are available regardless of whether the application is running, as the device actually already collects and saves historical data by default [40]. Accounting for the possibility that frequency analysis would improve performance of the IDA, a modified version of the application ran a FFT in real time at every sampling instance, normalised the results and subsequently located and printed the value of the highest peak. Running the application with the device connected to the Xcode programming environment showed that battery consumption classified as low, which suggests that sensor monitoring and frequency analysis will not seriously impact battery life.


●●○○○ Tele2   14:21   99 %

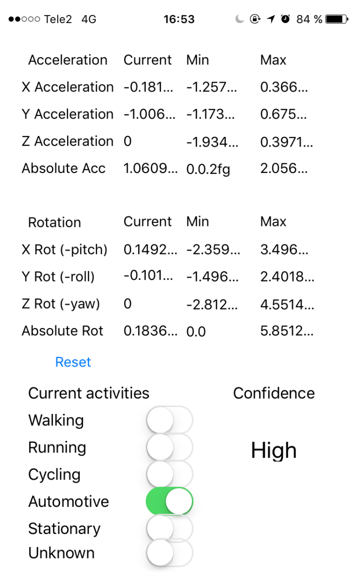
Acceleration	Current	Min	Max
X Acceleration	-0.000...	-0.930...	1.084...
Y Acceleration	-0.487...	-0.914...	0.605...
Z Acceleration	0	-2.079...	0.725...
Absolute Acc	1.0130...	0.0.2fg	2.2197...

Rotation	Current	Min	Max
X Rot (-pitch)	0.0106...	-7.022...	5.7139...
Y Rot (-roll)	-0.014...	-8.80...	10.48...
Z Rot (-yaw)	0	-21.61...	12.995...
Absolute Rot	0.0183...	0.0	21.615...

Reset

Current activities		Confidence
Walking		
Running		High
Cycling		
Automotive		
Stationary	✓	
Unknown		

(a) *Phone was lying still on a desk.*


●●○○○ Tele2 4G   16:53   84 %

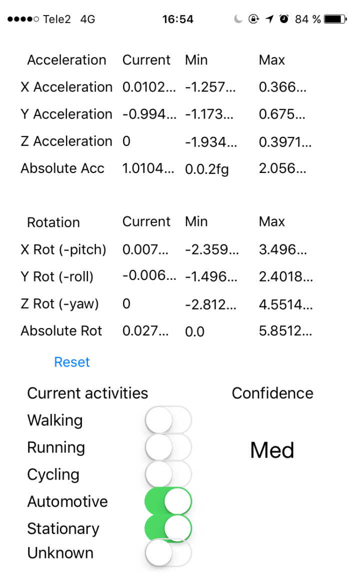
Acceleration	Current	Min	Max
X Acceleration	-0.181...	-1.257...	0.366...
Y Acceleration	-1.006...	-1.173...	0.675...
Z Acceleration	0	-1.934...	0.3971...
Absolute Acc	1.0609...	0.0.2fg	2.056...

Rotation	Current	Min	Max
X Rot (-pitch)	0.1492...	-2.359...	3.496...
Y Rot (-roll)	-0.101...	-1.496...	2.4018...
Z Rot (-yaw)	0	-2.812...	4.5514...
Absolute Rot	0.1836...	0.0	5.8512...

Reset

Current activities		Confidence
Walking		
Running		High
Cycling		
Automotive	✓	
Stationary		
Unknown		

(b) *Riding the tram.*


●●●●○ Tele2 4G   16:54   84 %

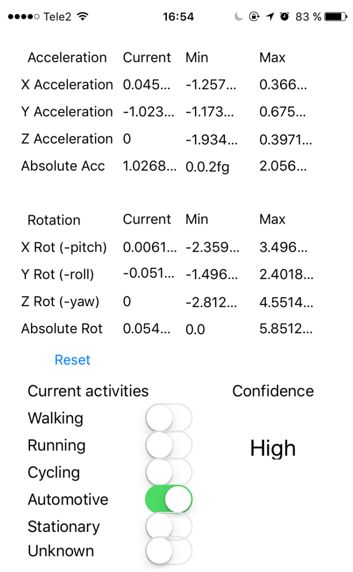
Acceleration	Current	Min	Max
X Acceleration	0.0102...	-1.257...	0.366...
Y Acceleration	-0.994...	-1.173...	0.675...
Z Acceleration	0	-1.934...	0.3971...
Absolute Acc	1.0104...	0.0.2fg	2.056...

Rotation	Current	Min	Max
X Rot (-pitch)	0.007...	-2.359...	3.496...
Y Rot (-roll)	-0.006...	-1.496...	2.4018...
Z Rot (-yaw)	0	-2.812...	4.5514...
Absolute Rot	0.027...	0.0	5.8512...

Reset

Current activities		Confidence
Walking		
Running		Med
Cycling		
Automotive	✓	
Stationary	✓	
Unknown		

(c) *Tram stationary at a tram stop.*


●●●●○ Tele2   16:54   83 %

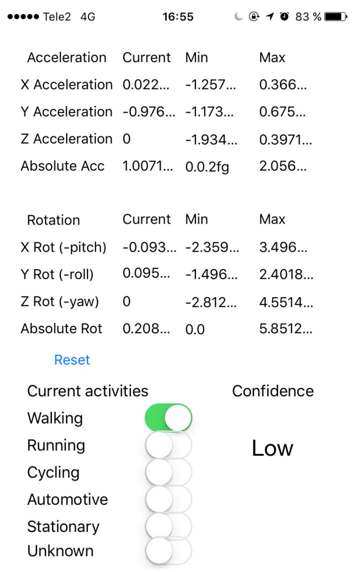
Acceleration	Current	Min	Max
X Acceleration	0.045...	-1.257...	0.366...
Y Acceleration	-1.023...	-1.173...	0.675...
Z Acceleration	0	-1.934...	0.3971...
Absolute Acc	1.0268...	0.0.2fg	2.056...

Rotation	Current	Min	Max
X Rot (-pitch)	0.0061...	-2.359...	3.496...
Y Rot (-roll)	-0.051...	-1.496...	2.4018...
Z Rot (-yaw)	0	-2.812...	4.5514...
Absolute Rot	0.054...	0.0	5.8512...

Reset

Current activities		Confidence
Walking		
Running		High
Cycling		
Automotive	✓	
Stationary		
Unknown		

(d) *Tram moving again after having stopped.*


●●●●● Tele2 4G   16:55   83 %

Acceleration	Current	Min	Max
X Acceleration	0.022...	-1.257...	0.366...
Y Acceleration	-0.976...	-1.173...	0.675...
Z Acceleration	0	-1.934...	0.3971...
Absolute Acc	1.0071...	0.0.2fg	2.056...

Rotation	Current	Min	Max
X Rot (-pitch)	-0.093...	-2.359...	3.496...
Y Rot (-roll)	0.095...	-1.496...	2.4018...
Z Rot (-yaw)	0	-2.812...	4.5514...
Absolute Rot	0.208...	0.0	5.8512...

Reset

Current activities		Confidence
Walking	✓	
Running		Low
Cycling		
Automotive		
Stationary		
Unknown		

(e) *Just getting off the tram.*

Figure 13: *Testing of the built-in activity recognition of an Apple iPhone 6, using a test application developed in this thesis.*

3.6.2 Automatic ICE Notification in Apple iOS

In iOS it is not possible to automatically send neither a text message or an e-mail. It is possible to display a message composition view with populated recipient, subject and message body fields, but actually sending the message requires user interaction [41]. However, the *CFNetwork* library supports both open and authenticated communication over HTTP and FTP; thus, an external server can be used to handle sending of text messages and e-mails [42].

3.7 Smartphone Application Development: Google's Android

Before the IDA could be implemented some test applications were programmed. Some served the purpose to try different functionalities that most likely would be incorporated in the final application, others were used to evaluate existing functionality and if it could be used instead of creating similar methods, thus saving time.

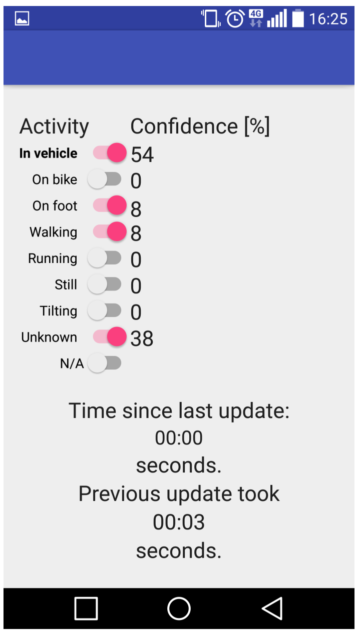
3.7.1 Choosing of API-level

Which API-level the application should be developed for is the first decision when programming for Android (see Section 2.9.1). Regularly (about once a month) Google collects information of devices visiting the *Google Play Store* and presents different statistics, one of which is the ratio of API-levels [43]. At the writing moment, an API-level of 15 will cover 97.3 % of all Android devices that visit the Google play store, and is chosen as this project's API-level. In perspective, the API-level was raised to 15 on December 16, 2011 as a second update to the *Ice cream sandwich* package [44]. On October 5, 2015 the API-level was updated to 23 [45] with the *Marshmallow* package, which is the highest level so far. If possible, the API-level might be lowered in order to support more devices, although this is only possible if new functionality in API-level 15 isn't used.

3.7.2 Built-in Activity Classification Algorithms in Android

Activity classification in Android is dynamically updated. This means that a programmer can't obtain activity classification information at any given time. Instead, a request has to be sent, and a result is returned once the activity classification determines it has useful/trustworthy information. A simple application was made, which displayed the current classifications as on/off switches next to their respective confidence levels. A timer also displayed how long it had taken to reach the current classification and how much time had passed since it was updated (see Figure 14 for a screen shot of the application). In experiments conducted using this built-in classification, requests were sent asking for results as soon as possible, which were obtained every 2 to 40 seconds. If the activity exercised by the operator was constant and continuous, updates came every 3 seconds on average, however if the activity changed (for example the operator stopped walking and just stood still) a new update could take up to 40 seconds to arrive, and often with a low confidence of the new activity (barely over 50 %).

In Figure 14 a series of screen shots taken demonstrate how a bus stopping at a station is classified. Before the screen shot presented in Figure 14a "In vehicle" was (correctly) classified with a confidence of 98 % and rather quickly updated to the result presented in Figure 14a when the bus stopped at a bus stop. This result is still correct, but fails to identify that the bus has stopped. In Figure 14b a more correct assessment of the situation is presented, however with low confidence meaning without knowledge of prior results it is not that trustworthy. Figure 14c presents a more trustworthy and correct situation, however a long time (27 seconds) is needed to get a confidence barely over 50 %. After a long update (such as the one presented in Figure 14c) a more confident result will quickly be updated (if the new activity is continued), as seen in Figure 14d, meaning the classification algorithm seems to have trouble dealing with quick changes. When the bus starts to move again a new update, Figure 14e, is acquired which has failed to catch the change of activity (or rather, failed to identify the new activity since "Still" has a low confidence, which is correct). After another rather long time (20 seconds) a correct result is presented, as seen in Figure 14f.

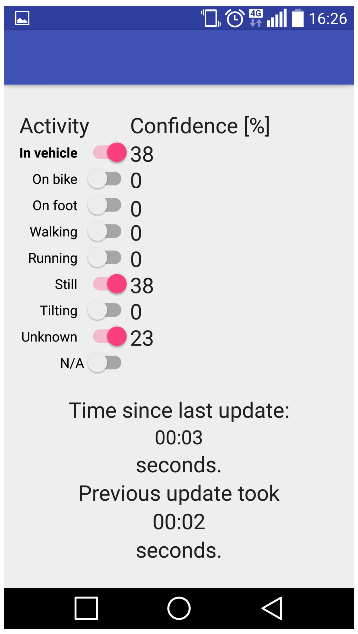


16:25

Activity		Confidence [%]
In vehicle	●	54
On bike	○	0
On foot	●	8
Walking	●	8
Running	○	0
Still	○	0
Tilting	○	0
Unknown	●	38
N/A	○	

Time since last update:
00:00
seconds.
Previous update took
00:03
seconds.

(a) *Bus stopped at station.*

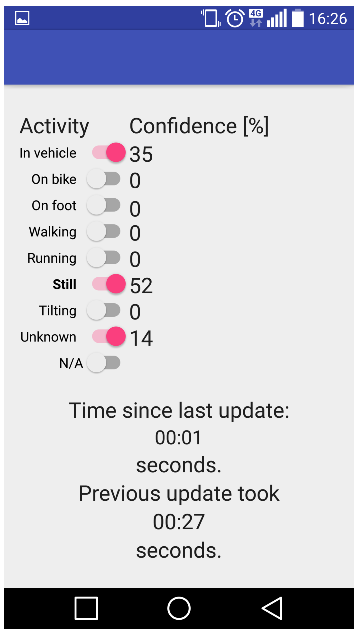


16:26

Activity		Confidence [%]
In vehicle	●	38
On bike	○	0
On foot	○	0
Walking	○	0
Running	○	0
Still	●	38
Tilting	○	0
Unknown	●	23
N/A	○	

Time since last update:
00:03
seconds.
Previous update took
00:02
seconds.

(b) *First update after stop.*

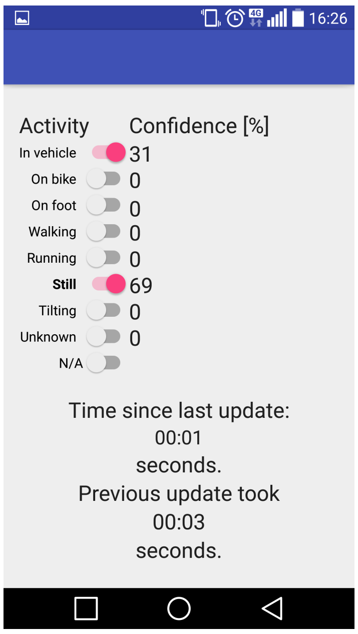


16:26

Activity		Confidence [%]
In vehicle	●	35
On bike	○	0
On foot	○	0
Walking	○	0
Running	○	0
Still	●	52
Tilting	○	0
Unknown	●	14
N/A	○	

Time since last update:
00:01
seconds.
Previous update took
00:27
seconds.

(c) *Second update after stop.*

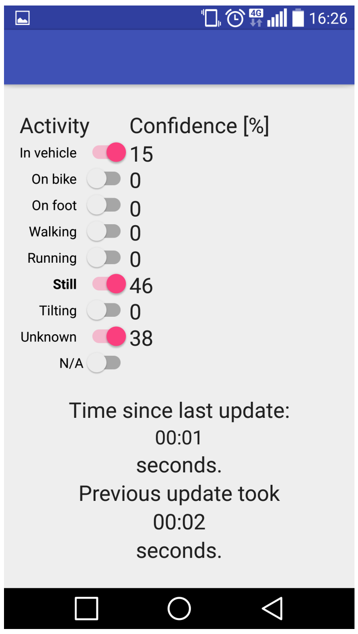


16:26

Activity		Confidence [%]
In vehicle	●	31
On bike	○	0
On foot	○	0
Walking	○	0
Running	○	0
Still	●	69
Tilting	○	0
Unknown	○	0
N/A	○	

Time since last update:
00:01
seconds.
Previous update took
00:03
seconds.

(d) *Third update after stop.*

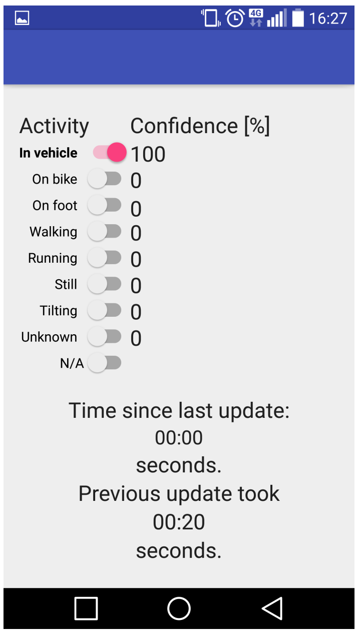


16:26

Activity		Confidence [%]
In vehicle	●	15
On bike	○	0
On foot	○	0
Walking	○	0
Running	○	0
Still	●	46
Tilting	○	0
Unknown	●	38
N/A	○	

Time since last update:
00:01
seconds.
Previous update took
00:02
seconds.

(e) *Bus starts to move again.*



16:27

Activity		Confidence [%]
In vehicle	●	100
On bike	○	0
On foot	○	0
Walking	○	0
Running	○	0
Still	○	0
Tilting	○	0
Unknown	○	0
N/A	○	

Time since last update:
00:00
seconds.
Previous update took
00:20
seconds.

(f) *First update after moving again.*

Figure 14: *Testing of the built-in activity recognition of an Android Smartphone, using a test application developed in this thesis.*

3.7.3 Automatic ICE Notification in Android

To contact an ICE with relevant information about a possible accident, some sort of communication is necessary. Rural areas, such as forests, sometime lack coverage for cellular services, so a minimally demanding transfer protocol is desired. Of those text based communications available for Android programmers, regular text messages require the least from the network. To use the text message functionality permission from the user is needed [46], which is acquired before the application is installed. Therefore is it possible to compose a message containing information such as last known location, time stamp, battery level etc. and send it to the previously chosen ICE contact without any user interaction. Before the application can be used, an ICE must be chosen and the application shouldn't be functional if no ICE exists.

4 Results

Three different algorithms make up the IDA; OC-SVM, ACC and RC, the RC is however dependent on real-time running methods on the smartphone, and can therefore not be evaluated on previously collected data. So, here OC-SVM and ACC are evaluated for different datasets and their results presented. Worth mentioning is that all user smartphone interaction should be handled by sensors not logged during data collection, meaning that these samples will not be handled by either OC-SVM or ACC and therefore occur as false alarms in all datasets.

4.1 The Incident Detection Algorithm

Tests with the OC-SVM proved it to be insufficient as an IDA on its own, what it uses as input can be found in Section 3.2.4. However, no accidents were missed (which is very desirable) so if the extra predictions, those that are false positives, could be cancelled based on additional sensor information, an accurate IDA would have been achieved. Thus, following a positive prediction from the OC-SVM, all Accident Confirmation Criteria (*ACC*) must be fulfilled within a limited time, see Figure 15 for an overview of how the IDA works. This proved very successful and, during the test cases, all false positives were canceled; however, two scenarios existed that still caused problems. The first was the short periods of time where the user interacted with their phone, and the second was the act of mounting/dismounting an ATV. These motion patterns have the potential to both trigger the OC-SVM and fulfill all

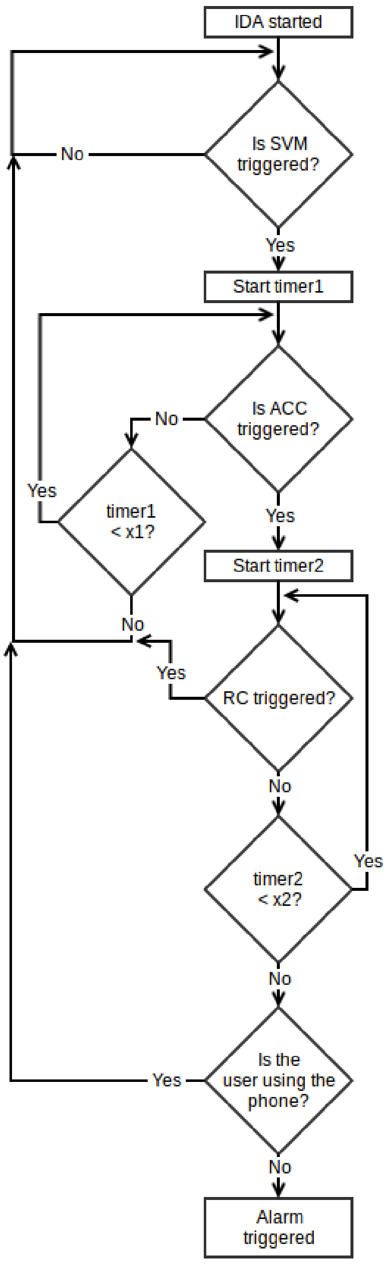


Figure 15: *Flow chart of the IDA.*

ACC. Both mounting and initial smartphone interaction could be handled with a delayed start of the IDA, based on the assumption that the user starts the application while mounted on, or shortly before mounting, an ATV. In order to handle dismounting, the IDA checks for Rejection Criteria (RC). The RC are based on the built-in activity classification systems in Android and iOS. If a normal activity, such as walking, is identified with a high level of confidence, this is a good indicator that an accident hasn't occurred. If none of the RC are fulfilled within a specific timeframe, it is likely that an accident has occurred; however, as mentioned, abnormal data can be generated when the user interacts with the smartphone. Therefore, one last check is done to rule out this false alarm possibility by checking whether the system detects user interaction, such as the screen being lit and the proximity sensor detecting that the smartphone is not near anything (such as the inside of a pocket), before proceeding. If user interaction is not detected, the anomaly is classified as an accident, see Figure 15.

4.2 IDA Performance During Walking

When the driver dismounts an ATV, the application should be manually stopped by the driver. However it seems likely that it sometimes will be left on, by purpose or accident, and in that case no alarm should be sent. If the application constantly generates false alarms due to walking, the driver will most likely stop using the application completely. So in order to be able to release a viable application, walking shouldn't generate false alarms.

In Figure 16 it can be seen how OC-SVM handles walking data, both in the beginning and end there are some concentrations of alarms as well as some around sample $0.5 \cdot 10^4$ and one almost at $2 \cdot 10^4$. Since no manipulation has been done to the raw file, the concentrations in the beginning and end consist of user smartphone interaction, i.e, when the user starts and stops the logging, respectively. The other occasions are false alarms, and are succeeding the two highest spikes of the absolute acceleration, with the exception of during user smartphone interaction.

For the same dataset the ACC is evaluated, and the result can be seen in Figure 17. Only during user smartphone interaction is anything considered as an accident, meaning that the false positives the OC-SVM generated are cancelled, as is desirable.

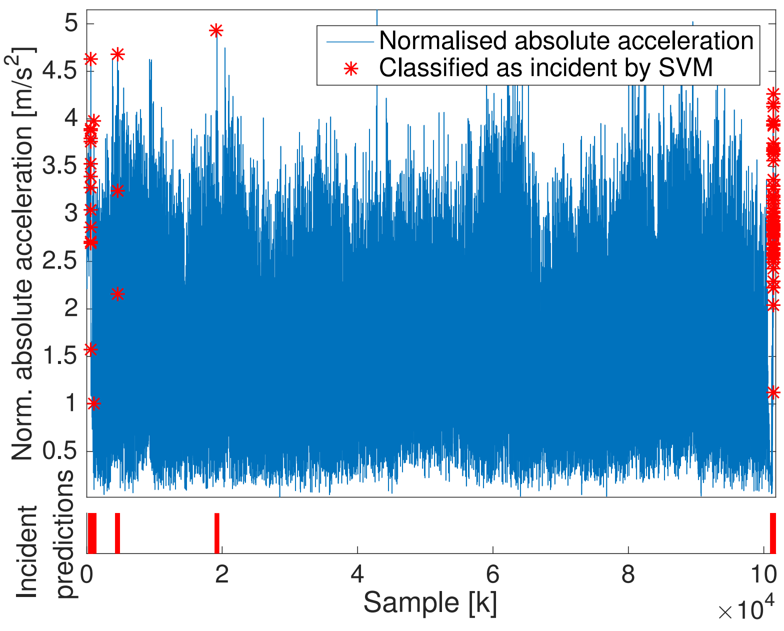


Figure 16: *Samples labeled as accidents by OC-SVM during a walk, false alarms are raised at sample* $0.5 \cdot 10^4$ *and* $2 \cdot 10^4$.

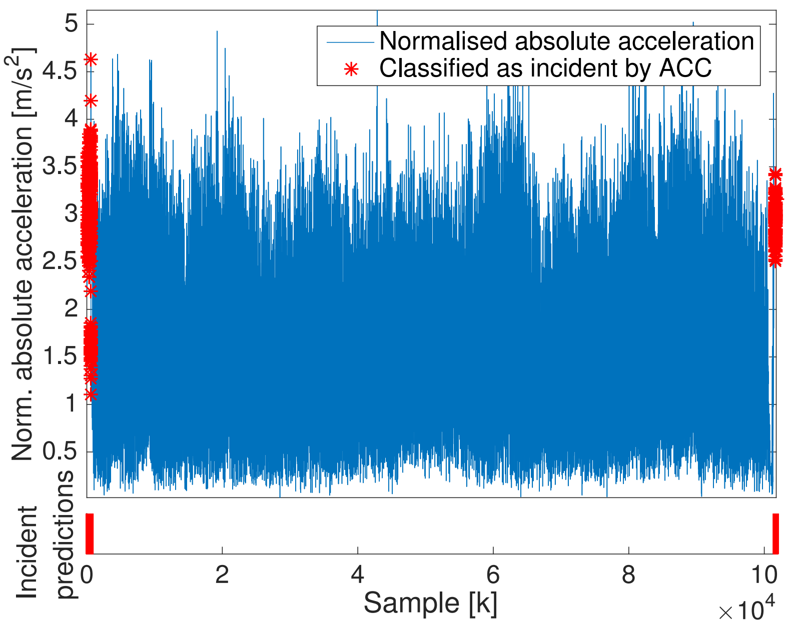


Figure 17: *Samples labeled as accidents by ACC during a walk, no false alarms are confirmed.*

4.3 IDA Performance During Mounting/Dismounting an ATV

Included in the dataset are several mountings and dismounts, see Table 3, all logged by a smartphone placed in a breast pocket. Several false positives are raised by the OC-SVM, as can be seen in Figure 18. However, most of these should be handled by the methods discussed in Section 3.4; for mounting a delayed start of the IDA, and for dismounting the RC is used.

With a temporally altered ACC to accommodate the dataset, while still being representative, false alarms are raised as seen in Figure 19, which further motivates the need for RC.

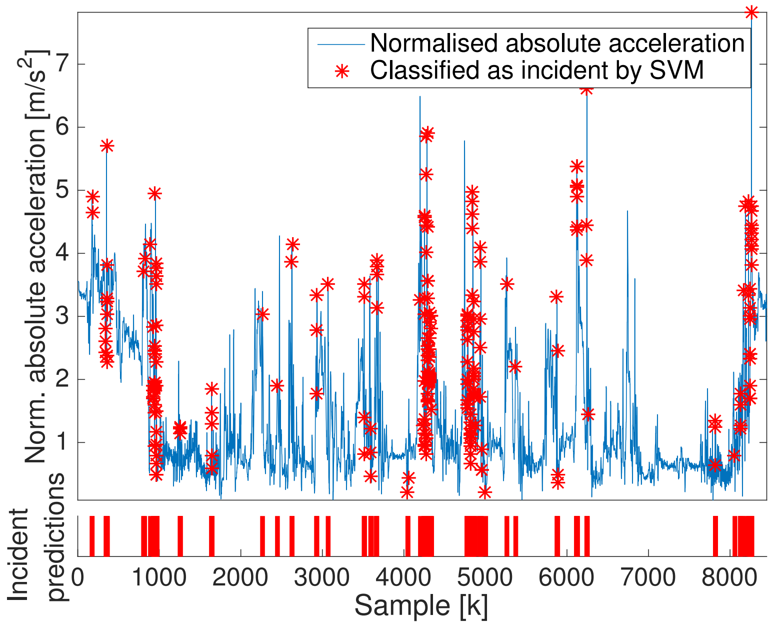


Figure 18: *Samples labeled as accidents by OC-SVM during mounting/dismounting an ATV, several false alarms are raised in this dataset.*

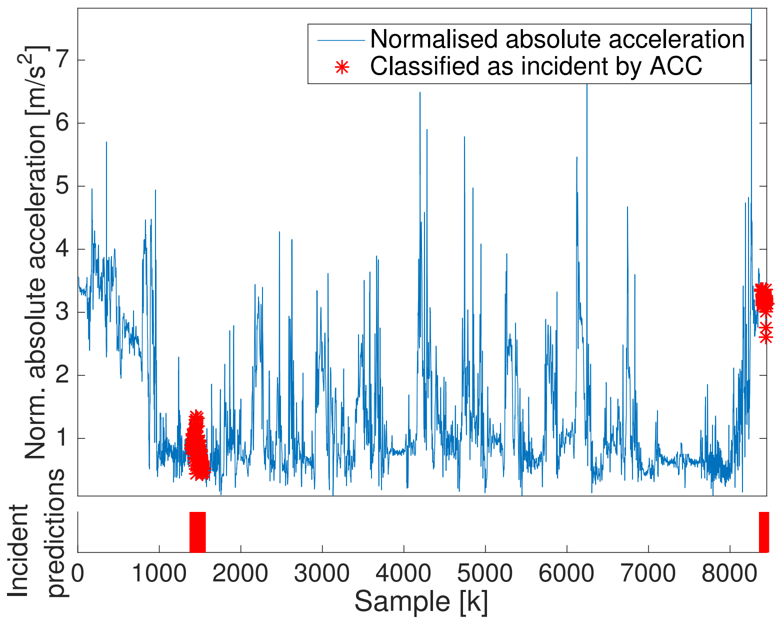


Figure 19: *Samples labeled as accidents by ACC during mounting/dismounting an ATV, from sample 1400 false alarms are raised for a period of ∼241 samples.*

4.4 IDA Performance During Wheelie Accidents

A wheelie accident happens when the ATV lift its forward wheels of the ground and continue this motion until the ATV falls over backwards, having rotated around the rear wheel axis. In Figure 20 six wheeling accidents are logged, which are the peaks located on or about sample 0.5, 0.8, 1.05, 1.34, 1.55 and $1.8 \cdot 10^4$. All of these are identified by the OC-SVM, and some false alarms are raised by the operator standing up, as can be seen just before sample $0.9 \cdot 10^4$. These false alarms are not considered in the performance evaluation of the IDA since it doesn't reflect the proceedings of the IDA, which can be seen in Figure 15. The two concentrations in the beginning and end are user smartphone interactions.

All accidents correctly labeled by the OC-SVM are confirmed by the ACC, see Figure 21, while no false alarms are raised.

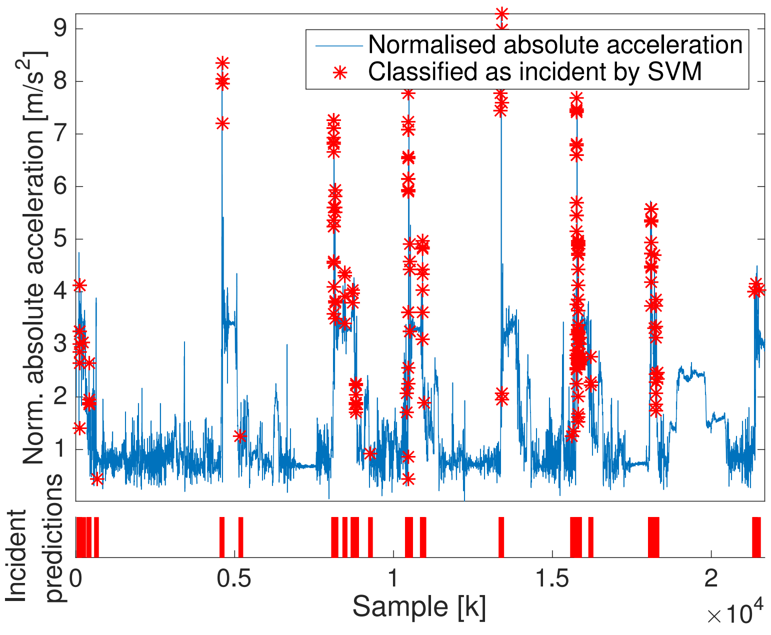


Figure 20: *Samples labeled as accidents by OC-SVM during wheeling simulations, accidents occurring around sample* $0.5,\ 0.8,\ 1.05,\ 1.34,\ 1.55$ *and* $1.8 \cdot 10^4$.

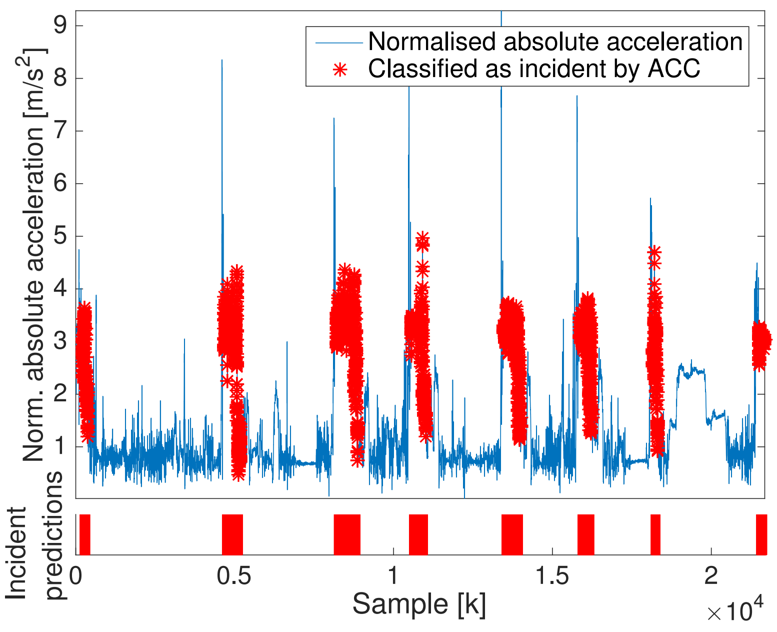


Figure 21: *Samples labeled as accidents by ACC during wheeling simulations, accidents occurring around sample* $0.5,\ 0.8,\ 1.05,\ 1.34,\ 1.55$ *and* $1.8 \cdot 10^4$.

4.5 IDA Performance During Sudden Stops and Roll Over Accidents

To evaluate the IDA for forward crashes (involving hitting objects straight on, such as trees) and flying over the handle bars, rolling over to the side of the ATV (for example if one side suddenly falls into a ditch, throwing the driver off) or rolling over forward, which could happen if the driver turns to sharply and the ATV looses grip. These three types of accidents are present in the dataset presented in Figure 22, further information about the positive predictions are presented in Table 4, there is some user smartphone interaction in the beginning and end of the set as well.

Table 4: *Explanations for positive incident predictions in Figure 22.*

Generated by	**Location** $\cdot 10^4$ [-]	**Type**
Forward crash	0.85, 1.2, 1.5, 1.9, 2.15	Accident
Roll over: side	2.75, 2.9, 3.1, 3.3, 3.45	Accident
Roll over: forward	3.94, 4.56, 4.8, 5.17	Accident
Placing chair	0.6	False alarm
Sitting down fast	1.35, 1.67	False alarm
Leaning over	3.72, 4.44	False alarm

Compared to OC-SVM, ACC performs better, all false alarms presented in Table 4 are canceled while accidents are confirmed, as seen in Figure 23 (worth noting is that locations of the peaks etc. are slightly shifted from the value presented in Table 4 due to the nature of the ACC). Omitting the known problem areas containing user smartphone interaction, there were 42 instances of false alarms over all crash scenarios; however, 13 of these were generated by the operator initiating a fall, but stopping suddenly and aborting the fall. Thus, only 29 false alarms were caused by completely irrelevant events.

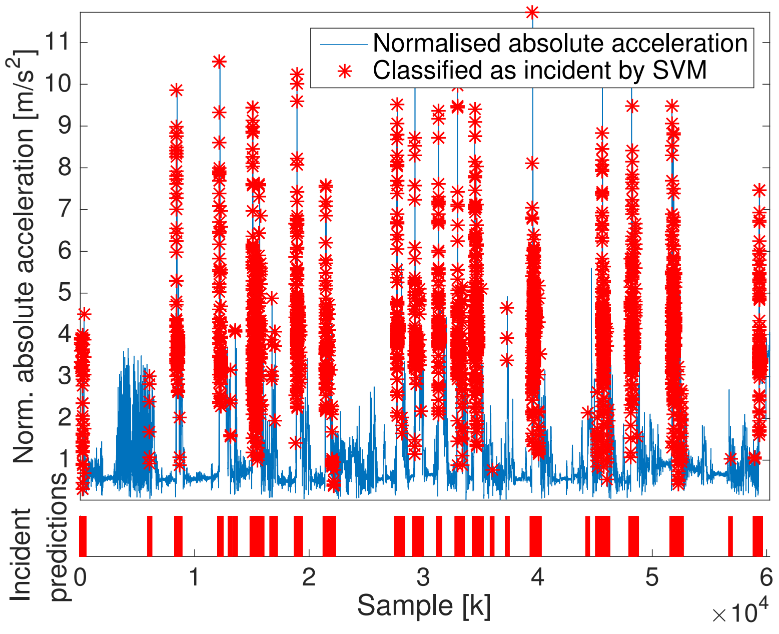


Figure 22: *Samples labeled as accidents by OC-SVM during sudden stops and roll over simulations.*

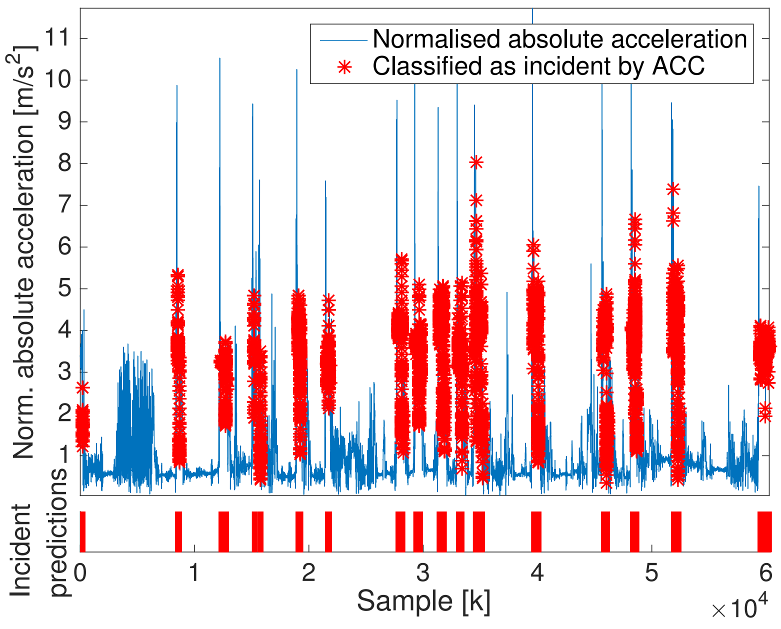


Figure 23: *Samples labeled as accidents by ACC during sudden stops and roll over simulations.*

4.6 IDA Performance During Normal Driving

Although the OC-SVM was trained on a small fraction of the available data (about 1.5 hours), owing to the randomised observation structure described in Section 3.2.4, the resulting precision on the large test set (the remaining ∼20 hours) was high. For the case including all observations in the dataset, shown in the second row of Table 5, 99.29% of observations were classified correctly. However, some of the individual logs produced rather worse results than others; therefore, these were examined more closely.

Table 5: *Results when letting the OC-SVM classify all observations in the dataset.*

Case	**Number of samples examined**	**Number of samples "normal"**	**Precision**
All observations	7807604 (∼ 21 h 41 min 16 s)	7751903 (∼ 21 h 31 min 59 s)	99.29 %
Barring abnormal logs	7569697 (∼ 21 h 1 min 37 s)	7523585 (∼ 20 h 53 min 56 s)	99.39 %

4.6.1 Inspecting Datasets with the Highest Levels of False Alarms

The two individual logs producing the most false alarms had specificities (true negative rate) of 95.90% and 96.71%, respectively, and turned out to both be from the same driver. Examining plots of their absolute accelerations revealed erratic values and possible lapses in sampling which proved difficult for both the OC-SVM to classify correctly, see Figure 24, as well as having false alarms cancelled by the ACC, see Figure 25. Possible causes of these erratic patterns are data corruption, the driver forgetting the logging application is running and doing some other activity, driving an ATV with the smartphone in a lose pocket/bag so that the smartphone moves around more independently than if it were confined in a tighter pocket.

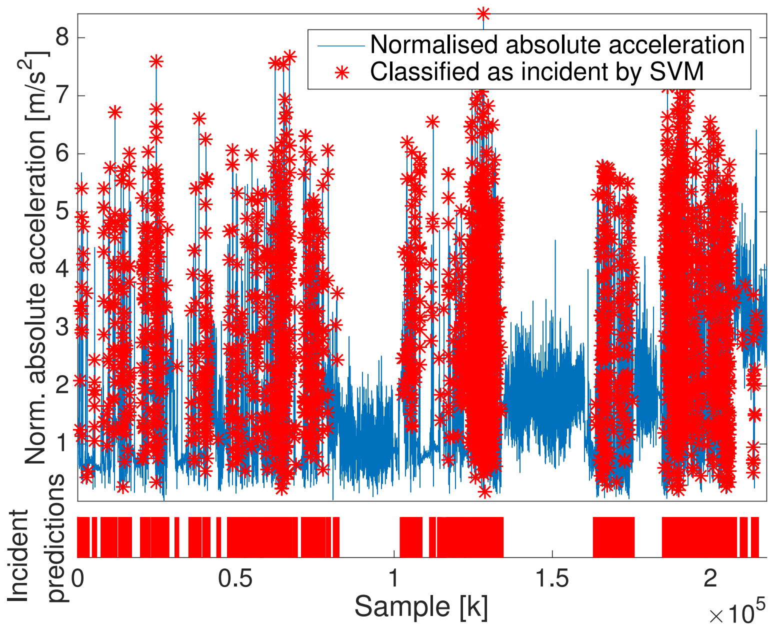


Figure 24: *Samples labeled as accidents by the OC-SVM in the worst observed individual log in the original dataset.*

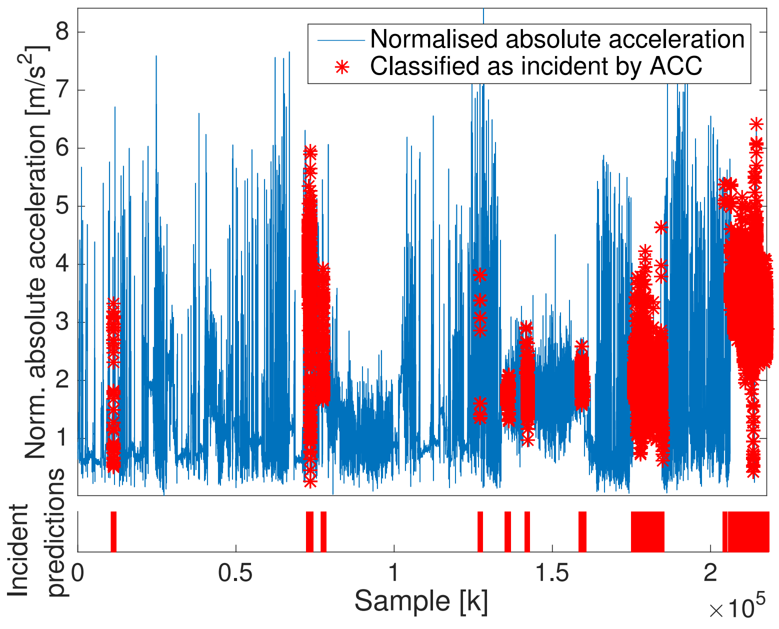


Figure 25: *Samples labeled as accidents by ACC in the worst observed individual log in the original dataset.*

Since it is impossible to discern why the two logs included such erratic data, and because they did not make up a significant part of the overall test set, these were omitted to obtain the precision in the last row of Table 5. For comparison, the log with the worst results, barring the two aforementioned, was plotted and examined. The OC-SVM had performed on this log with a 97.47% precision and had produced false alarms in a few sampling instances, as visible in Figure 26; however, as shown in Figure 27, the ACC produced no alarms and thus all false alarms produced by the OC-SVM would have been cancelled in this case.

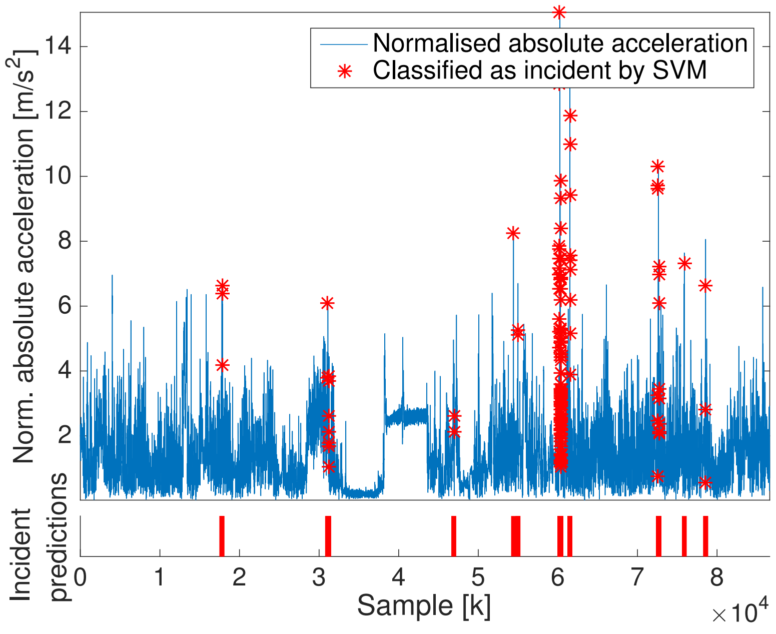


Figure 26: *Samples labeled as accidents by the OC-SVM in the worst observed individual log in the modified dataset.*

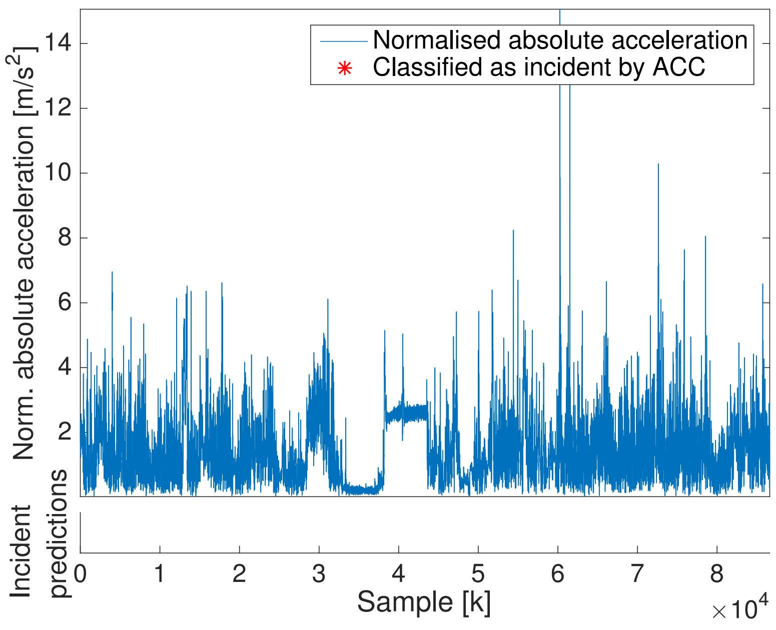


Figure 27: *Samples labeled as accidents by ACC in the worst observed individual log in the modified dataset.*

4.6.2 Estimated F_1-Score for the OC-SVM During Normal Driving

Barring the two erratic cases described in Section 4.6.1, and not taking into account post-processing by the ACC, the OC-SVM obtains an estimated F_1 score of 99.69 %. Specifically, this was calculated as shown in Equation (2), where *precision* is the fraction of normal driving samples correctly classified (i.e 99.39 %), and *recall* is the fraction of simulated accidents that were discovered by the OC-SVM (i.e 100 %).

$$F_1 = 2 \cdot \frac{(precision) \cdot (recall)}{(precision) + (recall)} = 2 \cdot \frac{(0.9939) \cdot (1.00)}{0.9939 + 1.00} \approx 0.9969 \qquad (2)$$

4.7 IDA Performance Summarised

All final alarms from the IDA assume that the alarm is not cancelled by RC after the IDA has classified the event as an incident. RC is dependent on real-time built-in methods in the smartphone and can't be fairly evaluated with collected data; furthermore, user smartphone interaction related alarms are not included as these can be handled using other methods, see Section 3.5.

The number of alarms generated by the OC-SVM and ACC, as well as the calculated number of alarms from the IDA which would have been sent, are presented in Table 6. Alarms sent by the IDA rely on the prevalence of ACC alarms in conjunction with OC-SVM alarms (as well as no *Rejection Criteria*-confirmations) and, as such, each period of continuous alarms is viewed as one incident case. For the OC-SVM and ACC, results represent the total number of sampling instances where each respective classifier classified that an accident had occurred, barring areas of user smartphone interaction. As a reference: a one-second period where all samples are classified as accidents will approximately equal 100 samples (due to the sampling frequency).

Note that the number of final alarms raised by the IDA to the smartphone application was not calculated for the worst observed driving data – these numbers are only included for the interest of the reader.

Table 6: *Summarised performance of OC-SVM, ACC and IDA. The number of final alarms was not calculated for the worst observed driving data nor areas of user smartphone interaction but it is clear that both would be large. Note that these results assume no cancellation from RC.*

Dataset	**OC-SVM**		**ACC**		**IDA**	
	Alarms raised	Whereof false	Alarms raised	Whereof false	Alarms raised	Whereof false
Walking data	3	3	0	0	0	0
Mounting & Dismounting	75	75	141	141	1	1
Crashes	1515	42	4962	0	14	0
Wheelie accidents	123	3	5341	0	6	0
Worst observed driving in original dataset*	3763	3763	5568	5568	-	-
Worst observed driving in modified dataset	47	47	0	0	0	0

5 Discussion

Several aspects of this report needs further discussion, which can be found here as well as some topics not mentioned in any section earlier, but that significantly affects this report's subject.

5.1 The Three Classification Methods Used in the IDA

Initially only an SVM was going to be used to determine if an incident had occurred, it proved insufficient and additions were made.

First, the ACC were added, and immensely increased the performance of the whole IDA. However the inner workings of the ACC isn't suitable to run continuously, since false alarms would be raised; if it had been, the OC-SVM would've been scrapped since it is outperformed by the ACC. So a hybrid was created, the OC-SVM does a first check of the sensors' values and if it finds an anomaly the ACC is brought in to further analyse it. This combination works great for most cases, but proved to be insufficient in handling mounting/dismounting. No great method to handle those motions has been developed, and are currently handled by the RC, if walking is detected shortly after both OC-SVM and ACC has triggered an alarm the alarm is canceled. Mounting is currently handled by a simple delayed start of the IDA.

5.2 Cellular and Data Coverage in Rural Areas

As anyone who owns a smartphone knows, coverage varies with a lot of things. In an urban environment being in a building will lower the reception, and with thicker walls the coverage decreases even more. In a rural area, the topography of the land plays a significant role in how good coverage is available. A risk exists with varying coverage, if an accident occurs in a white spot (an area with no coverage) there is no way to communicate either location or the fact that an accident has occurred.

This problem can be handled by continuous updates to a server, for example an update every five minutes from the smartphone to the server containing current position and speed. If a certain amount of updates are missed, the server can raise an alarm and provide a *last-known-location.*

5.3 Notification of Emergency Services in Case of Accident via Text Message

It is not uncommon that text messages can be received unnoticed, if the user and smartphone is in separate rooms for example. People has a tendency to consciously ignore private and/or turn off notifications while at work, meaning an accident notification might not be discovered for several hours, losing valuable time for the hurt driver. SOS Alarm do have the technology to handle text messages, and do so for a small part of the Swedish population (see Section 3.3) but is restricted in allowing more people the same functionality. It is understandable that a group not capable of using a phone by conventional standards (i.e. people with hearing/talking disabilities) has priority over an automatic text message from an unofficial source, but other situation exists where emergency help would preferably be communicated via text messages. A common example is threatening situations, such as home invasions, being able to silently text emergency services instead of calling them seems as a reasonable technological advancement. Although decreasing the human contact between subject and operator has some disadvantages, it's harder to gather a correct picture of the situation without hearing how the subject speaks and prank texts will undoubtedly occur. Emergency applications, such as the one proposed in this report, would also most likely require a lot of resources. Even though many arguments can be made for why limitation is needed for this functionality, development should be made in order to further spread availability; one step in this process is *eCall*, which requires cars to automatically contact emergency services if an accident has occurred [47].

5.4 IDA Performance

Four main areas has been investigated for the IDA, most of which performed satisfactory at least, further discussion of the result follows here. What proved to be hardest was not to correctly classify normal driving and simulated accidents, but rather making the IDA more consumer friendly by adapting it to situations likely to occur.

5.4.1 Walking

An IDA that would be triggered by walking would cause great annoyance for the user, since several scenarios can be imagined that require the driver to dismount the ATV for a short period of time, where temporarily turning of the IDA shouldn't be necessary. As presented in Section 4.2, thanks to ACC no alarms are raised, together with the RC the IDA would be paused. Just using one walk from one person might seem as a small set to evaluate over, and yes it is. This report's focus however isn't the ability to distinguish walking from driving, and is only done to prove that it is viable for a consumer version of the IDA, where the user most likely doesn't want to take out his or her smartphone every time they dismount an ATV. Therefore this simple evaluation is included since the initial goal was to create such an application, but before an actual release further evaluation must be conducted.

5.4.2 Mounting/Dismounting an ATV

Before a commercialised application could be released, improvements must be done with handling of mounting/dismounting. Some major simplifications has been made, such as that the user will only turn on the IDA shortly before mounting/driving away the ATV, for example, if they would turn it on, walk around looking for a helmet high and low in the house, an alarm could be triggered. As with dismounting, if the driver doesn't walk around for 30 seconds continuously, an Android version would most likely raise an alarm (see Section 3.7.2 for explanation). The smartphone application proposal presented in Figure 11 should be able to handle the driver dismounting the ATV, walk around for a bit and then mounting the ATV again, if the RC is triggered (i.e. the alarm aborted), otherwise if an alarm is raised and confirmed by the ACC and the driver has time to mount the ATV before RC cancels it, it would proceed to notify the driver unnecessarily. For a better performing application, the motion of mounting/dismounting would ideally be identified, so that the driver state could be monitored and the IDA automatically paused/started.

5.4.3 Simulated Accidents

With the combination of OC-SVM and ACC all collected accidents were correctly classified, and no false alarms raised. However, it is possible that overfitting might have occurred, and to further evaluate the IDA more data must be collected, especially crashes. While each type of accident were simulated several times, alternations varied little, mostly due to insufficient protection for the operator and limitation in time and equipment. Due to the limitations, the simulated accident are deemed to be in the lower spectrum of severity, meaning that more realistic accidents would probably be even more distinct and easier to identify. To get even more variance in a future data collection, several test persons should be used in crash simulations since individuals might handle falling off an ATV differently.

5.4.4 Normal Driving

With the OC-SVM used, the majority of all samples of normal driving were correctly classified; however, it is possible that the training set failed to capture a certain type of driving style prevalent in the two "worst case" logs. This could be remedied by allowing the training set to take more observations into account or by increasing generalisation parameters during training as well as during run-time on the user's smartphone.

As an additional step in the evaluation of the IDA, use of the ACC should be included in the OC-SVM results to obtain information on how many of the false alarms could have been canceled. For example, in the case with 97.49% accuracy detailed in Section 4.6.1, all false alarms would have been cancelled by the ACC, giving the IDA in this case an accuracy of 100%. Furthermore, it is impossible to discern whether an alarm would have actually been sent, since the RC are only applicable to run-time conditions and not possible to replicate in MATLAB using the dataset available for the project.

5.5 Ethical & Privacy Considerations

Many applications already use motion information from accelerometers, gyroscopes and magnetometers and the success of activity tracking applications such as *Runkeeper* indicate positive sentiment towards the use of these sensors. However, use of other sensors and data could be viewed as excessively intrusive and, overall, access to sensors and information not obviously required for the primary use of an application may lead to user concerns about privacy. For example, when the *Facebook Messenger* application required that extensive permissions be granted before installing, this sparked an irate article in the Huffington Post about the need to set a stop to excessive requirements by application developers [48]. Although there seemed to be reasons behind each permission requirement, these were not clearly explained to users at install time. For the purposes of improving the IDA, it could be valuable to e.g use the microphone to listen for the starting, stopping or revving of a combustion engine; however, this could make users hesitant about using the application at all. The collection of the application of sensitive information such as health data could also be problematic, although the more obvious connection between a safety application and possessing information about e.g blood type and age may assuage user privacy concerns in this matter.

5.6 Self-fulfilling Prophecies & Psychological Considerations

As a consequence of the IDA being trained and tested on a finite number of ATV rides, and, although SVM's generalise quite well, a situation may arise during normal driving which, for some reason, does not classify as normal. If the subsequent checks for confirmation and rejection criteria also indicate that there has been an accident, this will lead to the application notifying the user that an alarm is about to be sent. Given that the user notices this, he or she may quickly grow anxious that his or her ICE will receive frightening news, it is conceivable that concentration on driving will waver; thus, possibly leading to an accident where none would have occurred.

An entirely different concern is the possibility that having a safety application running will produce a false sense of security or invulnerability with the user and thus promote reckless driving. Although accidents occurring as a result of this may trigger alarms and, thus, the relatively swift arrival of medical professionals, the fact of the matter is that the accident might never have happened without the application.

6 Conclusion

The preliminary aims were:

1. to create an IDA which can identify accidents and crashes.
2. Optimise the IDA in order to obtain a high F_1-score and
3. release it as an application through official distribution channels, such as the Google and Apple online application stores.

Regarding the first aim, all parts of the IDA has been constructed and evaluated with good results, but no merged version exists since that only complicates things while evaluating parameters. Also, the F_1-score is high but based on simulated crash data.

For the third aim, work has been done to evaluate the possibility of running this type of computation in real time on smartphones, which has only generated positive results. So, to actually release an application a lot of pieces has to be put together, but all of the pieces are there and have been checked that they work for each platform chosen.

Also, no other hardware other than that available in a smartphone has been used for any purpose in the IDA or the supporting structure around it.

Even though the goals chosen at the start of the project hasn't been fully completed, the major objectives has. A working IDA exists and functions well, it is fully implementable on both Apple's iOS and Google's Android and would only use onboard sensors and built in functions to evaluate data.

7 Future Development Suggestions

During the course of the study, several interesting ideas and theories came up. However, not all of them were possible to realise or even test. Some of the more interesting ideas are shortly discussed here.

Evaluation of Close Call Situations

A dataset not actively collected and evaluated is close call situations. One was accidentally logged (when an operator begins to throw himself out of a chair, but aborts the simulation and subsequently remains in the chair) which resulted in 13 false alarm predictions (see Section 4.5). It would however be interesting to actively collect data that are almost accidents, but not really. The best way to do this would of course be to drive around on a real ATV in situations deemed problematic, such as rough terrain (driving over logs, through cairns etc.) and also ask regular drivers about situations they consider hazardous and relatively frequently occurring. If such a set could be collected, better evaluation of the IDA performance would be possible.

Implement Concussion Test

In order to handle drivers that think they are fine after an accident (and thus manually cancel the alarm), but actually quickly worsen shortly after it, the implementation of a concussion test has been discussed. The general idea is that a short questionnaire appears after a canceled alarm, see Figure 28. A possible model for the concussion test could be that used by many athletes [49], however, this test requires a reference test to be able to indicate whether a concussion might have happened or not.

Another possibility is also to monitor that the driver actually is conscious during a period after the canceled alarm, by monitoring for example changes in GPS location, accumulated magnitude for accelerometer and/or gyroscope or just regular speed.

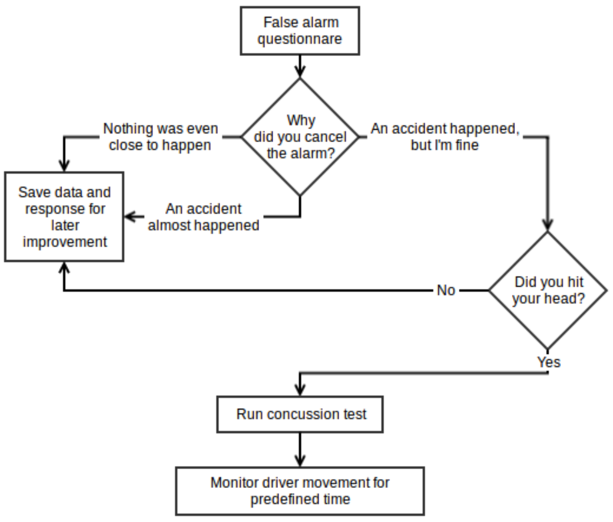

False alarm questionnare
Why did you cancel the alarm?
Nothing was even close to happen
An accident happened, but I'm fine
Save data and response for later improvement
An accident almost happened
Did you hit your head?
No
Yes
Run concussion test
Monitor driver movement for predefined time

Figure 28: *Questionnaire after a canceled alarm.*

Implementation of eCall Protocol

A coming future development in automatic emergency safety is something called *eCall*, which is a standard for cars to communicate with emergency services if an accident has occurred (such as an airbag deployment) [47]. By April 2018 all new cars sold within the European union must have this functionality, however, it seems unlikely that any other operator will be allowed to use this infrastructure in the near future, especially smartphone applications. Without some sort of licensing/cooperation an open access for smartphone application developers to this service would most likely result in a high percentage of false alarms. If, however, in the future the service is made available, it is highly attractive to implement it.

Automatic Detection of Mounting/Dismounting in Order to Ensure Running of Algorithm

To handle the state of the driver (i.e. sitting on the ATV or not) optimally, ideally each mounting/dismounting should be detected. If so, further improvement could be done such as letting the application (but not the IDA) run in the background continuously, until a mounting motion is detected and then start the IDA. This would also improve the handling of false alarms from dismounted drivers (as discussed in Section 5.4).

Accessing Medical Information Stored on an iOS Device

Using a framework called *HealthKit*, which is available on Apple iOS devices, it is possible for users to store their health information on the device for use by authorised applications [50]. Emergency health information such as age, blood type, illnesses and medication is viewable from the lock screen of the device so that a passerby can better aid an unconscious user. It could be useful to include such information in an ICE-contact message so that he or she may relay accurate and vital information to emergency services. Naturally, this possibility would be governed by the prevalence of users who actually fill out their emergency health information page.

References

[1] Folksam, “Bicycle helmet test 2015,” *www.folksam.se*, 2015, [Accessed: 19 May, 2016]. [Online]. Available: http://www.folksam.se/media/folksam-bicycle-helmet-test-2015_tcm5-24933.pdf

[2] The Specialty Vehicle Institute of America. *”What is an ATV”*. www.svia.org. [Accessed: 26 June, 2016]. [Online]. Available: http://www.svia.org/#/aboutATV

[3] Trafikanalys, “Road traffic injuries 2015,” *www.trafa.se*, 2015, [Accessed: 19 May, 2016]. [Online]. Available: http://www.trafa.se/globalassets/statistik/vagtrafik/vagtrafikskador/vaegtrafikskador_2015.pdf

[4] The Swedish Transport Administration, *Better Safety on Quad Bikes. Joint strategy version 1.0 for the years 2014-2020.* The Swedish Transport Administration, 2013.

[5] Statistiska Centralbyrån / Statistics Sweden, “Privatpersoners användning av datorer och internet 2015 [use of computers and the internet by private persons in 2015],” *www.scb.se*, 2015, (in Swedish). [Accessed: 20 January, 2016]. [Online]. Available: http://www.scb.se/Statistik/_Publikationer/LE0108_2015A01_BR_00_IT01BR1501.pdf

[6] S. Candefjord, L. Sandsjö, R. Andersson, N. Carlborg, A. Szakal, J. Westlund, and B. A. Sjöqvist, “Using smartphones to monitor cycling and automatically detect accidents - towards ecall functionality for cyclists,” In: Proceedings, International Cycling Safety Conference 2014, 18–19 November 2014, Gothenburg, Sweden; 2014:1-9. [Online]. Available: hhttp://bada.hb.se/handle/2320/14570

[7] L. Bergbom, C. Engelbrektsson, S. Granberg, and L. Streling, “Säkerhetsapp för ryttare [Safety app for horse riders],” Bachelore Thesis, Chalmers Univeristy of Technology, Gothenburg, Sweden, 2015, (in Swedish). [Online]. Available: http://publications.lib.chalmers.se/records/fulltext/219261/219261.pdf

[8] S. Garland, “National estimates of victim, driver, and incident characteristics for atv-related, emergency department-treated injuries in the united states from january 2010 – august 2010 with an analysis of victim, driver and incident characteristics for atv-related fatalities from 2005 through 2007,” Directorate for Epidemiology, Division of Hazard Analysis, U.S Consumer Product Safety Commission, 2014. [Online]. Available:

https://www.cpsc.gov/Global/Research-and-Statistics/Injury-Statistics/Sports-and-Recreation/ATVs/ATVSpecialStudyReport.pdf

[9] Everyday Mysteries. *"What is a GPS? How does it work?"*. www.loc.gov. [Accessed: 26 June, 2016]. [Online]. Available: http://www.loc.gov/rr/scitech/mysteries/global.html

[10] Federal Communications Commission. *"Understanding Wireless Telephone Coverage Areas"*. www.fcc.gov. [Accessed: 26 June, 2016]. [Online]. Available: https://www.fcc.gov/consumers/guides/understanding-wireless-telephone-coverage-areas

[11] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. Reyes-Ortiz, "Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine," *Ambient Assisted Living and Home Care*, vol. 4, 2012.

[12] M. J. Abady, L. Luceri, M. Hassan, T. Chou, C, and M. Nicoli, "A collaborative approach to heading estimation for smartphone-based PDR indoor localisation," *2014 International Conference on Indoor Positioning and Indoor Navigation*, 2014.

[13] S. Mazilu, M. Hardegger, Z. Zhu, D. Roggen, G. Tröster, M. Plotnik, and J. Hausdorff, "Online Detection of Freezing of Gait with Smartphones and Machine Learning Techniques," *International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, vol. 6, 2012.

[14] F. Sposaro and G. Tyson, "iFall: An android application for fall monitoring and response," *Engineering in Medicine and Biology Society*, vol. 6, 2009.

[15] J. Hu, D. Li, Q. Duan, Y. Han, G. Chen, and X. Si, "Fish species classification by color, texture and multi-class support vector machine using computer vision," *Computers and Electronics in Agriculture*, vol. 88, 2012.

[16] D. M. J. Tax, "One-Class Classification," *ASCI dissertation series*, vol. 65, 2001.

[17] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support Vector Method for Novelty Detection," *NIPS*, vol. 12, pp. 582–588, 1999.

[18] Azure Machine Learning Team. *"Microsoft Azure Machine Learning: Algorithm Cheat Sheet"*. http://azure.microsoft.com. [Accessed: 20 March, 2016]. [Online]. Available: http://aka.ms/MLCheatSheet

[19] R. Caruana and A. Niculescu-Mizil, “An Empirical Comparison of Supervised Learning Algorithms,” Cornell University, Ithaca, USA, Tech. Rep., 2006.

[20] S. Omar, A. Ngadi, and H. H. Jebur, “Machine Learning Techniques for Anomaly Detection: An Overview,” Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia, Tech. Rep., 2013.

[21] F. Yuan and R. L. Cheu, “Incident detection using support vector machines,” *Transportation Research Part C: Emerging Technologies*, vol. 11, no. 3–4, pp. 309 – 328, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X03000202

[22] R. A. Fisher, “The Use of Multiple Measurements in Taxonomic Problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[23] I. Steinwart and A. Christmann, *Support Vector Machines*, 1st ed. Springer, 2008.

[24] D. Tax and R. Duin, “Support Vector Data Description,” *Machine Learning*, vol. 54, pp. 45–66, 2004.

[25] Jalp Systems. *”Jalp – Säkerhetslösningar för oskyddade trafikanter [Jalp – Safety solutions for vulnerable road users]”*. www.jalp.se. (in Swedish). [Accessed: 28 June, 2016]. [Online]. Available: http://www.jalp.se

[26] S. Candefjord, L. Sandsjö, and B. A. Sjöqvist, “Statistikinsamling och automatiskt olyckslarm för trafik med fyrhjulingar via en smartmobilplattform [Statistical collection and automatic incident alarm for traffic by quadwheeler via a smartphone platform],” SAFER, Chalmers University of Technology and University of Borås, Tech. Rep., February 2016, (in Swedish).

[27] Apple Inc. *”Numbers for Mac”*. www.apple.com. [Accessed: 29 June, 2016]. [Online]. Available: http://www.apple.com/mac/numbers/

[28] D. G. Lister, J. Carl, J. H. Morgan, M. Denning, David A amd Valentovic, B. Trent, and B. L. Beaver, “Pediatric all-terrain vehicle trauma: a 5-year state-wide experience,” *Journal of Pediatric Surgery*, vol. 33, no. 7, pp. 1081–1083, 1998.

[29] MedlinePlus. *”Concussion”*. U.S. National Library of Medicine. www.nlm.nih.gov. [Accessed: 8 Mars, 2016]. [Online]. Available: https://www.nlm.nih.gov/medlineplus/concussion.html

[30] Thompsons Solicitors. *"Compensation Claims for Fractured Bones / Broken Bones"*. www.thompsons.law.co.uk. [Accessed: 27 June, 2016]. [Online]. Available: http://www.thompsons.law.co.uk/other-accidents/compensation-claim-fractured-bone-broken-bone.htm

[31] IDC Research Inc. *"Smartphone OS Market Share, 2015 Q2"*. www.idc.com. [Accessed: 23 May, 2016]. [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[32] Apple Inc., *The Swift Programming Language*, 2nd ed. Apple Inc., 2015.

[33] ——. *"Measure Energy Impact with Xcode"*. developer.apple.com. [Accessed: 3 May, 2016]. [Online]. Available: https://developer.apple.com/library/watchos/documentation/Performance/Conceptual/EnergyGuide-iOS/MonitorEnergyWithXcode.html#//apple_ref/doc/uid/TP40015243-CH34-SW1

[34] ——. *"vDSP Programming Guide"*. developer.apple.com. [Accessed: 25 February, 2016]. [Online]. Available: https://developer.apple.com/library/prerelease/ios/documentation/Performance/Conceptual/vDSP_Programming_Guide/About_vDSP/About_vDSP.html#//apple_ref/doc/uid/TP40005147-CH2-SW1

[35] ——. *"Accelerate Framework Reference"*. developer.apple.com. [Accessed: 26 February, 2016]. [Online]. Available: https://developer.apple.com/library/prerelease/ios/documentation/Accelerate/Reference/AccelerateFWRef

[36] ——. *"CMMotionActivity Class Reference"*. developer.apple.com. [Accessed: 25 February, 2016]. [Online]. Available: https://developer.apple.com/library/ios/documentation/CoreMotion/Reference/CMMotionActivity_class/

[37] Android Developers. *"ActivityRecognitionResult"*. developers.google.com. [Accessed: 27 June, 2016]. [Online]. Available: https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionResult#public-methods

[38] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A Practical Guide to Support Vector Classification," National Taiwan University, Dept. of Computer Science, Taipei 106, Taiwan, Tech. Rep., 2010.

[39] SOS Alarm, "SMS 112 in Sweden," *www.sosalarm.se*, 2010, [Accessed: 19 May, 2016]. [Online]. Available: https://www.sosalarm.se/PageFiles/1155/SMS%20112%20Systembeskrivning_EN%20_2_.pdf

[40] Apple Inc. *"CMMotionActivityManager Class Reference"*. developer.apple.com. [Accessed: 3 May, 2016]. [Online]. Available: https://developer.apple.com/library/ios/documentation/CoreMotion/Reference/CMMotionActivityManager_class/index.html#//apple_ref/occ/instm/CMMotionActivityManager/queryActivityStartingFromDate:toDate:toQueue:withHandler:

[41] ——. *"MFMessageComposeViewController Class Reference"*. developer.apple.com. [Accessed: 27 May, 2016]. [Online]. Available: https://developer.apple.com/library/ios/documentation/MessageUI/Reference/MFMessageComposeViewController_class/index.html

[42] ——. *"CFNetwork Programming Guide"*. developer.apple.com. [Accessed: 30 May, 2016]. [Online]. Available: https://developer.apple.com/library/mac/documentation/Networking/Conceptual/CFNetwork/Introduction/Introduction.html#//apple_ref/doc/uid/TP30001132-CH1-DontLinkElementID_30

[43] Google Inc. *"Android usage statistics"*. developer.android.com. [Accessed: 24 Mars, 2016]. [Online]. Available: http://developer.android.com/about/dashboards/index.html

[44] X. Ducrohet. "Android 4.0.3 Platform and Updated SDK tools". 11 December 2011 [Blog entry]. *Android Developers Blog*. [Accessed: 24 May, 2016]. [Online]. Available: http://android-developers.blogspot.se/2011/12/android-403-platform-and-updated-sdk.html

[45] B. Rakowski. "Get ready for the sweet taste of Android 6.0 Marshmallow". 5 October 2015 [Blog entry]. *Android Official Blog*. [Accessed: 24 May, 2016]. [Online]. Available: http://officialandroid.blogspot.se/2015/10/get-ready-for-sweet-taste-of-android-60.html

[46] Android Developers. *"SmsManager in Android documentation"*. developer.android.com. [Accessed: 27 May, 2016]. [Online]. Available: https://developer.android.com/reference/android/telephony/SmsManager.html

[47] European Commission. *"eCall: Time saved = lives saved"*. European Commission. www.ec.europa.eu. [Accessed: 16 February, 2016]. [Online]. Available: https://ec.europa.eu/digital-agenda/en/ecall-time-saved-lives-saved

[48] S. Fiorella. "The Insidiousness of Facebook Messenger's Android Mobile App Permissions (Updated)". 11 August 2014 [Blog entry]. *The Huffington*

Post, The Blog. [Accessed: 26 May, 2016]. [Online]. Available: http://www.huffingtonpost.com/sam-fiorella/the-insidiousness-of-face_b_4365645.html

[49] P. McCrory, K. Johnston, W. Meeuwisse, M. Aubry, R. Cantu, J. Dvorak, T. Graf-Baumann, J. Kelly, M. Lovell, and P. Schamasch, "Summary and agreement statement of the 2nd International Conference on Concussion in Sport, Prague 2004," *British Journal of Sports Medicine*, vol. 39, pp. 196–204, 2005.

[50] Apple Inc. *"HealthKit Framework Reference"*. developer.apple.com. [Accessed: 1 Jun, 2016]. [Online]. Available: https://developer.apple.com/library/ios/documentation/HealthKit/Reference/HealthKit_Framework/