

Development of an Integrated Bicycle Accident Detection System

Introducing ALARM:
Accident Localisation And Recognition Method

TU Delft Bicycle Lab

Joris Kuiper



Development of an Integrated Bicycle Accident Detection System

**Introducing ALARM:
Accident Localisation And Recognition Method**

by

Joris Kuiper

In partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering
at the faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology.
Monday 14th June, 2021

Supervisor:	Dr. ir. A. L. Schwab
Daily supervisor:	J. K. Moore, PhD
Chair exam committee:	Dr. ir. R. Happee
Advisor from Royal Gazelle:	B. Oor
Institution:	Delft University of Technology
Place:	Faculty of Mechanical, Maritime and Materials Engineering, Delft
Project Duration:	November, 2020 - June, 2021

Preface

Imagine yourself cycling home at night over a secluded bicycle path. Suddenly a gust of wind pushes you off the road and you lose control over the bicycle. After a harsh fall on the concrete you try to get up, notice that your knee is severely bruised and that your phone is broken; there is no one to help you. Luckily your bike is equipped with **ALARM**: the **A**ccident **L**ocalisation **A**nd **R**ecognition **M**ethod. The bike has already noticed that it has been in an accident and send your emergency contacts an alert with your last known location; help is on its way.

My MSc project is aimed to develop a method that can be used to implement a reliable accident detection system in bicycles connected to the internet. This project is undertaken at the request of Royal Gazelle, a Dutch bicycle manufacturer which supplements some e-bikes with an Internet of Things (IoT)-module. This module is currently used for theft detection, but the same hardware can also be used for accident detection and geolocation.

I chose an experimental approach for which I collected artificial accident and actual normal cycling data because I believe one should not settle for models when developing a practical solution. Due to safety concerns, I was not able to collect accident data with riders, so the bicycle needed to be pushed into different objects to simulate accidents. Observing normal cycling was of course allowed, so a lot of normal cycling test runs are made. This not only resulted in an extensive database to train the classification algorithms on, but it was also fun to do.

I would like to thank my supervisors for their guidance during this project. I would also like to thank everyone involved in this project at Royal Gazelle for the insight into the practical implementation of such a system. Lastly, I would like to thank everyone who helped me collect data for both normal cycling and the simulated accidents.

*Joris Kuiper
Delft, June 2021*

Abstract

Bicycles connected to the internet present an opportunity for integrated accident detection and geolocation. Such a system can reduce the time it takes for help to arrive by automatically alerting predefined contacts with the location of the accident. I developed a systematic method for the practical implementation of bicycle accident detection in connected bicycles and present the performance of a prototype system. The method uses accelerometer and gyroscopic measurements as well as localization and velocity estimations. Supplementing existing research, a bicycle accident detection system is validated on normal cycling, edge cases, and three types of single bicycle accidents with constraints set by a bicycle manufacturer. Edge cases are movements of a bicycle that occur during regular usage, but can not be described by normal cycling.

This method uses a data-driven approach. For the prototype system, the input signals are collected during 71 different simulated accidents and 54 hours of normal cycling and edge cases. A three-layer detection algorithm determines if an accident has occurred and sends the last known location to a set of predefined contacts. Multiple combinations of thresholds and classification algorithms are compared. This resulted in a prototype system with a K-Nearest Neighbours classifier which detects 75% of accidents. Normal cycling and edge cases are correctly detected 99.997% of the time. From all warnings send, 85.7% are true accidents.

The prototype system proves that the proposed method can be used to integrate reliable accident detection in connected bicycles. Bicycles with such a system automatically inform emergency contacts with a message containing the location of the accident, in a time where every second counts.

Contents

Preface	i
Abstract	ii
Nomenclature	v
1 Introduction	1
1.1 Types of accidents	1
1.1.1 Collisions	2
1.1.2 Skids	2
1.1.3 Falls	2
1.1.4 Crashes	3
1.2 Connected bicycles of Royal Gazelle	4
1.3 Objective	4
2 Methods	6
2.1 Approach	6
2.2 Data Collection	6
2.2.1 Normal Cycling and Edge Cases	7
2.2.2 Simulated Accidents	9
2.2.3 Time shifts	11
2.3 Sample Description and Feature Calculation	12
2.4 Accident Detection Algorithm	13
2.4.1 Performance Metrics	15
2.5 Threshold Based Classifiers	16
2.6 Machine Learning Classifiers	18
2.6.1 Principal Component Analysis	19
2.6.2 Support Vector Machine	20
2.6.3 K-Nearest Neighbours	21
2.6.4 Naive Bayes	21
2.6.5 Decision Tree	21
3 Results	22
3.1 Performance of different classifiers in layer 1	22
3.2 Performance of the system	23
3.3 Sample Time	25
4 Practical Implementation	26
4.1 Minimum Viable Product	26
4.2 Demonstration Prototype	27
5 Discussion	28
5.1 Performance	28
5.2 Shortcomings	29
5.3 Improvements	29
5.4 Implication of Research	30
6 Conclusion	31
References	34
A Protocol for the observational study of normal cycling and edge cases	35

B Test Runs	38
B.1 Cycling	38
B.2 Accidents	42
C Visualisation of various test runs	45
D Briefing Document for Minimum Viable Product	54
E Thresholds	59
E.1 Useful thresholds	59
E.2 Threshold effectiveness	60
E.3 Sample distribution for each feature	61
F Performance of all validated algorithms	73

Nomenclature

This section only summarises abbreviations and symbols that are frequently used in this report.

Abbreviations

Abbreviation	Definition
IMU	Inertial Measurement Unit
GPS	Global Positioning System
PCA	Principal Component Analysis
IoT	Internet of Things
GNSS	Global Navigation Satellite System
SVM	Support Vector Machine
KNN	K-Nearest Neighbours
DT	Decision Tree
NB	Naïve Bayes

Symbols

Symbol	Definition	Unit
v	Velocity	[m/s]
\ddot{x}	Acceleration in lateral direction	[m/s ²]
\ddot{y}	Acceleration in vertical direction	[m/s ²]
\ddot{z}	Acceleration in longitudinal direction	[m/s ²]
a_{tot}	Total Acceleration = $\sqrt{\ddot{x}^2 + \ddot{y}^2 + \ddot{z}^2}$	[m/s ²]
θ	Roll angle	[°]
ϕ	Pitch angle	[°]
$\dot{\theta}$	Angular velocity about the longitudinal (z) axis	[rad/s]
$\dot{\psi}$	Angular velocity about the vertical (y) axis	[rad/s]
$\dot{\phi}$	Angular velocity about the lateral (x) axis	[rad/s]
ω	Total angular velocity = $\sqrt{\dot{\theta}^2 + \dot{\psi}^2 + \dot{\phi}^2}$	[rad/s]

Coordinate Definitions

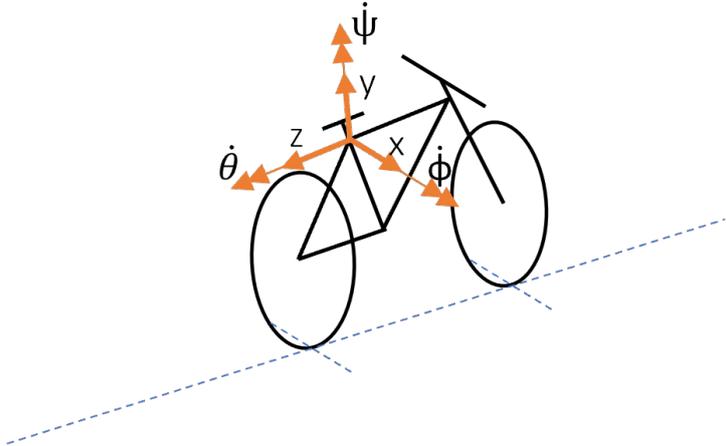


Figure 1: Rear frame fixed coordinate system of an IMU sensor mounted to the seat post of a bicycle.

Introduction

E-bikes are gaining popularity and while they are not necessarily more dangerous than conventional bicycles, they provide the ability for relatively vulnerable people to cycle more [43, 15, 42]. This increases the number of cyclist injuries as the users are exposed to more dangerous scenarios [33]. Of the cyclists involved in an accident without another road user, 63% are alone during the accident [28] and only 36% are able to ride home [7]. An integrated bicycle accident detection system can decrease the time between injury and treatment.

E-bikes that are connected to the internet can be supplemented with new features like accident detection with a firmware update, utilizing the existing digital infrastructure. An accident detection system automatically recognizes accidents and can send the location to emergency contacts.

There are already several products available that offer accident detection and post-crash emergency assistance. For example, the eCall system, which is mandatory for all new passenger cars in the EU from 31 March 2018 [29]. This system is triggered either manually or by the airbags and sets up a voice connection with the nearest emergency centre. Such a 'mayday' system aims to reduce the time needed for emergency services to get to the location of the crash; they do not prevent crashes.

There are also already products available that offer bicycle crash detection. These include sensors on helmets, offered by Specialized (ANGi), ICE (ICEdot), and Abus (Quin). Two e-bike companies offer an integrated crash detection system with sensors in the bicycles: *Cowboy* (2 and 3) and *Angell*. *Garmin* (Edge) makes bicycle computers with crash detection capabilities. This system is known to be oversensitive. *Mountainbike United* (TILT) developed a crash detection system with a sensor module that can be mounted to the front axis. There are also several apps available that provide bicycle accident detection services. All these systems have the same shortcomings: (1) they all require the rider to have a working and connected smartphone, (2) do not mention any performance metrics, and (3) do not handle all accident types.

These systems generally start a timer when an accident is detected. The user can stop the timer when no assistance is required. A message is sent to predefined contacts with the location of the accident when the user is not capable of stopping the timer.

1.1. Types of accidents

This study categorizes accidents into four types: collisions, skids, falls, and crashes. The first three types are single bicycle accidents, where no contact is made between the cyclist and another active road user, these account for approximately 60% of all cyclists admitted to Dutch hospitals [31]. Each of these accidents has distinctive kinematics which is why they all need to be considered for a proper accident detection. Existing bicycle accident detection research often does not consider edge cases in addition to normal cycling. Normal cycling is done on the road and also includes standing still. Edge cases are movements of the bicycle that happen during regular usage which does not comply with normal cycling. Examples of edge cases are: placing the bike in storage, cycling up or down curbs, lifting the bike, pushing the bike up or down stairs, etc. A robust accident detection should not give a

warning during these edge cases.

1.1.1. Collisions

A collision is defined as an accident where the bicycle or rider collides with a stationary object. This also includes 'dooring', where a car door suddenly opens just before the cyclist in its path, since the car is in this case a stationary object. Typical objects are: tree roots, curbs, bollards, and parked vehicles [28, 13, 31]. Collisions are described by the ICD-10 V17 external cause code (bicycle accident with a fixed object) and account for 2.3% of the annual bicycle fatalities in the Netherlands [32]. ICD-10 is a code for the list of international statistical classification of diseases and related health problems from the World Health Organisation. Despite the relatively low fatality rate, collisions account for approximately 18.3% of all bicycle accidents [31].

Existing research on collision detection generally uses accelerometer and gyroscope measurements. A self-organizing map (SOM) based two-phase detection algorithm can be used to detect collisions and road hazards for motorcycles [36]. This can be applied to bicycles because crash types and patterns of fault are similar for both two-wheelers [19]. The first phase distinguishes between smooth and irregular riding, and the second phase classifies these irregularities in hazards and non-critical scenarios. A second option is placing thresholds on the total measured acceleration a_{tot} and total angular velocity ω as defined in section . However, placing a threshold only on a_{tot} is not sufficient [9], but this is only validated on two collisions and four falls. A the third option is to add a piezo sensor at the front of the vehicle to detect deformations of the frame during a collision and place a threshold on the output of this sensor [4].

1.1.2. Skids

A skid is defined as an accident where the bicyclist loses control due to loss of friction between the tire and road. Typical examples include a lateral tire movement during turning on a slippery road or loss of traction on gravel. Of all cyclists admitted to Dutch hospitals, 23% attributed the road surface as a major contributor to their single bicycle accident [28]. The occurrence of the skid accident type is approximately 12.2% of all bicycle accidents [31]. All single bicycle accidents without collision are described by the ICD-10 V18 external cause code and account for 18% of all bicycle fatalities. However, this also includes falls as described in section 1.1.3.

Skids can be detected by placing a threshold on the filtered total acceleration a_{tot} and total angular velocity ω , measured by an inertial measurement unit (IMU), again placing a threshold only on a_{tot} is not sufficient, which is shown for motorcycles [6]. This is validated on real accidents with a stunt driver and edge cases, which include extreme braking, zigzags, accelerating, and fall-like maneuvers. The total measured acceleration during a skid first decreases due to gravity during free fall, which is why a threshold on the minimum value can also be possible [6, 23].

Another option is using a multivariate cumulative sum (MCUSUM) algorithm to provide an accident trigger. A cumulative sum algorithm assumes that observations follow a normal distribution (mean μ_0 , standard deviation σ_0), which can be learned from normal driving. The MCUSUM also takes the correlation between different types of measurements into account, like the correlation between a_{tot} and ω , thus uses the covariance matrix. This method proved to be effective for airbag deployment in motorcycles [2].

1.1.3. Falls

A fall is defined as an accident due to an unstable state of the bicycle and the driver. A bicycle without a rider is inherently stable when it has a sufficiently high velocity [12, 24, 26]. This type of accident includes low-speed mounting and dismounting accidents, which account for 10.7% of all single bicycle accidents [28]. These mounting mistakes happen significantly more often on e-bikes than on conventional bicycles [10, 33]. The drivers' behaviour can also attribute to an unstable state, for example by excessive braking or abrupt steering. Another example of a fall is attributed to general forces on the frame or steering assembly of the bicycle, like a gust of wind from a passing vehicle. Falls account for approximately 19.8% of all bicycle accidents [31].

Falls can be detected using a sensor mounted on a cycling helmet, which measures total acceleration, lean, and tilt [11]. Another option is by placing a magnetic, angular rate, and gravity (MARG) sensor on the steering assembly [39]. This measures four signals: acceleration, angular velocity, angle, and magnetic field. Data is gathered during normal cycling and simulated accidents where the bicycle is pushed and let go. There are 24 features gathered during these trials: the average and standard deviation in the x,y, z-direction of the four signals. These features are first normalized. This is a lot, thus principal component analysis (PCA) is used to reduce the dimensionality to 6 components while keeping the majority of variance in the data. A support vector machine (SVM) with a Gaussian kernel function is used to classify between cycling and falls. Adding roll rate can improve hazard detection for motorcycles by also identifying falls [36].

Significantly more research is done on fall detection for activities of daily living (ADL), some of these principles can be translated for the bicycle case. A waist-mounted accelerometer is the most popular sensor position since this is closest to the centre-of-mass of a person [8]. This would translate to a sensor positioned on the top bar or seat post of the bicycle. Including posture/lean angle decreases the number of false positives (wrong warning) of ADL fall detection, since a fall can be confirmed by a period of lying [8]. However, this makes the system less sensitive as this would not be the case for all falls. A typical fall pattern of the measured acceleration consist of a decrease during the fall, followed by an increase during impact [23].

Machine Learning vs Thresholds

Another principle that can be learned from ADL fall detection is the difference in the performance of different fall detection algorithms and classifiers. All prior research uses threshold triggers or machine learning classifiers trained on labeled data. A threshold is a value that needs to be exceeded to provide a trigger. Machine learning classifiers generally perform better than thresholds and they provide a general classification model, which thresholds do not [3]. Further improvement is suggested by combining machine learning (e.g. for impact detection) with thresholds (e.g. for post-impact confirmation) in a state machine [1, 3].

Machine learning algorithms can work with multiple signals, for example, acceleration and angular velocity in three directions. Calculating multiple features (mean, median, standard deviation, etc.) increases the dimensionality of the data which slows the computation down. A solution is to apply principal component analysis (PCA). This method maps the data on the principal components of the initial covariance matrix to reduce the dimensionality while keeping most of the variance. PCA can safely be applied for ADL fall detection without losing much performance as long as most of the variance in the data is kept [21].

1.1.4. Crashes

A crash is an accident involving contact with another active road user, these account for approximately 40% of all bicycle accidents [31]. This excludes evasive manoeuvres. These road users are typically cars, vans, or trucks, but they can also be other cyclists or pedestrians. Motorized vehicles account for approximately 64% of all collision partners in bicycle accidents [34, 13]. Cyclists are more often hit at the left side than the right, which is the other way around for left-side driving countries [41, 34, 5]. Frontal car crashes on the side of the cyclist are the most common [5, 30]. Even though multi-vehicle crashes occur less than single-bicycle accidents, they account for 72.5% of fatalities in bicycle accidents [32]. Crashes with a low occurrence, but particular high severity include frontal car, bicycle rear contact, and blind-spot impacts [20, 14].

Another type of crash is caused by contact with another cyclist or pedestrian. These include a crash where the handlebars of two bicycles lock when the cyclists drive too close to each other. Of all bicyclists admitted to Dutch hospitals, 2% and 13% were the result of a crash with a pedestrian and another cyclist respectively [13]. Little is known about specific bicycle crash detection as this is more dangerous to recreate than single bicycle accidents. However one might argue that when a system can detect falls, skids, or collisions, it should also be able to detect more severe crashes.

The occurrence of the described accident types can be summarized in figure 1.1. The presented values are estimations, based on literature as described above. The other/unknown category includes accidents due to bicycle malfunction and stunting. Many accidents occur due to a combination of

these categories, which is why the division has been made based on the initial cause.

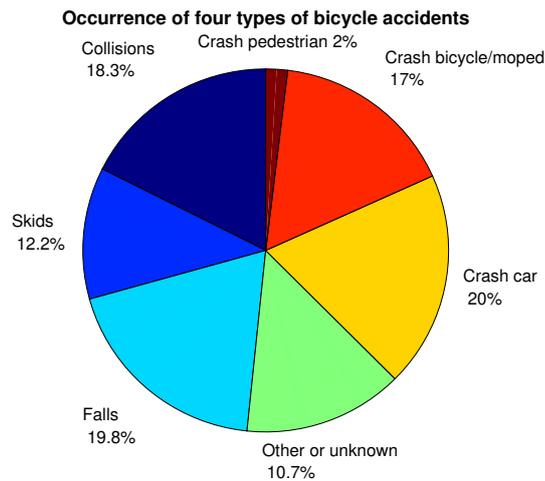


Figure 1.1: Pie chart with estimations for the occurrence of the described accident types.

1.2. Connected bicycles of Royal Gazelle

Royal Gazelle is a Dutch bicycle manufacturer that supplements some of its e-bikes with 'connected' technology. This translates to an Internet of Things (IoT)-module in the frame which can send information about the state and location of the bicycle over the internet to a central server or via Bluetooth to the users' smartphone [27]. The data collection rate from the receiving server is currently limited to 2 seconds. These modules support GPS, Glonass, and Galileo for location tracking and also have an IMU sensor [16]. This system is currently used for theft detection, but there is a demand to expand this with an accident detection feature [37, 27].

Currently, the following metrics are sent by the module: front and rear light status, power assist level, charging status, battery voltage, state of charge, Global System for Mobile Communication (GSM) signal strength, speed of bike, odometer, remaining range of bike, battery sock of the module, battery voltage of module, battery temperature of the module, average GPS horizontal dilution of precision (hdop), GPS hdop, GPS latitude, GPS longitude [16]. The IoT-module has an IMU which is currently only used to "wake up" the software. It is possible to add IMU signals to the data package. The location and orientation of this module are dependent on the bicycle model. Thus each bicycle model will need to supply the location and orientation of the module to the accident detection computation, which uses a bicycle-oriented reference frame, as in figure 1.

1.3. Objective

The aim of this study is to develop a method for the practical implementation of bicycle accident detection and geolocation for connected bicycles which I call **ALARM: Accident Localisation And Recognition Method**. A demonstration prototype is made as a validation and a minimum viable product is proposed for bicycle manufacturers.

The hypothesis for this research is as follows:

With a comprehensive data set of normal cycling, edge cases, and multiple types of simulated accidents a machine learning based algorithm can be created that utilizes the IoT technology of connected bicycles and can detect at least 80% of single bicycle accidents.

Such a system should comply with the following requirements and constraints:

- **The system provides no false positives.** This means that no warnings are sent when no accident has occurred since this would be problematic for the user and the emergency contacts. Such a requirement is a trade-off between false warnings and missed accidents, thus it is expected that some low severity accidents will be missed.
- **The system has a minimum sample time of 3 seconds.** This is based on the minimum sample time of the Royal Gazelle's cloud server.
- **The input to the accident detection system is provided only by sensors in the IoT-module of Royal Gazelle.** These are an IMU, wheel speed sensor, and GPS/GNSS receiver. The IMU provides gyroscopic and accelerometer signals relative to the orthogonal body-fixed axis.
- **The system automatically sends a message with the location of the accident within 30 seconds to a predefined set of contacts.** The current e-bikes are not yet able to set up a voice connection like the e-call system. However, they can send their location over the internet. The user can set a predefined set of ICE (In Case of Emergency) contacts which will receive a message with instructions and the location of the accident. This message is sent within 30 seconds after the accident has occurred to provide quick assistance.

The accident detection method is described in chapter 2. First the data collection is explained in section 2.2. This is followed by a description of the input for the accident detection algorithm in section 2.3 and explanation of the algorithm in section 2.4. Different classifiers can be substituted in the algorithm, the threshold and machine learning based classifiers are explained in section 2.5 and 2.6 respectively. Section 3.1 shows the performance differences of the algorithm for these different classifiers, from this the best performing classifier is found. This is followed by the resulting performance of the whole system in section 3.1. Then the minimum viable product and a prototype system are explained in chapter 4. This is followed by a discussion on the interpretation of the results, the shortcomings of the system, and a list of proposed improvements in chapter 5. Finally, the conclusions can be found in chapter 6.

2

Methods

2.1. Approach

There are three types of approaches possible for the development of an accident classification algorithm: mathematical, simulation, and data-driven [6]. The first describes the interactions between the bicycle, the rider, and the environment with a mathematical model. This is very challenging since an accident often has multiple influencing and unknown factors. A simulation-based approach relates simulator data to real live data, this is possible with a physics engine like MADYMO [40, 30]. However, just like a mathematical approach, a simulation approach would make several assumptions that could leave out relevant dynamics during an accident as there are often multiple influencing factors at play. It is also very challenging to make realistic simulations of complex motions like accidents. This is why **ALARM** uses a data-driven approach, where cycling behaviour is described with data collected during normal cycling, which includes edge-cases, while accident data is collected in simulated real-world accidents as explained in section 2.2.

The approach consists of multiple iterations to get to the end result. These iterations have an increasing algorithm complexity and data-set size. Some preliminary experiments consisted of IMU data collected during 2 hours of normal cycling and 7 falls. This confirmed the typical fall pattern as described in section 1.1.3 and that only a threshold on the acceleration a_{tot} would not be sufficient.

2.2. Data Collection

Representative IoT-module data is collected using smartphones that are rigidly mounted on the bicycles, as shown in figure 2.2. These smartphones log time series of acceleration and angular velocity from the IMU sensor in three directions as well as GNSS/GPS signals. The MATLAB Mobile app is used to log the data. The IMU data is sampled at approximately 100 Hz and GPS at 1 Hz.

Preliminary experiments are carried out to determine the performance of these smartphone sensors. The preliminary experiments consist of 7 falls at various velocities and 2 hours of normal cycling. The resulting measurements of the acceleration and angular velocity showed a clear distinction between cycling and falling. The typical fall pattern: a decrease of measured acceleration followed by an increase, is also visible in the data. No detectable drift can be seen in the data. Thus the performance of this setup is deemed sufficient. Since this research is not aimed at finding the exact accelerations during cycling but instead focuses on the differences between cycling and accidents, no extensive calibration is needed.

The acceleration and angular velocity signals are truncated at $10^{-6} m/s^2$ and $10^{-6} rad/s$ respectively. The GPS signals are truncated at 10^{-5}° for longitude and latitude at $10^{-3} m/s$ for speed. The data is collected with *Lenovo K6* and *Nokia 3.4* smartphones. The first also has a magnetometer from which the data is also stored, but no GPS receiver. The second does not have a magnetometer.

A quarter of the normal cycling data and 80% of accident test runs are used to train machine learning classifiers, which will be explained in section 2.6. Another quarter of normal cycling and the remaining 20% of accidents are used to compare the performance of the classifiers. These classifiers can be

substituted in layer 1 of the accident detection algorithm, which will be explained in section 2.4. The remaining 50% of normal cycling data is used to set thresholds to detect movement, standing still, and lying down. These thresholds are part of layer 0 and layer 2 of the algorithm (section 2.4). These runs all have at least 90% GPS coverage. This part of the normal cycling data set also has 6 test runs that contain both cycling and accidents, which are used to validate the whole system. This data distribution is visualized in figure 2.1 and the properties of each individual test run can be found in appendix B.

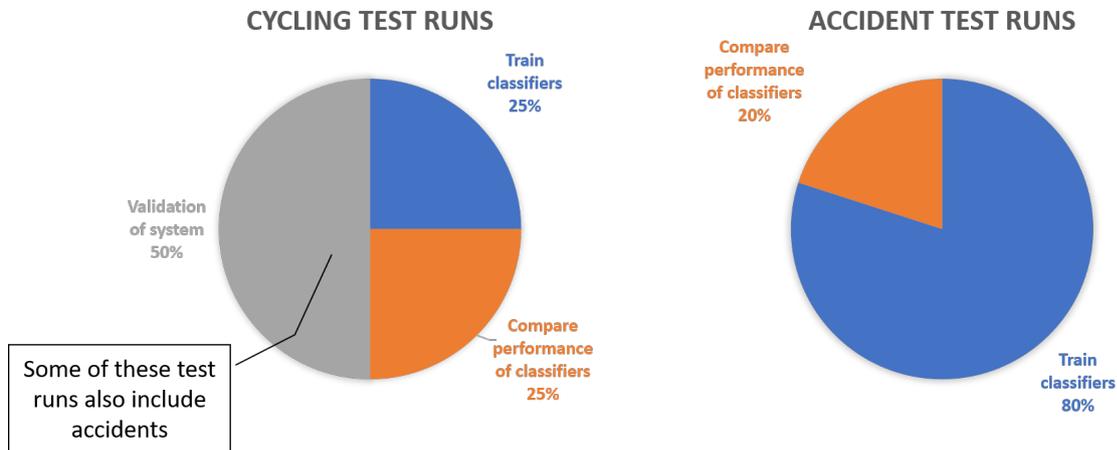


Figure 2.1: Data distribution of test runs. Left: the normal cycling runs are randomly distributed in 25% training and 25% validation for the classifiers. The remaining 50% also includes some accidents which are used to validate the system. Right: the accident test runs are randomly distributed in 80% training and 20% validation.



Figure 2.2: Smartphone rigidly mounted at the back of the seat post.

2.2.1. Normal Cycling and Edge Cases

A total of 54 hours of normal cycling is collected by 7 riders in 59 test runs for training and validation purposes. These test runs range from long rides on smooth cycling paths to short trips in a city centre on stone roads and over bridges around South Holland and Utrecht. All participants were given instructions (Appendix A) to turn on the measurements before unlocking their bike and turn off the measurements after locking the bike. This was to make sure that any edge-cases that could occur while storing the

bike were also captured. The participants were also asked to drive roughly sometimes, like driving over bumps and curbs, as long as this is within the traffic regulations to find out if the system falsely detects accidents during any realistic usage of the bicycle. The following edge cases are included in the normal cycling data set:

- Cycling up and down curbs.
- Cycling in a circle on a steep road.
- Cycling in the road verge.
- Cycling on dirt roads.
- Dinking, cycling with someone on the bicycle rack.
- Swerving.
- Placing bicycle in and out of storage.
- Store bicycle on a double bicycle rack.
- Harsh braking and accelerating.
- Evasive manoeuvres.
- Slowly lying bicycle on the ground.
- Push bicycle up or down stairs.
- Lift bicycle up or down stairs.
- Lift bicycle up or down ledges.
- lay bicycle on the ground.
- Push bicycle forward.
- Move bicycle in an elevator.
- Lean bicycle against walls and railings.

A visualisation of the gathered data during an example test run can be found in figure 2.3. This figure shows the measured accelerations, angular velocities, and GPS coordinates during the run. On the lower right corner, 10 features of the total acceleration and angular velocity are shown. This test run includes normal cycling and the edge cases: swerving, placing bicycle out of storage, harsh braking and accelerating, and lifting the bicycle down some stairs. The GPS signal during this test run is accurate. Another test run is visualized in appendix C.1, this run includes cycling in the city centre which does not include GPS data, because a phone without a GPS receiver was used for this test run.

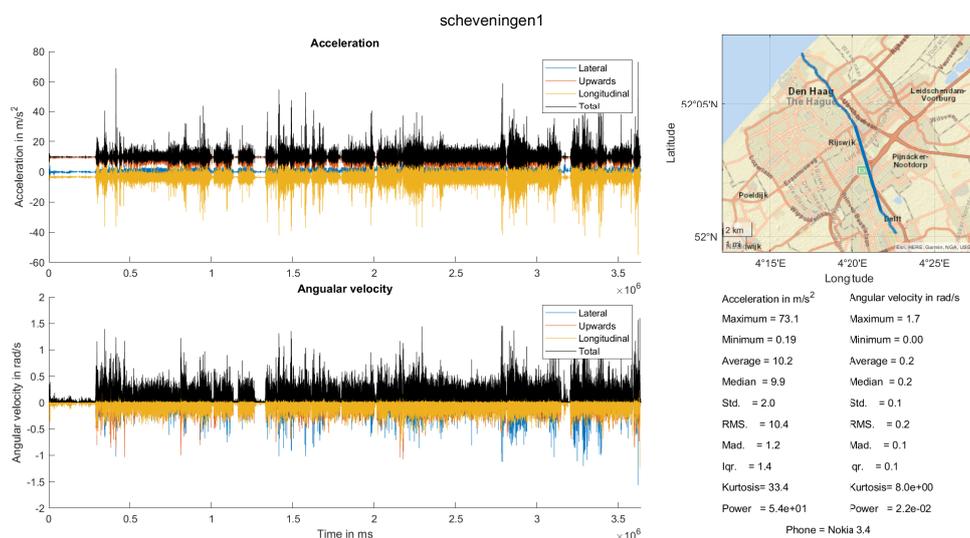


Figure 2.3: Visualisation of the collected data during a test run from Delft to Scheveningen which includes normal cycling and various edge cases. This data contains acceleration and angular velocity in three directions and the total values. From these total acceleration and angular velocity signals, 10 features are shown in the lower right corner. The GPS coordinates are shown on the map in the upper right corner.

2.2.2. Simulated Accidents

Simulated accidents are conducted to gather accident data for training and validation purposes. These accidents consisted of pushing the bike at different velocities into different objects, to the ground, or pulling it with a rope. The velocities are approximated using the wheel speed sensor from the bicycle. These test runs are conducted on grass to protect the bike unless otherwise stated. The test runs at high speed (22 km/h) are done at least twice. The simulated accidents are:

Falls

The bicycle is pushed and let go at the following velocities: 0, 5, 12, 22, 22 km/h, which resulted in test runs: F1,F2,F3,F4,F5 respectively.

Falls: Push

The bicycle is pushed forwards and then the frame is pushed at the side at the following velocities: 0, 5, 12, 22, 22 km/h, which resulted in test runs: P1,P2,P3,P4,P5 respectively. For example, P4 is shown in figure 2.4 and appendix C.3.

Collision: Curb (90°)

The bicycle is pushed into a curbstone perpendicular to the movement of the bike and let go at the following velocities: 5, 12, 22, 22, 22 km/h, which resulted in test runs: C1,C2,C3,C4,C5 respectively.

Collision: Curb (45°)

The bicycle is pushed into a curbstone laid 45° to the movement of the bike and let go at the following velocities: 5, 12, 22, 22 km/h, which resulted in test runs: S1,S2,S3,S4 respectively. For example, S4 is shown in figure 2.5 and appendix C.4.

Collision: Curb (15°)

The bicycle is pushed into a curb laid 15° to the movement of the bike and let go at the following velocities: 5, 12, 22, 22 km/h, which resulted in test runs: G1,G2,G3,G4 respectively.

Collision: Wall (90°)

The bicycle is pushed into a wall perpendicular to the movement of the bike and let go at the following velocities: 5, 12, 22, 22 km/h, which resulted in test runs: W1,W2,W3,W4 respectively. For example, W3 is shown in figure 2.6 and appendix C.5.

Collision: Wall (45°)

The bicycle is pushed into a wall laid 45° to the movement of the bike and let go at the following velocities: 5, 12, 22, 22 km/h, which resulted in test runs: H1,H2,H3,H4 respectively. For example, H4 is shown in figure 2.7 and appendix C.6.

Falls: Pull Steer

The bicycle is pushed forwards and let go while a rope attached to the steering assembly pulls the bike to the right at the following velocities: 5, 12, 22, 22 km/h, which resulted in test runs: T1,T2,T3,T4 respectively.

Skids: Pull Frame

The bicycle is pushed forwards and let go while a rope attached to the bottom of the frame pulls the bike to the right at the following velocities: 5, 12, 22, 22 km/h, which resulted in test runs: Q1,Q2,Q3,Q4 respectively. For example, Q4 is shown in figure 2.8 and appendix C.7.

Skids and falls: Slippery Surface

Some of the test runs are also conducted on a snowy surface to simulate skids. The following runs are conducted on snow:

- **Falls:** at 0, 12, 20, 20 km/h, resulted in test runs D1,D2,D3,D4 respectively.
- **Push Falls:** at 5, 12, 20, 20 km/h, resulted in test runs D5,D6,D7,D8 respectively.
- **Push Falls:** at 5, 12, 20, 20 km/h, resulted in test runs D9,D10,D11,D12 respectively, these runs are conducted on a snowy road, not on grass. For example, D10 is shown in figure 2.9 and C.8.

- **Roll:** at 5, 12, 20 *km/h*, resulted in test runs D13,D14,D15 respectively. During these test runs, the bicycle is pushed in two directions at the bicycle rack to create a roll motion.



Figure 2.4: Push fall (P4), a simulated fall where the bicycle is pushed forward and then to the side at 22 *km/h*. See also appendix C.3.



Figure 2.5: Collision into a curb (S4), a simulated collision where the bicycle is pushed into a curb at 45° at 22 *km/h*. See also appendix C.4.



Figure 2.6: Collision into a wall (W3), a simulated collision where the bicycle is pushed into a wall at 90° at 22 *km/h*. See also appendix C.5.



Figure 2.7: Collision into a wall (H4), a simulated collision where the bicycle is pushed into a wall at 45° at 22 km/h . See also appendix C.6.



Figure 2.8: Skid (Q4), a simulated skid where the bicycle is pushed and let go at 22 km/h . When the bicycle is let go, the bottom of the frame is pulled to the right with a rope. See also appendix C.7.



Figure 2.9: Skid-Fall (D10), a simulated skid where the bicycle is pushed forward and then to the side at 20 km/h on a snowy road. See also appendix C.8.

2.2.3. Time shifts

An accident usually takes less than 3 seconds. During this time, high peak accelerations and angular velocities can be measured. The system is constrained to have a minimum sample time of 3 seconds. To maximize the difference between an accident and a non-accident sample, it is vital to capture the whole accident in a single sample. However, there is no way to guarantee this since the data is temporal and only non-overlapping samples are used. The start time of the accident is manually determined so that the following 3 seconds contain the maximal variation in the data. This is the optimal case since such a sample captures most elements of the accident, shown on the left in figure 2.10. However, as explained, this is not always the case, shown as an example on the right in figure 2.10. That is why the algorithm is also trained on accident samples with a shifted start time of -1 second and -2 seconds,

to make sure that these sub-optimal cases are also captured. The time shift which cuts the accident exactly in half is the least optimal since this would result in two samples that each capture only half of the accident.

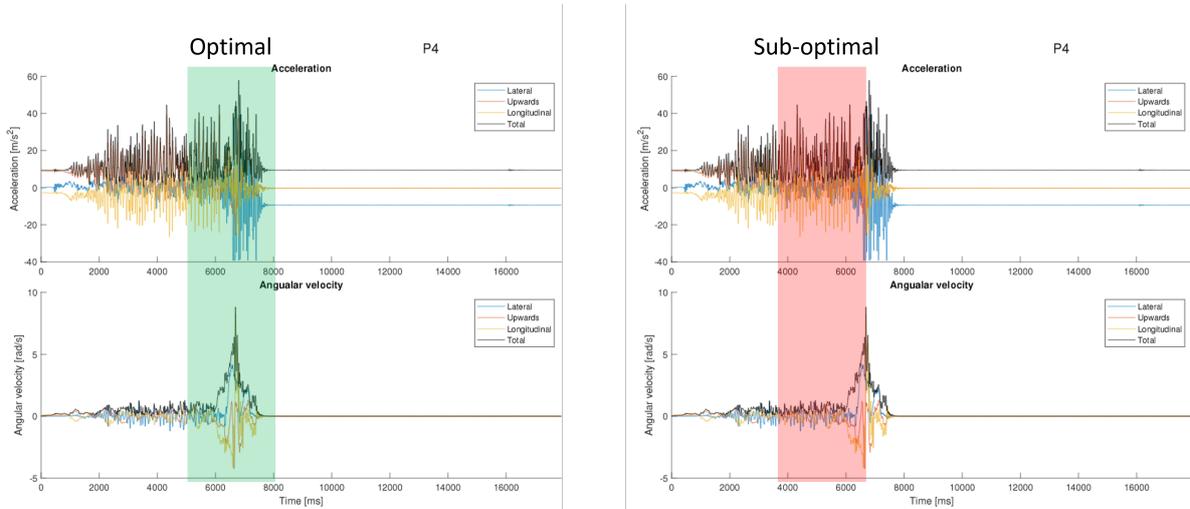


Figure 2.10: A visualisation of the IMU data from a typical fall (P4) with two possible time shifts. On the left the optimal case and on the right a sub-optimal one. The fall happens within the optimal region.

2.3. Sample Description and Feature Calculation

Figure 2.11 shows the different steps that are undertaken to go from the (raw) IMU and GPS signals to the features of a sample. These features are used in the accident detection algorithm, which will be explained in section 2.4. From the GPS signal, only longitude, latitude, and speed are used. The IMU data consists of 6 signals: x,y,z, and total for both acceleration and angular velocity. The test runs are first divided into accident or normal cycling. All normal cycling time series signals are sectioned into three-second samples s_i . The start time of the accident test runs is manually labeled and time shifts are applied to this start time to get the three-second accident samples. For each 3-second sample of each signal, 12 features are computed. This results in 96 features per sample. These features are based on previous research [39, 23, 1, 25] and can be calculated for each signal x consisting of n values using equations 2.1-2.12:

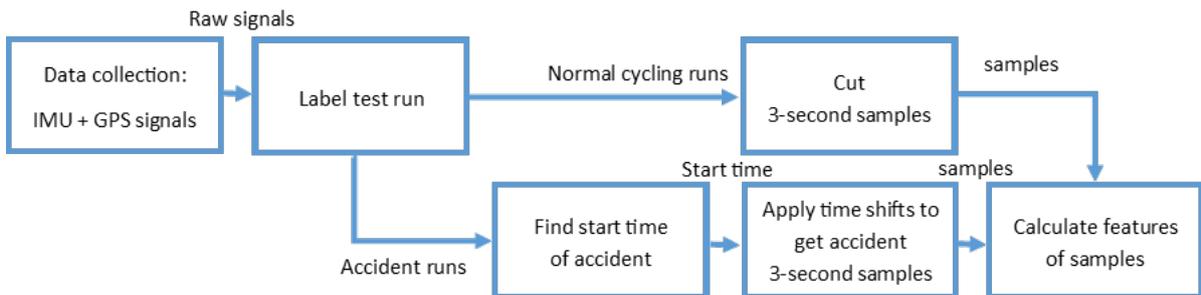


Figure 2.11: Data-flow from data collection to feature calculation. The boxes represent axis and the labelled arrows represent data.

Maximum

$$\max(x) = \max(x_1, x_2, \dots, x_n) \quad (2.1)$$

Minimum

$$\min(x) = \min(x_1, x_2, \dots, x_n) \quad (2.2)$$

Average

$$\text{avg}(x) = \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad (2.3)$$

Median

$$med(x) = x_{\frac{n}{2}} \quad (2.4)$$

Standard deviation

$$std(x) = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}} \quad (2.5)$$

Root mean square

$$rms(x) = \sqrt{\frac{\sum_{i=1}^n x_i^2}{n}} \quad (2.6)$$

Mean absolute deviation

$$mad(x) = \frac{\sum_{i=1}^n |x_i - \bar{x}|}{n} \quad (2.7)$$

Interquartile range

$$iqr(x) = x_{n*0.75} - x_{n*0.25} \quad (2.8)$$

Skewness

$$ske(x) = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^3}{std(x)^3} \quad (2.9)$$

Kurtosis

$$kur(x) = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^4}{[\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2]^2} - 3 \quad (2.10)$$

Total average power

$$pwr(x) = \frac{\sum_{i=1}^n x_i^2}{2n + 1} \quad (2.11)$$

Energy

$$eng(x) = \sum_{i=1}^n x_i^2 \quad (2.12)$$

2.4. Accident Detection Algorithm

The features of the three second samples are used in in the accident detection system. This system uses a 3 layer algorithm to determine whether an accident has occurred based on the IMU and GPS input data. This combines the robustness of machine learning with the logic of thresholds in a state machine [1, 3]. The algorithm is depicted as a flowchart in figure 2.12. The algorithm is started when the bicycle is turned on. Layer 0 uses the first three second sample to detect if the bicycle is moving, for which either GPS speed or range of total acceleration needs to exceed the corresponding threshold. Layer 1 is run when cycling is detected and uses the next sample. This layer classifies between cycling and an accident, for example where at least one of three thresholds needs to be exceeded to classify an accident. When an accident is detected, layer 2 is looped three times to confirm the accident, using a counter parameter C. A new three second sample is used each time and at least one of three thresholds need to be exceeded to confirm an accident and send an e-mail with the location. When layer 1 returns cycling or layer 2 is run three times, the algorithm moves back to layer 0.

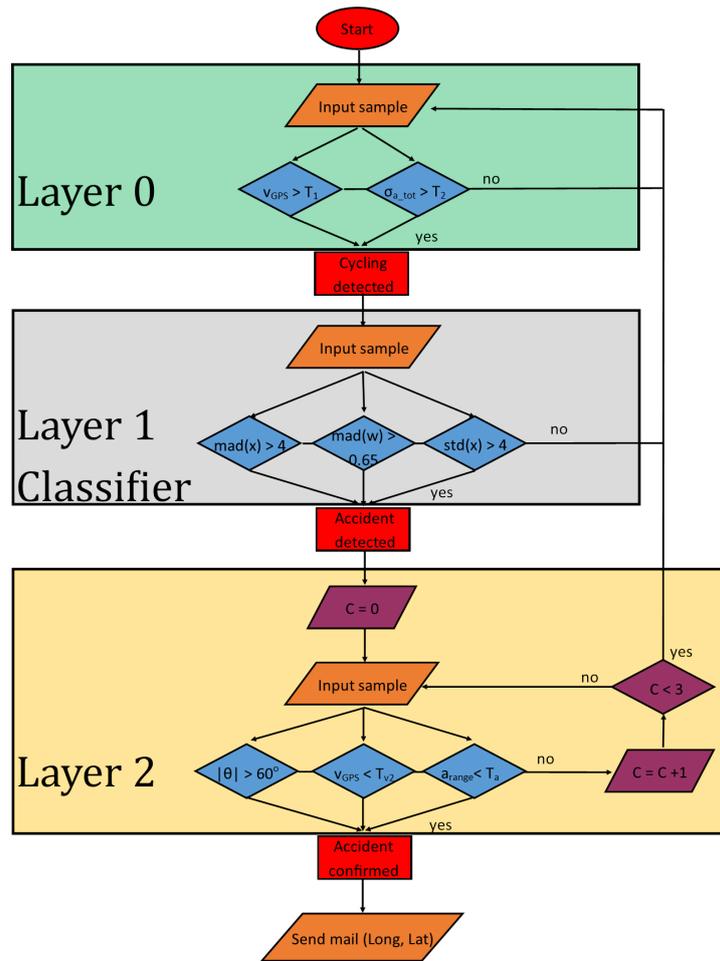


Figure 2.12: Flowchart of the 3 layer algorithm with an example threshold based classifier. The rectangles are state descriptions. A yellow parallelogram indicates input or output and a purple parallelogram indicates a parameter action. A diamond indicates an if-statement which can either be true (yes) or false (no).

Layer 0 checks whether the bicycle is moving on the previous sample s_{i-1} . This is to prevent any false accident warnings, for example when the bicycle falls over when it is parked. To check whether the bicycle is moving, the GPS-based speed and the range of total acceleration a_{tot} are used. The latter is added since GPS is not always reliable and the sensors measure vibrations when the bicycle is moving. Either one of them should exceed the corresponding threshold to indicate cycling.

Layer 1 is a classifier which distinguishes accidents from normal cycling on sample s_i . Two threshold based classifiers and multiple machine learning based classifiers can be substituted in this layer. These threshold classifiers are explained in section 2.5 and the machine learning classifiers are explained in section 2.6.

Layer 2 is used to confirm whether an accident has occurred on sample s_{i+1} , s_{i+2} , and s_{i+3} . This is because an accident is probably followed by a period of lying down. However, this should not necessarily be during the next sample, which is why this layer is looped three times. In every loop, the lean angle θ , GPS speed, and range of a_{tot} are used. This lean angle is approximated by measuring the acceleration in lateral direction \ddot{x} , which should measure $1g$ when lying perfectly flat. Thus an accident is confirmed when \ddot{x} has exceeded its threshold. The GPS-based speed and range of a_{tot} are added because a bicycle should not necessarily be laying flat to confirm an accident. They are used the same way as in layer 0, but should now be lower than their corresponding thresholds to confirm an accident.

The 3 layer algorithm is depicted as a flowchart in figure 2.12 and as pseudo-code in algorithm 2. An example threshold based classifier is shown in algorithm 1.

Algorithm 1: Example Threshold based Classifier

```

Result: class.
set C1,C2,C3;
if mad lateral acceleration is larger than C1 OR mad total angular velocity is larger than C2 OR
  std lateral acceleration is larger than C3 then
  | return accident;
else
  | return cycling;
end

```

Algorithm 2: Three Layer Accident Detection

```

Result: Send a warning with longitude and latitude when an accident is confirmed.
set T1,T2,T3,T4,T5;
init classifier;
while bike is tuned on do
  | input new features;
  | if speed is larger than T1 OR range of accelerations is larger than T2 then
  | | % cycling is detected;
  | | input new features;
  | | call classifier returning class;
  | | if class is accident then
  | | | % accident is detected;
  | | | set loop is 1;
  | | | while loop is less than 4 do
  | | | | input update features;
  | | | | compute absolute lean angle;
  | | | | if absolute lean is larger than T3 OR speed is smaller than T4 OR range of
  | | | | accelerations is smaller than T5 then
  | | | | | % accident is confirmed;
  | | | | | output warning with longitude and latitude;
  | | | | end
  | | | | add one to loop;
  | | | | end
  | | | end
  | | end
  | end
end

```

2.4.1. Performance Metrics

The performance of any system can be represented in many ways. Standard performance metrics for binary classification algorithms are a Receiver Operating Characteristics (ROC) curve, average accuracy, precision, recall, and specificity. The most relevant performance metrics for the accident detection system are precision, sensitivity, and specificity because these combined provide intuitive information about the output of the algorithm (accident or normal cycling). Using only one performance metric is not sufficient to depict the performance of the system. To get the best indication of the performance all three metrics should be discussed. These are all based on the values in the confusion matrix, where a positive result indicates a detected accident, as shown in table 2.1. Each row represents samples from the true class, which is the type of test run the sample is from (accident or normal cycling). Each column represents samples from the predicted class, which is the resulting output from the algorithm.

Confusion matrix		True class	
		Accident	Cycling
Predicted class	Accident	TP	FP
	Cycling	FN	TN

Table 2.1: Confusion matrix. TP = true positives: accident detected as accident, FP = false positives: cycling detected as accident, FN = false negatives: accident detected as cycling, TN = true negatives: cycling detected as cycling.

Precision

Precision is a measure that tells what proportion of confirmed accidents is a true accident, which is calculated using equation 2.13. This should be 100% when the user does not want to alarm the predefined contacts when no accident has occurred (no false warnings). However, the use of precision as a performance metric introduces a problem when increasing the amount of accident data while developing the algorithm. Adding more accidents to the data set while developing the algorithm would be logical since accidents are so rare that any new accident gives more insight. An increase of accident samples will increase the number of true positives, while the false positives stay roughly the same, which will result in higher precision. This is undesirable as the amount of data should not correlate with the performance of the system while developing. Another problem with (only) using precision is that when a system manages to only capture 1 accident, and it is correct, it will be 100% precise, but this is not useful.

$$Precision = \frac{TP}{TP + FP} \quad (2.13)$$

Sensitivity or Recall

Sensitivity is a measure for the proportion of accident samples that are classified as an accident. This is calculated with equation 2.14. When no missed accidents are desired, sensitivity should be 100%. Sensitivity is not about classifying correctly, but more about capturing all accidents. So a problem with (only) using sensitivity is that when a system classifies everything as an accident, it will have a sensitivity of 100%, but this is not useful.

$$Sensitivity = \frac{TP}{TP + FN} \quad (2.14)$$

Specificity

Specificity is a measure for the proportion of normal cycling samples that are predicted as normal cycling, as calculated using equation 2.15. This is the exact opposite of sensitivity. When it is desired that normal cycling is always detected as normal cycling, thus there are no false warnings, the specificity will be 100%. So a problem with (only) using specificity is that when a system classifies everything as normal cycling it will have a specificity of 100%, but this is not useful.

$$Specificity = \frac{TN}{TN + FP} \quad (2.15)$$

2.5. Threshold Based Classifiers

Many accident detection systems use thresholds to classify between falling and cycling/ADL [8, 11, 17, 1, 38, 9]. A threshold is a value that a signal has to exceed to be classified into a different category. An evaluation of the gathered data shows that the acceleration and angular velocity produce high peaks during an accident. However, placing a threshold on these raw signals is not sufficient for accident detection, since normal cycling and edge cases can also produce these high peaks. Multiple combinations of thresholds can form a classifier that can be substituted in layer 1 of the algorithm. These classifiers place thresholds on the features of a sample, which can be calculated using equation 2.1 to 2.12. The features used in these classifiers can be determined as follows:

1. Approximately 2 hours of normal cycling data and 39 simulated accidents (not on snow) data is collected as described in section 2.2. The start time of the accidents is labeled for each accident test run.
2. Edge case data is collected on a steep road, while swerving, cycling over curbs, and placing the bicycle in storage.

3. The normal cycling and edge case data is cut into three-second samples. The accident samples were determined using the start time. The features are calculated for each sample. The features from the accidents are compared with features from normal cycling and edge cases. Thresholds are manually placed by the features with a clear boundary between cycling and accidents. This resulted in a list of 14 potential thresholds. The distribution of samples for each feature and the list of thresholds can be found in appendix E.
4. Apply a -2 second time shift to the start time of the accident samples and calculate the resulting features. None of these features show a clear boundary between cycling and accidents. However, almost all samples have at least one feature that passes a threshold, as shown in figure 2.13. The table in appendix E.1 shows which thresholds are passed for each accident. This shows that a fully threshold based classifier can not detect accidents C1 and T2 with a least-optimal time shift.

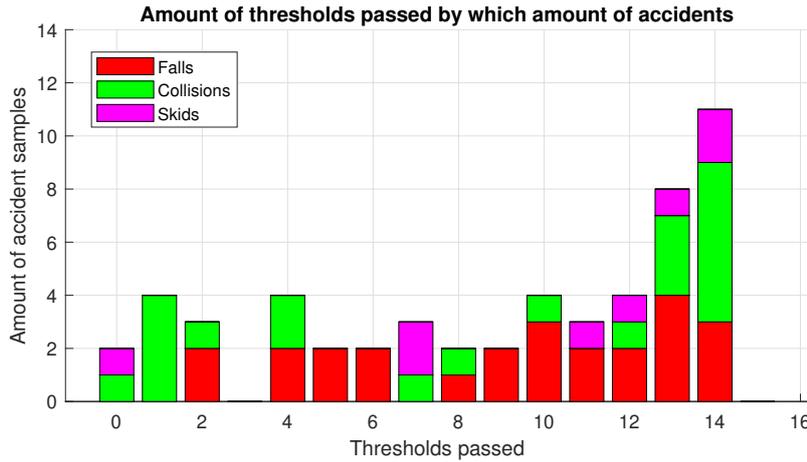


Figure 2.13: The number of accident samples with a least-optimal time shift that passes a certain amount of thresholds. Normal Cycling and edge cases do not pass any thresholds.

5. From table E.1, it can be concluded that the combination with the least amount of thresholds able to detect most accidents is: maximum total angular velocity $max(\omega)$ with minimum angular velocity in longitudinal direction $min(\dot{\theta})$. This combination is only not able to detect accidents C1 and T2 with the least optimal time shift. However, using minimum and maximum values of a signal might be problematic, since these could be outliers that are not representative of the data. The next combination with the least amount of thresholds able to detect most accidents is: mean absolute deviation of acceleration in lateral direction $mad(\ddot{x})$, mean absolute deviation of total angular velocity $mad(\omega)$, and standard deviation of acceleration in lateral direction $std(\ddot{x})$. This combination of thresholds is not able to detect accidents F2, C1, C2, S1, W1, W2, H1, and T2. The first combination is called classifier T1 and the second is called classifier T2. The latter is shown as an example in algorithm 1.

The values for the thresholds of T1 and T2 found using the method explained above are:

- Maximum total angular velocity $max(\omega) > 3.75rad/s$.
- Minimum angular velocity in longitudinal direction $min(\dot{\theta}) < -2.1rad/s$.
- Mean absolute deviation of acceleration in lateral direction $mad(\ddot{x}) > 4m/s^2$.
- Mean absolute deviation of total angular velocity $mad(\dot{\omega}) > 0.65rad/s$.
- Standard deviation of acceleration in lateral direction $std(\ddot{x}) > 4m/s^2$.

2.6. Machine Learning Classifiers

To try to increase the robustness of the system, multiple machine learning based classifiers are substituted in layer 1 of the algorithm. These classifiers are trained on 25% of all normal cycling data and 80% of the accident data as shown in figure 2.1. Validation is done on 25% of all normal cycling data and the remaining 20% of accidents which are distributed randomly. The accident data for training purposes consists of only one sample per accident since this generally takes less than 3 seconds. As explained in section 2.2.3, three time shifts are applied to the accident training data set. Only 25% of the cycling data is used to decrease the computational complexity of the training phase. This is possible since there is much more normal cycling data available than accident data.

Figure 2.14 shows the follow up data-flow from feature calculation to evaluation of the system, this figure adds to the figure shown in section 2.3. First the features of a single sample are normalized between 0 and 1, and reduced in dimension using PCA, which will be explained in section 2.6.1. The test runs and corresponding (reduced) samples are divided according to figure 2.1. One part is used to train the classifiers, one to validate them and one part is used evaluate the whole system. This last validation is done using the best performing classifier.

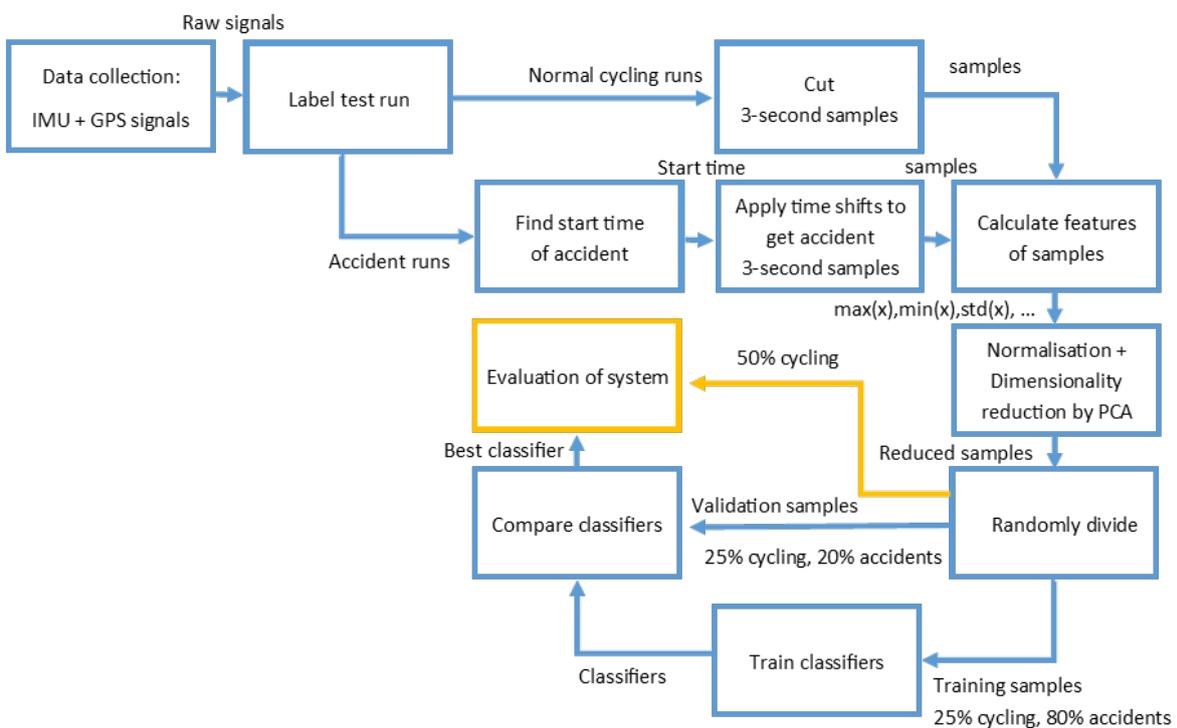


Figure 2.14: Data-flow from data collection to system evaluation. The boxes represent axis and the labelled arrows represent data. The yellow parts are used to form the final results.

Cost Parameter

To train the machine learning classifiers, approximately 32000 cycling samples and 126 accident samples are used. A cost parameter is used to compensate for this imbalance in the data. This parameter places a higher penalty on misclassifying accidents as cycling and a lower penalty is on misclassifying cycling as an accident. A simple machine learning algorithm would not work well without a cost parameter on unbalanced data. The performances of the classifiers are compared for cost parameters with a value of 20, 50, and 100. The following classifiers are compared and discussed: Support Vector Machines (SVM), K-Nearest Neighbours (KNN), Naive Bayes (NB), and Decision Trees (DT).

2.6.1. Principal Component Analysis

As described in section 2.4, one sample consists of 96 features. To decrease the computational complexity, while keeping most of the variance, the dimensionality is reduced with Principal Component Analysis (PCA). This is also beneficial for lowering the data costs when classification is done on an external server.

The training test runs are cut into three second samples. For each sample, the 96 features are calculated as explained in section 2.4. The training data set is a feature matrix of $\mathbf{T}_{m \times n}$ where $n = 96$ is the number of features and m is the number of samples. This training data set is first mapped between 0 and 1 resulting in a normalized feature matrix $\mathbf{A}_{m \times n}$. The covariance matrix \mathbf{C} of $\mathbf{A}_{m \times n}$ is computed. From this covariance matrix the eigenvectors \mathbf{v}_i and corresponding eigenvalues λ_i are calculated. These eigenvalues are sorted in descending order ($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_n$) and the corresponding eigenvectors are placed in matrix \mathbf{w} in this order. The training data projected on the PCA subspace $\mathbf{P}_{m \times k}$ is the projection of matrix \mathbf{A} on the k largest eigenvectors of \mathbf{w} .

Figure 2.15 shows the projection on the two largest eigenvalues (principal components). One C-shaped point cloud mainly contains cycling and edge cases. One oval point cloud mainly contains falls, collisions, and skids. A good classifier should have the decision boundary in between these two point clouds. The explained variance of each eigenvector can be calculated by dividing the corresponding eigenvalue by the sum of eigenvalues. To prevent over-fitting while keeping useful dimensions, the PCA is set to keep at least 85% of the variance for which $k = 6$ eigenvalues need to be used as shown in figure 2.16.

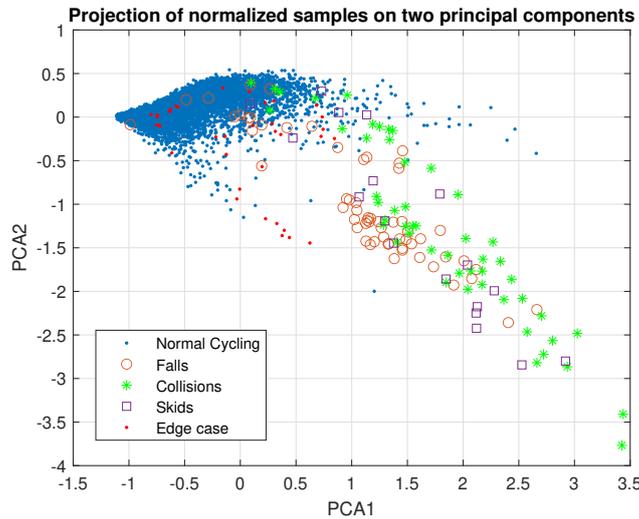


Figure 2.15: Projection of feature samples on the two largest principal components PCA1 and PCA2. These samples resulted from test runs of normal cycling, 3 types of accidents, and edge cases.

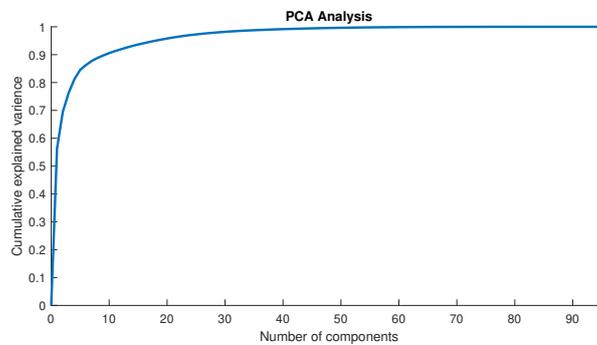


Figure 2.16: Explained variance for each number of components resulting from the Principal Component Analysis of the training data set.

2.6.2. Support Vector Machine

A Support Vector Machine (SVM) is often used as an accident classifier [39, 18, 21, 3]. Three different SVMs are trained on the projected samples $\mathbf{P}_{m \times k}$, each using a different kernel function. Support vector classifiers focus on the edges of the two point clouds (cycling and accidents), a threshold is placed between these. SVMs have a bias/variance trade-off where some misclassification is allowed to make them less sensitive to outliers. This results in 'soft margins', where some outliers are purposely misclassified to get overall better performance. A support vector classifier results from cross-validating multiple soft margin classifiers. The support vectors are samples that lie within the soft margin. Since these samples are 6 dimensional, the support vector classifier results in a hyperplane.

The data is not linearly separable, which means that no straight line or flat plane can be drawn which divides the cycling and accident samples. Support Vector Machines transform the data from a relatively low dimension (6 in this case) to a higher dimension. A support vector classifier is found that separates the higher dimensional data into two classes. SVMs use kernel functions to find these support vector classifiers. For this research, three types of kernel functions are tested:

- **Polynomial Kernel** increases the dimension by changing the degree of the polynomial that maps the data, it then computes the relationships between each pair of samples, those relationships are used to find a classifier.
- **Gaussian or Radial Basis Function Kernel** finds classifiers in infinite dimensions using Taylor series. This kernel behaves like a weighted nearest neighbours function, thus the closest values to a new sample have a lot of influence on its classification.
- **Linear Kernel** assumes that the data is linearly separable and divides accidents from cycling with a hyperplane.

A visualisation of SVMs with different kernel functions with the training data set projected on the two largest eigenvalues is shown in figure 2.17. This figure shows the resulting decision boundaries using the three different kernels and highlights the corresponding support vectors. This figure also shows that the Gaussian and polynomial kernels have a better fit than the linear one since these more closely follow the contour of the oval accident point cloud. The soft margin size and scaling of the kernel functions are automatically found using the automatic hyperparameter optimization of MATLAB that minimizes five-fold cross-validation.

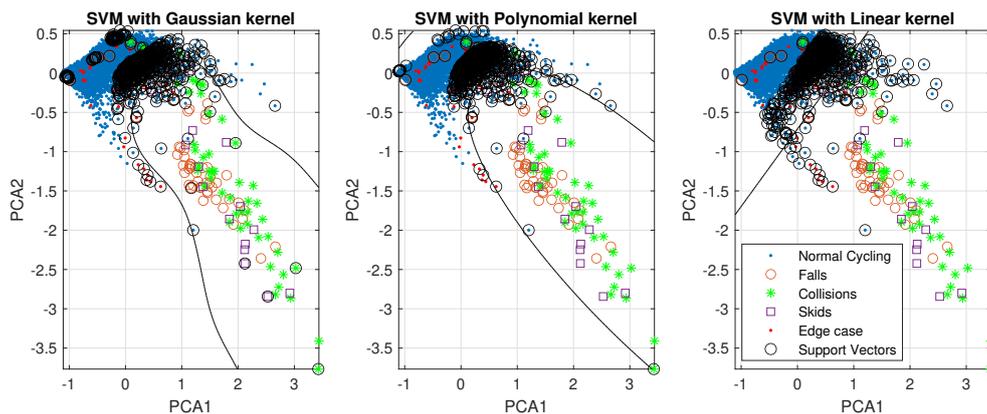


Figure 2.17: The decision boundary of three different SVMs using a Gaussian, polynomial or linear kernel fitted to the projected normalized samples of cycling, accidents, and edge cases on the two largest principal components PCA1 and PCA2.

2.6.3. K-Nearest Neighbours

A K-Nearest Neighbours (KNN) is a more intuitive classifier. This algorithm clusters the training data into cycling and accident clusters. A new sample will be allocated to the cluster it is the closest to. This is determined by looking at the K nearest samples for which different distance metrics can be used. Low values for K make the classifier sensitive to outliers while large values for K smooth over small clusters.

The number of neighbours K and distance metric are found using the automatic hyperparameter optimization of MATLAB that minimizes five-fold cross-validation. A KNN classifier is fitted on both $\mathbf{T}_{m \times n}$ and on the projected samples $\mathbf{P}_{m \times k}$, where $k = 6$ is the number of components.

2.6.4. Naive Bayes

A Naive Bayes (NB) classifier uses the probability of a certain value of a feature to determine if it is cycling or an accident. A Gaussian distribution is made for each feature in both classes. For a new sample: the initial guess of the probability of a sample being normal cycling is way larger than being an accident, these are the prior probabilities. The prior probability of normal cycling is multiplied with the likelihood of each feature given that it is normal cycling, based on these Gaussian distributions. The same is done with an initial guess of the sample being an accident. The class with the highest score determines the classification. To reduce the complexity of the distributions, the NB classifiers are trained on the normalized data $\mathbf{A}_{m \times n}$ and on the projected data $\mathbf{P}_{m \times k}$. The optimal distribution type and distribution smoothing parameter 'width' are automatically found using the automatic hyperparameter optimization of MATLAB that minimizes five-fold cross-validation.

2.6.5. Decision Tree

A Decision tree consists of a root node, nodes, and leaves. Each node has a threshold and makes a decision based on whether this threshold is passed or not. The leaf that resulted from a series of decisions determines the classification. A Decision tree is made in the following manner. Each feature is sorted from low to high. The value between every two adjacent samples (average) is calculated for each feature. The total Gini impurity for each of these averages is calculated with equation 2.17 and the lowest one determines the node. This process is repeated for the remaining samples of the most impure branch. To prevent overfitting the tree is pruned to a maximum of 7 nodes.

s = total number of samples in a node.

b = total number of samples in a branch.

a = number of accident samples in a branch.

c = number of cycling samples in a branch.

The Gini impurity I of a branch can be calculated using formula 2.16.

$$I = 1 - P(\text{accident})^2 - P(\text{cycling})^2 = 1 - \left(\frac{a}{b}\right)^2 - \left(\frac{c}{b}\right)^2 \quad (2.16)$$

The total Gini impurity I_{tot} is the weighted average of Gini impurities of the two branches that result from a node, this can be calculated with equation 2.17. The weight of a branch is: $weight = (n/s)$.

$$I_{tot} = \frac{b}{s} * I_1 + \frac{b}{s} * I_2 \quad (2.17)$$

3

Results

3.1. Performance of different classifiers in layer 1

The performance comparison of the different classifiers in layer 1 is done with data from separate accident test runs and normal cycling. The machine learning classifiers are trained on 80% of accidents and 25% of normal cycling data. To find the best classifier for layer 1, the remaining 20% of accidents and another 25% of normal cycling data are used for validation. A total of 83 different classifiers are trained and validated. These include the two threshold-based classifiers T1 and T2, the three types of SVM classifiers, two DT classifiers, two KNN classifiers, and two NB classifiers. All machine learning classifiers are trained on projected training data in 6, 12, and 2 dimensions using PCA with the three cost parameters: 20, 50, and 100. The DT, KNN, and NB classifiers are also trained on the non-reduced data with all 96 dimensions and the three cost parameters.

The performances of the three-layer algorithm with all these different classifiers substituted in layer 1 can be found in appendix F and the most promising results are visualized in figure 3.1. These are the most promising since they have the overall highest performance metrics. Figure 3.1 shows us that the specificity is generally high and above 99.5%. The sensitivity is also generally high and above 75%. The precision and sensitivity can only have a limited set of values since the comparison is done using only 11 accident test runs, which means that there can not be more than 11 false negatives or true positives.

Figure 3.1 also shows us that the system performs better with the T1 classifier than with the T2 classifier substituted in layer 1. The three SVM classifiers result in comparable sensitivity and specificity, but the Gaussian one has significantly higher precision. The decision tree classifiers result in comparable performance to T2. There is no significant difference in the performance of a decision tree using all 96 dimensions compared to using the reduced data using PCA. The KNN and NB classifiers perform better when using reduced data.

The best performing classifier in layer 1 is the one that results in the highest performance metrics. However, there is no classifier with the highest precision, sensitivity, and specificity. Thus best performing classifier is chosen to be the one that results in the highest precision and specificity, since these values include the false positive count. This is a KNN classifier that works with data projected on 6 principal components and is trained with a cost parameter of 20. This classifier uses 5 neighbours and a Minkowski distance metric. This classifier is used in the final accident detection system.

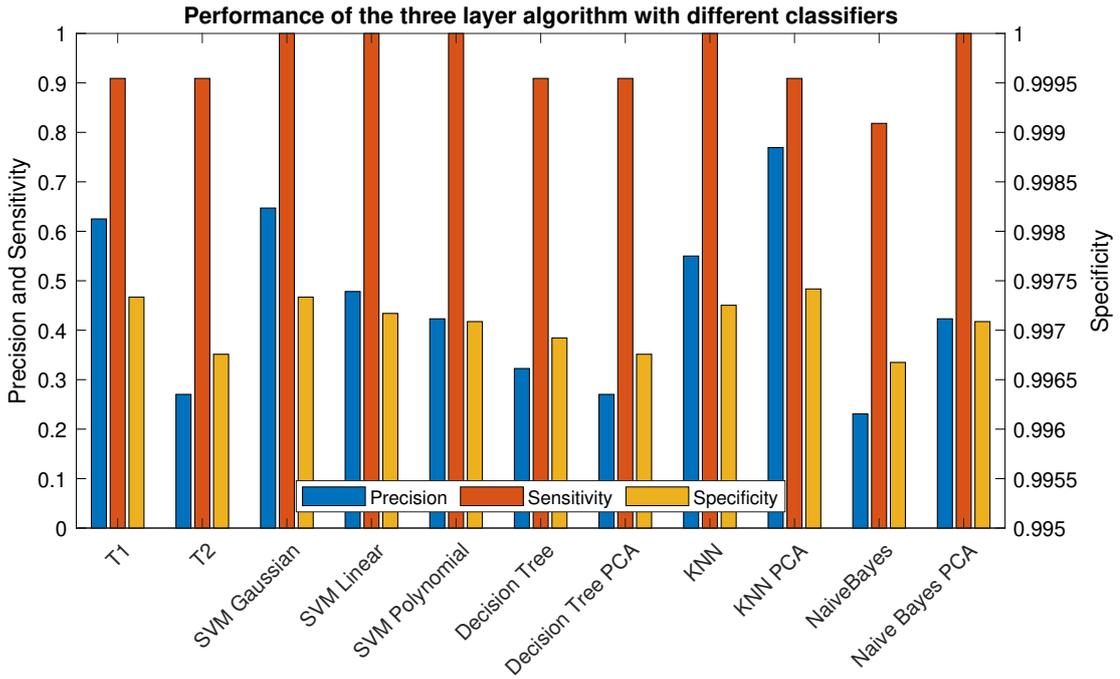


Figure 3.1: Performance comparison of the three-layer algorithm with 11 different classifiers in layer 1. The machine learning classifiers are trained on accidents with time shifts (0,-1, and -2 sec.), 6 PCA components, and a cost parameter of 20. The DT, KNN, and NB classifiers are also trained and validated on all 96 dimensions. Separate cycling and accident test runs are used for validation. Be aware of the different scale for specificity.

3.2. Performance of the system

The performance of the whole system is validated on the remaining 50% of normal cycling test runs. Some of these runs also contain a simulated accident during the test drive. A total of 8 accidents are simulated during these runs. These test runs all have at least 90% GPS coverage. The system correctly identified the location of 6 accidents, 2 accidents are missed, and 1 known standstill fall is correctly dismissed. Only 1 false warning is provided during an edge case at the end of a test run. Layer 1 passed 4 false accidents which are correctly dismissed by layer 2. This results in the confusion matrix in table 3.1 from which the following performance metrics can be determined:

- Precision: 85.7%
- Sensitivity: 75%
- Specificity: 99.997%

Confusion matrix		Samples from test runs	
		Accident	Cycling and edge cases
Warning send with location	Yes	6	1
	No	2	33313

Table 3.1: Confusion matrix of the accident detection and localisation system, validated on normal cycling test runs of which some include an accident.

An explanatory test run includes a standstill fall, walking the bike, placing the bike in a double bicycle storage, cycling, a simulated fall (comparable to F2), pulling the bicycle up, and placing it in storage. The data of this run is visualized in appendix C.2. The three layers of the algorithm are visualized in figure 3.2. This shows the measured accelerations during this test drive and the conclusions of each of the three layers. These conclusions are depicted as "yes" or "no" for each 3-second sample. The 'cycling detected' graph in figure 3.2 shows the conclusions of layer 0. This shows that cycling is detected during cycling and while walking the bike. This layer also detects cycling while picking the

bicycle up, since this also results in vibrations in the frame. The standstill fall (at $0.7 \cdot 10^5$ ms) is correctly discarded since the previous sample was not cycling. Placing the bicycle in and out of the upper layer of a double bicycle storage (at $1.4 \cdot 10^5$ ms and $1.7 \cdot 10^5$ ms) is measured as cycling due to the sufficiently large vibrations. Cycling and waiting to cross the road are detected correctly. The 'accident detected' and 'accident confirmed' graphs in figure 3.2 show the conclusions of layers 1 and 2. The accident (at $3.4 \cdot 10^5$ ms) is correctly detected. The following three samples correctly confirm the accident since the bicycle was lying on its side during this time. GPS was available with sufficient accuracy during the whole test run and the location of the accident is correctly displayed by the red dot in figure 3.3.

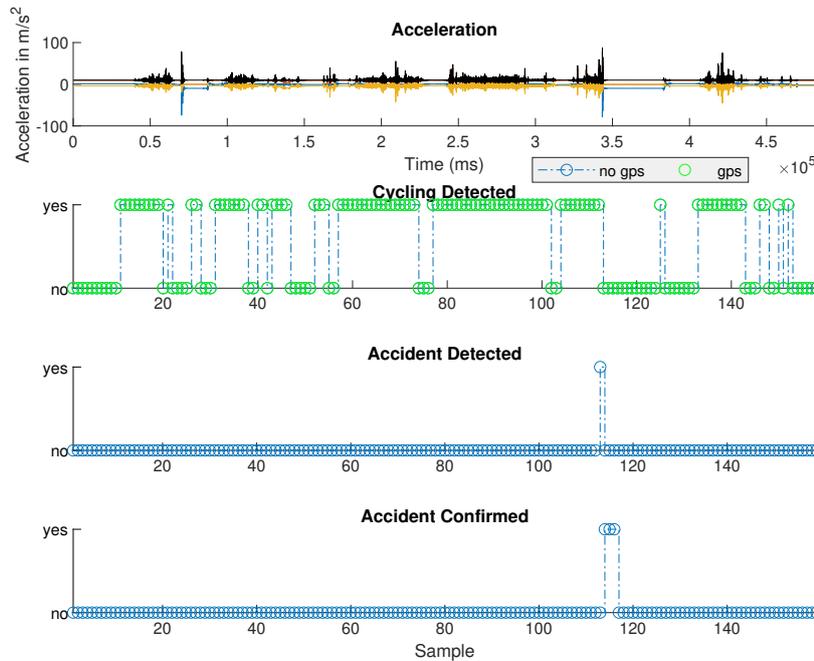


Figure 3.2: Results of the three layer algorithm on a test drive which includes cycling and falling. A green dot in the 'Cycling Detected' graph indicates that GPS is available for that sample.

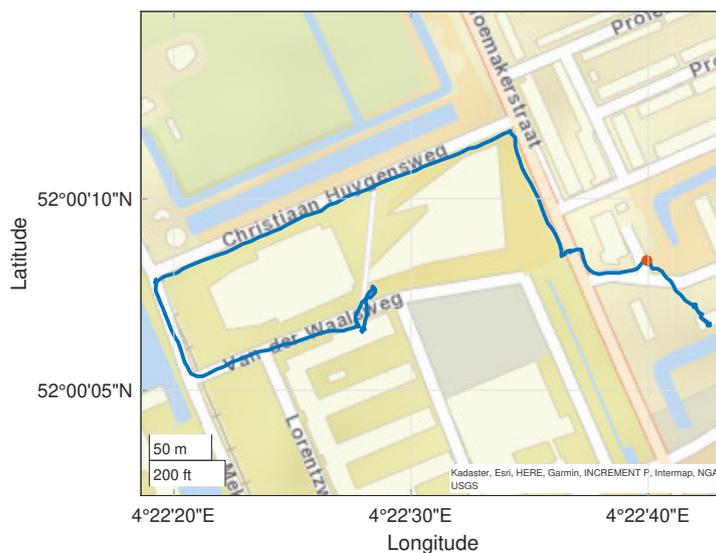


Figure 3.3: Test drive, the red dot shows the location of the accident

3.3. Sample Time

The system uses a sample time of 3 seconds since this is the current minimum sample time of the connected bicycle-server interaction of Royal Gazelle. The best performing classifier (KNN) is also trained and validated on samples with a sample time of 1,2,4, and 5 seconds. Again three time shifts are applied in the training phase, such that one is optimal, one least-optimal, and one in between these. The effects of sample time are displayed in figure 3.4. This shows that increasing the sample time generally increases the amount of missed accidents, but decreases the number of false warnings. Increasing the sample time from 4 to 5 seconds results in more false warnings and missed accidents. The difference in the number of false positives for a sample time of 3 seconds between figure 3.4 and 3.1 is due to the randomness of the training/validation test run distribution.

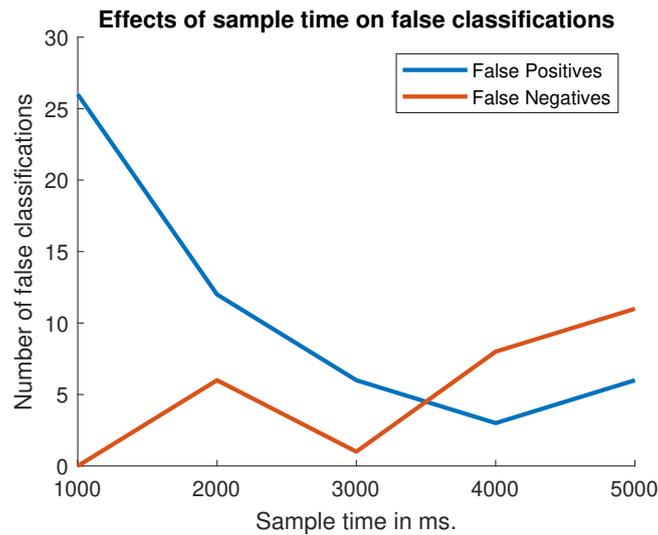


Figure 3.4: Performance metrics of the system with the best performing classifier (KNN) in layer 1 for different sample time lengths. Validation is done on 11 accidents, thus there is a maximum of 11 false negatives.

4

Practical Implementation

The current system of Royal Gazelle is capable of processing a data package containing information on the bicycle state every 3 seconds [27]. This package already contains wheel speed, longitude, and latitude. The first can be used in layers 0 and 1 as an indication of speed. The constraints from the system of Royal Gazelle shaped the development of **ALARM**. Two types of systems are proposed: the minimum viable product and a demonstration prototype.

4.1. Minimum Viable Product

The minimum viable product is a proposal for Royal Gazelle to add accident detection to their bicycles with the least amount of changes to the current setup. A briefing document to inform manufacturers how to implement such a system can be found in appendix D. This system uses the T2 threshold-based classifier to reduce complexity and data transmission costs, while not being sensitive to outliers.

A schematic of the whole system for the minimum viable product is shown in figure 4.1. The IoT-module in the bicycle receives GPS and IMU data for 3 seconds. The required features for T2 are calculated to form a sample and added to the existing data package. In addition to these features, the GPS speed and average acceleration in lateral direction need to be added to indicate movement and laying down respectively for layer 2. The package is sent over the GSM network to the server of Royal Gazelle. The server receives this package and runs the algorithm as proposed in section 2.4. If an accident is confirmed, an e-mail or SMS with the last known location is automatically sent to predefined contacts.

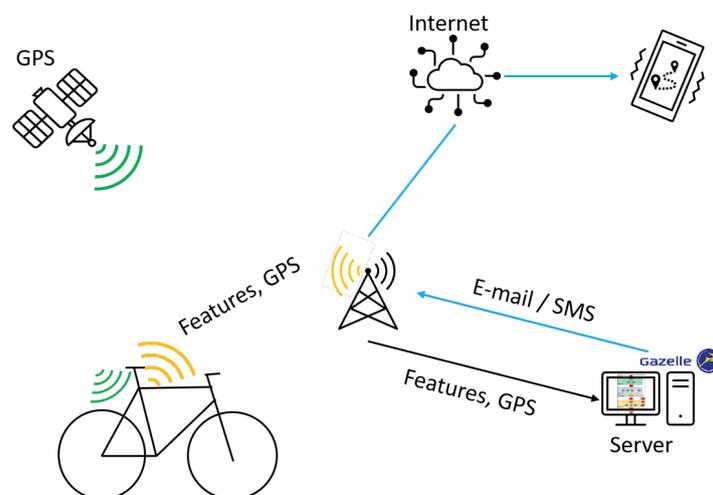


Figure 4.1: Schematic of the minimum viable product. This system utilizes the existing IoT-module and server capabilities from Royal Gazelle. The IoT module stores the IMU and GPS data and calculates the features for 3 seconds to form a sample. this sample is added to the existing outgoing data package which is received by a central server. This server runs the accident detection algorithm with the samples as input. If an accident is detected an SMS or e-mail is sent with the location of the accident.

4.2. Demonstration Prototype

A prototype system is developed for demonstration purposes. This system is schematically shown in figure 4.2 and consists of the following elements:

A smartphone rigidly mounted to the saddle pole, as shown in figure 2.2, collects and transmits IMU and GPS data in a comma-separated values (csv)-format using the User Datagram Protocol (UDP). This protocol uses a checksum as integrity verification but has no delivery guarantee since it only transmits. There are no retransmission delays, which makes it suitable for real-time applications such as this demonstration system. The transmitted data consists of accelerometer and gyroscope measurements at 100 Hz and GPS measurements at 100 Hz. These GPS measurements also include speed.

This data is transmitted to a laptop via a router in a local area network. The laptop runs a program that collects and temporarily stores the data every 3 seconds. The features are calculated and applied to the tree layer algorithm described in section 2.4 with the best performing classifier substituted in layer 1. The lean angle is based on the orientation values included in the IMU data.

When an accident is confirmed, an e-mail is automatically sent with the last known location of the smartphone using longitude and latitude from the GPS signal. This e-mail contains a link to google maps which displays this location.

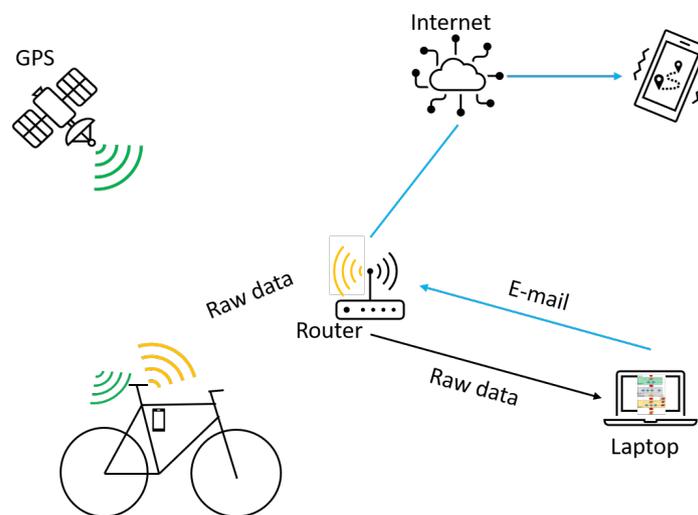


Figure 4.2: Schematic of the demonstration system which uses a smartphone mounted to the bicycle to collect and send GPS and IMU data. This data is sent over a local area network to a laptop. The laptop collects the data for 3 seconds and calculates the features from a sample. These samples are the input for the classification algorithm, which runs on the same laptop. An e-mail is sent with the GPS coordinates if an accident is confirmed.

The demonstration consists of the bicycle standing still, cycling on grass for 30 seconds, a stop-and-go, jumping off the bicycle, and pushing it to the ground (P2). An e-mail is automatically sent after the accident which contains a set of instructions and a link to google maps which displays the location of the accident.

5

Discussion

5.1. Performance

The best performing set of parameters for the accident detection contains a fitted KNN classifier on 6 PCA components and a cost parameter of 20. This means that the classifier considers misclassifying accidents as 20 times worse than misclassifying cycling to compensate for the unbalanced data during the training phase. The highest precision and specificity define the best performance in this case, since these metrics include the false positive count. Higher sensitivity is possible with different classifiers.

- **Precision:** From all warnings send, 85.7% are true accidents. This means that 14.3% of the warnings are false. This corresponds to one in seven. However, it should be noted that the only false warning was while storing the bicycle.

I used separate accident and cycling test runs to compare the performance of the different classifiers, this resulted in a precision of 76.9%. This is significantly higher than for the T1 and T2 classifiers that have a precision of 62.5% and 27.0% respectively. This is because the thresholds are determined using only 1 hour of normal cycling and limited edge cases, increasing this data-set would result in more precise thresholds. Higher thresholds will increase the precision. The main advantage of these classifiers is that developers can easily and intuitively tweak these values.

- **Sensitivity:** The system detects 75% of all accidents during the test drives with accidents. This value does not differ much for other classifiers in layer 1. This is because I only used a limited set of accident runs for validation which are all significantly different from normal cycling. These accident samples also include different time shifts from the same accident.
- **Specificity:** Of all 3-second normal cycling samples, the algorithm classifies 99.997% correctly. This means that a false warning is send approximately every $33314 * 3 = 99.942$ seconds ≈ 28 hours of cycling. Although it should be noted that the only resulting false warning resulted during an edge case of storing the bicycle. I compared the performance of the different classifiers in layer 1 on separate accident test runs and normal cycling which resulted in a specificity of 99.98%. An explanation for this difference is that these normal cycling test runs contained more edge cases which are more similar to accidents than normal cycling. The values for specificity are generally above 99% because I validated the algorithm on a lot of cycling samples which are all relatively similar. Still, I included a lot of cycling trips to capture as many edge cases and outliers as possible.

As stated before, the accident samples are manually labeled. These tags are the start time of the accident, used to capture as much of the accident as possible in a single sample. However, the algorithm receives consecutive samples, so the amount a single sample captures of an accident is dependent on the start time. The classifiers are trained on accident samples with a shifted start time

of -1 and -2 seconds to make them more generic. These shifted accident samples are of course from the same $N_{training} = \text{round}(54 * 80\%) = 43$ accidents, but these are still included because training on multiple time shifts makes the algorithm more sensitive. I validated the algorithms on the whole accident test runs, thus not on any cut accident samples. If the algorithm confirms an accident, it is considered a true positive.

5.2. Shortcomings

One major shortcoming in the accident test runs is the lack of crashes in the test runs. These accidents are more difficult to recreate due to safety concerns for the ones pushing the bicycle. Of all the accident types, crashes account for the minority of bicycle accidents that lead to admission in Dutch hospitals, but they account for a majority of fatal accidents. One might argue that if the system can detect the other less severe accidents, it would also be able to detect crashes. However, this should be validated. Also, the crash partner might be able to provide help after the accident, therefore automated accident detection and localisation would be less necessary for crashes.

It was not possible to recreate accidents with a rider. One can expect different kinematics when a person is sitting on the bike during an accident. To stabilize a bicycle, the rider will steer into the direction of a fall [22, 35], this active stabilisation is of course not possible on a bicycle without a rider. It is assumed during this research that pushing the bicycle against objects and to the ground is sufficient to collect accident data. However, this assumption is not confirmed. To collect more realistic accident data, one could use a stunt driver protected by padding [6] or conduct a large observational study to include accidents in real traffic scenarios.

5.3. Improvements

An accident generally lasts less than 3 seconds. Thus increasing the sample time leads to accident samples that include more cycling and post-accident data. This means that accident samples have more similarities with cycling samples, which will result in fewer warnings since this would make it more difficult to distinguish accidents from cycling. Fewer warnings lead to an increase of false negatives and a decrease of false positives. Thus increasing the sample time decreases both sensitivity and precision. Fewer accidents will be missed by the system when using a sample time of 1 second, however, this does increase the number of false warnings. One way to make the system more sensitive is to use overlapping samples, since this would increase the chance that the accident spans a whole sample.

The test setup only uses an IMU and GPS receiver for input signals. However, most e-bikes also have a hall effect wheel speed sensor. This can provide better speed indication in layers 0 and 2. The current system registers frame vibrations with an IMU during cycling and uses a threshold on the total acceleration as an indication of movement next to GPS because GPS is not always available. However, this introduces the following problem: When a bicycle falls during standstill, the fall could span two samples, which could provide a false warning. The first sample captures the start of the fall, which produces enough accelerations to pass the threshold. The second sample captures the end of the fall, which includes the ground contact. The second sample could be enough for the classifier to detect an accident. Such an accident would provide a confirmation in the second layer since the bicycle lies flat afterward. This problem would mitigate when substituting this threshold on acceleration with wheel speed measurements as an indication of speed.

Next to adding a wheel speed sensor, one could improve the system by applying the following:

- Adding a countdown and a button to prevent false warnings. When the algorithm confirms an accident, the bicycle could give a warning with a countdown, for example on the dashboard. The system only sends a message when the rider is unable to stop the countdown by pressing a button. The current e-bikes from Royal Gazelle cannot currently provide such a countdown, but the IoT-module can communicate with the user's smartphone via Bluetooth. An app could give a countdown as an extra confirmation in a third layer.

- Increase the computational resources of the IoT module. This would make it possible to run the classification algorithm on the IoT-module itself. When this is improved, the algorithm should only send a trigger with the location of the accident when needed, which decreases data plan costs and increases the privacy of the cyclist. This would increase privacy since the bicycle would only share its location in case of an emergency. The minimum viable product uses a threshold-based classifier due to computational restraints, this comes at the cost of lower precision and specificity than using the KNN classifier. A Bluetooth connection between the users' smartphone and the bicycle also provides a way to do the computations on the phone.
- Setting the sensitivity by shifting the decision boundary or increasing the thresholds. In this case, the rider could specify its driving style. For example, the system could be more sensitive for elderly riders who ride relatively slow over smooth cycling paths to capture more accidents. A younger cyclist who does not mind riding over a few bumps or curbs could set a low sensitivity to decrease the number of false warnings. The consumer could perhaps do this in an app, but it can also be set when acquiring the bicycle.

The first two would require the user's smartphone to be in working condition at the scene of the accident for such a system to work. A smartphone can be forgotten or break during an accident, which makes such a system less reliable. Another argument against using smartphones for the accident detection system is that the main demographic of Royal Gazelle is relatively old and does not always bring a smartphone with them [27, 37].

5.4. Implication of Research

Bicycle manufactures can apply **ALARM** to develop an integrated accident detection algorithm. These guidelines provide support for the development of a system that decreases the time between an accident and the arrival of help. This would provide the user a safe feeling and guarantee that emergency contacts are informed when needed. An accident detection system would have different effects on the following main consumer groups as described by Royal Gazelle [37]:

- **Conscious traditionalist:** cycles on more expensive low entry bicycles in rural areas. These consumers cycle on quiet back-roads where help is not always nearby. They are generally older which results in them having a higher risk of having an accident [43, 10, 33, 42]. This is the main consumer group of Royal Gazelle for which the main selling point would be a sense of independence and safety.
- **Pragmatic explorer:** cycles for environmental purposes and physical health. These consumers cycle a lot which exposes them to more dangerous scenarios which could result in accidents. Thus accident detection can provide a safer feeling for this consumer group.
- **Functional every-men:** cycles daily on a reliable bicycle but does not want to pay much. Accident detection might not be the best fit for this consumer group.
- **Ambitious adventurer:** cycles as a commute, is relatively young and urban. These consumers value technical gadgets and are early adopters. Thus accident detection utilizing smartphones as explained above could be a solution for this group.
- **Prestigious quality seeker:** cycles on expensive bicycles to radiate success. These consumers want the newest gadgets and features, which include accident detection.

6

Conclusion

I developed a method for the practical implementation of bicycle accident detection and geolocation in connected bicycles. Based on a comprehensive data set of cycling data and simulated accidents, it can be concluded that this method can effectively be applied by bicycle manufacturers to develop a machine learning based algorithm that utilizes the current technologies of connected bicycles. This method has a sample time of 3 seconds, and only uses an IMU and GPS/GNSS receiver as input to be compliant with existing hardware on connected bicycles. The resulting three-layer KNN-based algorithm distinguishes accidents from normal cycling and edge cases. This algorithm detects 75% of the accidents and detects normal bicycle usage correctly 99.997% of the time. From all warnings send, 85.7% are true accidents. A prototype system uses this algorithm and automatically sends the location of the bicycle to predefined contacts within 3 to 9 seconds after an accident. This prototype system uses a smartphone mounted to the bicycle frame that streams IMU and GPS data using UDP over a local network to a laptop as a substitute for the current IoT infrastructure.

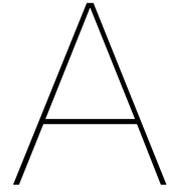
While the simulated accidents limit the inclusiveness of the results, the data-driven approach provides a way to include edge cases next to normal cycling for training and validation purposes. False warnings are inevitable when including edge cases, which makes the performance metrics more realistic. Future research should focus on gathering data from real accidents to validate the performance of the system on falls, collisions, skids, and crashes. Supplementing to previous research, this method is developed and validated using a large data set containing regular bicycle usage, edge cases, and three types of single bicycle accidents with constraints set by a bicycle manufacturer.

References

- [1] Bruno Aguiar et al. "Accelerometer-based fall detection for smartphones". In: *IEEE MeMeA 2014 - IEEE International Symposium on Medical Measurements and Applications, Proceedings*. IEEE Computer Society, 2014, pp. 1–6. ISBN: 9781479929207. DOI: 10.1109/MeMeA.2014.6860110.
- [2] Ferhat Attal et al. "The Powered Two Wheelers fall detection using Multivariate CUMulative SUM (MCUSUM) control charts". In: *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC*. Institute of Electrical and Electronics Engineers Inc., Nov. 2014, pp. 1280–1285. ISBN: 9781479960781. DOI: 10.1109/ITSC.2014.6957863.
- [3] Omar Aziz et al. "A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waist-mounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials". In: *Medical and Biological Engineering and Computing* 55.1 (Jan. 2017), pp. 45–55. ISSN: 17410444. DOI: 10.1007/s11517-016-1504-y.
- [4] Fahim Bin Basheer et al. "Design of accident detection and alert system for motor cycles". In: *c2013 IEEE Global Humanitarian Technology Conference*. 2013, pp. 85–89. ISBN: 9781479910953. DOI: 10.1109/GHTC-SAS.2013.6629894.
- [5] Ben Beck et al. "Bicycling crash characteristics: An in-depth crash investigation study". In: *Accident Analysis and Prevention* 96 (Nov. 2016), pp. 219–227. ISSN: 00014575. DOI: 10.1016/j.aap.2016.08.012.
- [6] Abderrahmane Boubezoul et al. "A simple fall detection algorithm for powered two wheelers". In: *Control Engineering Practice* 21.3 (Mar. 2013), pp. 286–297. ISSN: 09670661. DOI: 10.1016/j.conengprac.2012.10.009.
- [7] S Boufous et al. "Circumstances of on-road single-vehicle cyclist crashes in the Australian Capital Territory". In: *Proceedings, International Cycling Safety Conference*. 2014, pp. 1–9.
- [8] Alan K. Bourke et al. "Assessment of waist-worn tri-axial accelerometer based fall-detection algorithms using continuous unsupervised activities". In: *32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2010, pp. 2782–2785. ISBN: 9781424441235. DOI: 10.1109/IEMBS.2010.5626364.
- [9] S Candefjord et al. "Using Smartphones to Monitor Cycling and Automatically Detect Accidents-Towards eCall Functionality for Cyclists". In: *Proceedings, International Cycling Safety Conference*. 2014.
- [10] Christopher R Cherry and John H Macarthur. *E-bike safety. A review of Empirical European and North American Studies*. Tech. rep. 2019, pp. 1–13.
- [11] Ho-Rim Choi et al. "Evaluation of Algorithm for the Fall and Fall Direction Detection during Bike Riding". In: *International Journal of Control and Automation* 6.6 (Dec. 2013), pp. 209–218. ISSN: 20054297. DOI: 10.14257/ijca.2013.6.6.20.
- [12] David E. H. Jones. "The stability of the bicycle". In: *Physics Today* 23.4 (1970), pp. 34–40.
- [13] Fietsberaad. *Grip op enkelvoudige fietsongevallen, Fietsberaadpublicatie 19a*. Tech. rep. 2011.
- [14] Fietsberaad. *Grip op fietsongevallen met motorvoertuigen, Fietsberaadpublicatie 19b*. Tech. rep. 2011.
- [15] Elliot Fishman and Christopher Cherry. "E-bikes in the Mainstream: Reviewing a Decade of Research". In: *Transport Reviews* 36.1 (Jan. 2016), pp. 72–91. ISSN: 14645327. DOI: 10.1080/01441647.2015.1069907.
- [16] Gazelle and PON. *Requirements and testing - shared with PON*. 2-9-2020.
- [17] S. V. Georgakopoulos et al. "On-line fall detection via mobile accelerometer data". In: *IFIP Advances in Information and Communication Technology*. Vol. 458. Springer New York LLC, 2015, pp. 103–112. ISBN: 9783319238678. DOI: 10.1007/978-3-319-23868-5_8.

- [18] Weixi Gu et al. "BikeMate: Bike riding behavior monitoring with smartphones". In: *Proceedings of 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. Association for Computing Machinery, Nov. 2017, pp. 313–322. ISBN: 9781450353687. DOI: 10.1145/3144457.3144462.
- [19] Narelle Haworth and Ashim Kumar Debnath. "How similar are two-unit bicycle and motorcycle crashes?" In: *Accident Analysis and Prevention* 58 (2013), pp. 15–25. ISSN: 00014575. DOI: 10.1016/j.aap.2013.04.014.
- [20] J. Huijbers. "Een beschrijving van fietser- en bromfietserongevallen ten behoeve van prioriteitsindelingen bij het letselpreventie-onderzoek". In: *International IRCOBI Conference on the Biomechanics of Impacts, Delft* (1984).
- [21] Anice Jahanjoo, Marjan Naderan Tahan, and Mohammad Javad Rashti. "Accurate fall detection using 3-axis accelerometer sensor and MLF algorithm". In: *3rd International Conference on Pattern Analysis and Image Analysis, IPRIA 2017*. Institute of Electrical and Electronics Engineers Inc., July 2017, pp. 90–95. ISBN: 9781509064540. DOI: 10.1109/PRIA.2017.7983024.
- [22] J.D.G Kooijman, A.L. Schwab, and J.K. Moore. "Some observations on human control of a bicycle". In: 2009.
- [23] Simon Kozina et al. *Efficient Activity Recognition and Fall Detection Using Accelerometers*. Tech. rep. 2013, pp. 13–23.
- [24] J. Lowell and H. D. McKell. "The stability of bicycles". In: *American Journal of Physics* 50.12 (Dec. 1982), pp. 1106–1112. ISSN: 0002-9505. DOI: 10.1119/1.12893.
- [25] Paula Martiskainen et al. "Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines". In: *Applied Animal Behaviour Science* 119 (June 2009), pp. 32–38. ISSN: 01681591. DOI: 10.1016/j.applanim.2009.03.005.
- [26] J P Meijaard et al. "Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review". In: *Proceedings of the Royal Society A* 463.2084 (2007), pp. 1955–1982. ISSN: 11961198.
- [27] Bart Oor and Joris Kuiper. *Correspondance with Royal Gazelle during the run time of this project*. 9-11-2020/28-5-2021.
- [28] W Ormel, K Klein Wolt, and P den Hertog. *Enkelvoudige fietsongevallen*. Tech. rep. 2009.
- [29] The European Parliament and The Council of the European Union. *Decision No 585/2014/EU*. <https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:32014D0585>. 2014.
- [30] C Rodarius, J Mordaka, and T Versmissen Assignor. *Bicycle safety in bicycle to car accidents*. Tech. rep. 2008, pp. 1–48. URL: www.aprosys.com.
- [31] Paul Schepers and Karin Klein Wolt. *Single-bicycle crash types and characteristics*. Tech. rep. 2012, pp. 119–135. URL: www.wocref.org/cr.
- [32] Paul Schepers et al. "Bicycle fatalities: Trends in crashes with and without motor vehicles in The Netherlands". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 46 (Apr. 2017), pp. 491–499. ISSN: 13698478. DOI: 10.1016/j.trf.2016.05.007.
- [33] Paul ; Schepers, Karin ; Klein Wolt, and Elliot Fishman. "The safety of e-bikes in The Netherlands, International Transport Forum Discussion Paper". 2018. URL: <http://dx.doi.org/10.1787/21de1ffa-en>ThisVersionisavailableat:<http://hdl.handle.net/10419/194065>www.econstor.eu.
- [34] C C Schoon. *Botsingen van het type 'fietser-autofront'*. Tech. rep. 2004.
- [35] A.L. Schwab and J.P. Meijaard. "A review on bicycle dynamics and rider control". In: *Vehicle Systems Dynamics: International Journal of Vehicle Mechanics and Mobility* 51:7 (), pp. 1059–1090.
- [36] Donald Selmanaj, Matteo Corno, and Sergio M. Savaresi. "Hazard Detection for Motorcycles via Accelerometers: A Self-Organizing Map Approach". In: *IEEE Transactions on Cybernetics* 47.11 (Nov. 2017), pp. 3609–3620. ISSN: 21682267. DOI: 10.1109/TCYB.2016.2573321.
- [37] Linda Slob and Joris Kuiper. *Correspondance with Product Manager Digital Services of Royal Gazelle*. 11-3-2021.

- [38] Frank Sposaro and Gary Tyson. “iFall: An Android Application for Fall Monitoring and Response”. In: *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2009, pp. 6119–6122.
- [39] Fatemehsadat Tabei, Behnam Askarian, and Jo Woon Chong. “Accident Detection System for Bicycle Riders”. In: *IEEE Sensors Journal* (Sept. 2020), pp. 1–9. ISSN: 1530-437X. DOI: 10.1109/jсен.2020.3021652.
- [40] E Van Hassel, R De Lange, and H G M Mooi. *Bicyclist safety in bicycle to car accidents: an inventory study*. Tech. rep. 2006, pp. 1–24. URL: www.tno.nl.
- [41] L.T.B. Van Kampen and C.C. Schoon. *Tweewielerongevallen, Analyse van ongevallen-, letsel-en expositiegegevens voor het bepalen van prioriteiten voor nader onderzoek*. Tech. rep. 2002.
- [42] VeiligheidNL. *SEH-bezoeken door fietsongevallen*. Tech. rep. 2018. URL: www.veiligheid.nl/verkeer/feiten-cijfers.
- [43] Elke M.J. Verstappen et al. “Bicycle-related injuries in the emergency department: a comparison between E-bikes and conventional bicycles: a prospective observational study”. In: *European Journal of Trauma and Emergency Surgery* (2020). ISSN: 16153146. DOI: 10.1007/s00068-020-01366-5.



Protocol for the observational study of normal cycling and edge cases

This is a document with instructions that all participants were given for the observational study. In this study, the participants are asked to log IMU and GPS data during their daily cycling trips.

Observation protocol

Thank you for helping me with this observational study.

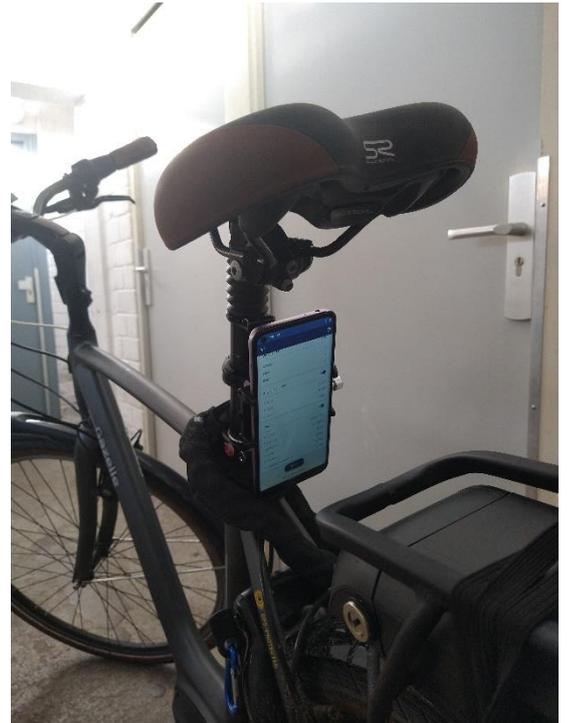
Setup

This study consists of a smartphone which measures and logs IMU (acceleration and gyroscopic) and GPS data connected to a bike. You will be asked to turn on these measurements every time you go for a bike ride.

Protocol

Once:

Mount the phone mount rigidly to the saddle pole so that the phone's screen is facing backwards and the phone is in an upright position. You could use the clamp spacers to make the assembly rigid (I've used one C-shaped spacer to make the clamp fixed).



Set up the MATLAB app:

Open the MATLAB app and log in with your TU Delft MATLAB licenced account. In the app, go to Sensors (using the menu in the upper left corner). Set to:

- Stream to: Log
- Sample rate: 100Hz
- More:
 - o Sensor Access: ON
 - o Acquire Data in Background: ON
 - o Set Auto Upload to WiFi
- Turn ON Sensor toggles:
 - o Acceleration Acceleration
 - o Angular velocity
 - o Position
- Leave the screen on!

The app should automatically upload the data once it is connected to your home Wi-Fi network. Thus make sure the phone can be connected to your home network. However, just to be extra sure you upload the files using the FILES app. Go to: Files > Internal storage > Documents > MATLAB > SensorLogs and select the runs you want to back-up. Click make Back-up on Google Drive.

Every time you go for a bike ride

First mount the phone to the bike and turn on the measurements on the MATLAB app (Start) before unlocking your bike. Make sure no button is pressed. Cycle normally like you usually do. Don't be hesitant to drive over bumps, make sharp turns, brake or accelerate harshly etc. but keep it safe! When you park your bike, first lock it before turning off the measurements and removing the phone.

You could leave the standard file name as is, but feel free to give your ride a characteristic name if you encounter something weird, especially if you were in an accident or fell over!

The goal of this study

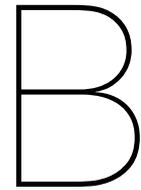
This observational study is aimed to verify the accident detection algorithms and classifiers I've developed. Normal cycling should result in a true negative (no accident) when your test run is provided to the algorithm. By having multiple people doing measurements, I not only gather more data, but also more diverse as everyone handles their bike differently. For example, I also have to get my bike from storage every time I use it, the algorithm should not detect an accident when I do this. That is why I want you to feel free to experiment yourself a bit, do drive up curbs and over bumps as long as it is safe and within the traffic regulations. Also: I'm interested in these 'edge cases' which occur when storing your bike for example. That is why it is more useful to gather multiple short trips than one long one, but the more data the better!

If you have any questions regarding this study, feel free to contact me at:

J.g.kuiper@student.tudelft.nl

Or: [REDACTED] (also on whatsapp)

Or on the element chat.



Test Runs

B.1. Cycling

This table shows the properties of all cycling test runs that are conducted during the project. These runs also contain edge cases that naturally occur when using a bicycle. The properties are:

- Name of the test run.
- Position of the phone on the bicycle. Left indicates that the measurements during this test run are done by a smartphone mounted to the left side of the top bar of the bicycle. Saddle indicates that the measurements during this test run are done by a smartphone mounted to the saddle pole as shown in figure 2.2.
- Frequency of the IMU measurements.
- Phone brand and model. Measurements are done by either a Lenovo K6 or Nokia 3.4. The first also has a magnetometer, from which the values are also logged and the latter has a GPS receiver.
- IMU measurements. Indicate whether the accelerations and angular velocities during these test runs are logged.
- Magnetometer measurements. Indicate whether the magnetometer values are logged. These are measurements of the magnetic field in three directions in μT and orientation values in degrees. Orientation consists of the azimuth (angle between the y-axis and the north pole), pitch and roll of the phone when lying flat.
- GPS indication. Indicates whether the GPS measurements are usable. Some test runs have partial GPS coverage.
- Time of the rest run in milliseconds. The length of some individual test runs is known. Test runs 'Damesfiets' to 'GB6' have a combined length of $9.6 * 10^7 ms$.
- Usage of test run. This explains for what the test run is being used. With:
 - PE = preliminary experiments
 - VC = validation classifiers
 - TC = training classifiers
 - VS = validation system
- Confirmed accidents. Shows how many accident are confirmed by the accident detection system during this test run.
- Notes. Shows some characteristics of the test run. Normal cycling in this case also includes edge cases that naturally occur when using a bicycle.

name	position	frequency	phone	IMU	Mag.	GPS	Time (ms)	Used	Conf.	notes
rondje	left	100	Lenovo K6	Yes	Yes	No		PE		normal cycling
Lage_vuursche	left	111	Lenovo K6	Yes	Yes	No		PE		normal cycling
delfshout19-1	left	111	Lenovo K6	Yes	Yes	No		PE		normal cycling
ah18-1	left	111	Lenovo K6	Yes	Yes	No		PE		normal cycling
ah18-1_terug	left	111	Lenovo K6	Yes	Yes	No		PE		normal cycling
ah19-1	left	111	Lenovo K6	Yes	Yes	No		PE		normal cycling
10Hz19-1	left	10	Lenovo K6	Yes	Yes	No		PE		normal cycling
50Hz19-1	left	50	Lenovo K6	Yes	Yes	No		PE		normal cycling
rondje10Hz22-1	left	10	Lenovo K6	Yes	Yes	No		PE		normal cycling
ah25	left	100	Lenovo K6	Yes	Yes	No		PE		normal cycling
ah26-1_terug	left	100	Lenovo K6	Yes	Yes	No		PE		normal cycling
rondje26-1	left	100	Lenovo K6	Yes	Yes	No	9,6E+07	PE		normal cycling
Damesfiets	saddle	100	Lenovo K6	Yes	Yes	No		VC		normal cycling
Amersfoor_D	saddle	100	Lenovo K6	Yes	Yes	No		VC		normal cycling
Noorderweg_berging	saddle	100	Lenovo K6	Yes	Yes	No		TC		normal cycling
Koninginnelaan1	saddle	100	Lenovo K6	Yes	Yes	No		VC		normal cycling
Koninginnelaan2	saddle	100	Lenovo K6	Yes	Yes	No		TC		normal cycling
GB1	saddle	100	Lenovo K6	Yes	Yes	No		VC		normal cycling
GB2	saddle	100	Lenovo K6	Yes	Yes	No		TC		normal cycling
GB3	saddle	100	Lenovo K6	Yes	Yes	No		VC		normal cycling
GB4	saddle	100	Lenovo K6	Yes	Yes	No		VC		normal cycling
AH1	saddle	100	Lenovo K6	Yes	Yes	No		TC		normal cycling
AH2	saddle	100	Lenovo K6	Yes	Yes	No		TC		normal cycling
AH3	saddle	100	Lenovo K6	Yes	Yes	No		TC		normal cycling
AH4	saddle	100	Lenovo K6	Yes	Yes	No		VC		normal cycling
centrum	saddle	100	Lenovo K6	Yes	Yes	No		TC		normal cycling
luuk1	saddle	100	Nokia3.4	Yes	No	Start only		VC		normal cycling
luuk2	saddle	100	Nokia3.4	Yes	No	Start only		TC		normal cycling
luuk3	saddle	100	Nokia3.4	Yes	No	Start only		TC		normal cycling
luuk4	saddle	100	Nokia3.4	Yes	No	Start only		VC		normal cycling
luuk5	saddle	100	Nokia3.4	Yes	No	Start only		VC		normal cycling
jason1	saddle	100	Nokia3.4	Yes	No	Start, middle		VC		Harsh riding!
jason2	saddle	100	Nokia3.4	Yes	No	Start, middle		VC		Harsh riding!
jason4	saddle	100	Nokia3.4	Yes	No	Start only		VC		Harsh riding!
jason5	saddle	100	Nokia3.4	Yes	No	Start only		TC		Harsh riding!
jason6	saddle	100	Nokia3.4	Yes	No	Start only		VC		Harsh riding!
omanaarlike	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
naarpluismeer	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
naaroma	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
ikenaarhuis	saddle	100	Nokia3.4	Yes	No	Yes, no end		TC		normal cycling
bosnaarhuis	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling

name	position	frequency	phone	IMU	Mag.	GPS	Time (ms)	Used	Conf.	notes
koen1	saddle	100	Nokia3.4	Yes	No	Start, end		TC		normal cycling
koen2	saddle	100	Nokia3.4	Yes	No	Start only		VC		normal cycling
koen3	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
koen4	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
koen5	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
koen6	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
koen7	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
koen8	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
koen9	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
koen10	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
jason7	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
jason8	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
jason9	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
jason10	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
arend1	saddle	100	Nokia3.4	Yes	No	Yes, no middle		VC		normal cycling
arend3	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
luuk14	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
luuk15	saddle	100	Nokia3.4	Yes	No	Yes		VC		bouncing bike at end
luuk16	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
luuk17	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
appie	saddle	100	Nokia3.4	Yes	No	Yes		TC		normal cycling
ahxl	saddle	100	Nokia3.4	Yes	No	Yes, no middle		VC		normal cycling
GB5	saddle	100	Nokia3.4	Yes	No	Yes		VC		normal cycling
GB6	saddle	100	Nokia3.4	Yes	No	Yes, no middle		VC		normal cycling
arend4	saddle	100	Nokia3.4	Yes	No	Yes	5593978	VS+TC	0	normal cycling
arend5	saddle	100	Nokia3.4	Yes	No	Yes	2087940	VS+TC	0	normal cycling
arend6	saddle	100	Nokia3.4	Yes	No	Yes	4095110	VS+VC	0	normal cycling
arend7	saddle	100	Nokia3.4	Yes	No	Yes	4830610	VS	0	contains 1 incident and 1 standstill fail
arend8	saddle	100	Nokia3.4	Yes	No	Yes	4146128	VS+VC	0	normal cycling
arend9	saddle	100	Nokia3.4	Yes	No	Yes	3208407	VS	2	contains 2 accidents
scheveningen1	saddle	100	Nokia3.4	Yes	No	Yes, no middle	3639799	VS+TC	0	normal cycling
scheveningen2	saddle	100	Nokia3.4	Yes	No	Yes, no middle and end	5020108	VS+TC	0	normal cycling
scheveningen3	saddle	100	Nokia3.4	Yes	No	Yes, no middle	3330918	VS+VC	0	normal cycling
scheveningen4	saddle	100	Nokia3.4	Yes	No	Yes, no end	4376043	VS+TC	0	normal cycling
arend10	saddle	100	Nokia3.4	Yes	No	Yes	4427709	VS	0	contains 1 incident
arend11	saddle	100	Nokia3.4	Yes	No	Yes	676377	VS	0	(one not confirmed)
arend12	saddle	100	Nokia3.4	Yes	No	Yes	4276070	VS	1	contains 1 incident
arend13	saddle	100	Nokia3.4	Yes	No	Yes	2549997	VS	1	does look like an accident
arend14	saddle	100	Nokia3.4	Yes	No	Yes	3605341	VS	0	normal cycling
arend15	saddle	100	Nokia3.4	Yes	No	Yes	2095603	VS	0	normal cycling
arend16	saddle	100	Nokia3.4	Yes	No	Yes	5685774	VS	1	contains 1 incident

name	position	frequency	phone	IMU	Mag.	GPS	Time (ms)	Used	Conf.	notes
arend17	saddle	100	Nokia3.4	Yes	No	Yes	3881757	VS	0	normal cycling
arend18	saddle	100	Nokia3.4	Yes	No	Yes	3804167	VS	0	normal cycling
arend19	saddle	100	Nokia3.4	Yes	No	Yes	4843776	VS	0	(one not confirmed)
arend20	saddle	100	Nokia3.4	Yes	No	Yes	5295968	VS	0	normal cycling
arend21	saddle	100	Nokia3.4	Yes	No	Yes	6257100	VS	0	normal cycling
arend22	saddle	100	Nokia3.4	Yes	No	Yes	6257100	VS	0	normal cycling
arend23	saddle	100	Nokia3.4	Yes	No	Yes	5472597	VS	1	incident
cyclingfall	saddle	100	Nokia3.4	Yes	No	Yes	482658	VS	1	contains 1 incident and 1 standstill fall
arend2	saddle	10	Nokia3.4	Yes	No	Yes		other		normal cycling
jason3	saddle	100	Nokia3.4	Yes	No	Start only		other		Harsh riding!
luuk6	saddle	10	Nokia3.4	Yes	No	Yes, no end		other		normal cycling
luuk7	saddle	10	Nokia3.4	Yes	No	Yes, no end		other		normal cycling
luuk8	saddle	10	Nokia3.4	Yes	No	Yes, no end		other		normal cycling
luuk9	saddle	10	Nokia3.4	Yes	No	Yes, no end		other		normal cycling
luuk10	saddle	10	Nokia3.4	Yes	No	Yes, no end		other		normal cycling
luuk11	saddle	10	Nokia3.4	Yes	No	Yes, no end		other		normal cycling
luuk12	saddle	10	Nokia3.4	Yes	No	No		other		normal cycling
luuk13	saddle	10	Nokia3.4	Yes	No	Yes, no end		other		normal cycling
luuk18	saddle	10	Nokia3.4	Yes	No	Yes		other		normal cycling

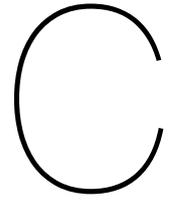
B.2. Accidents

This table shows the properties of all accident test runs that are conducted during the project. These runs consist of pushing the bicycle into different objects as explained in section 2.2.2. The properties are

- Name of the test run.
- Velocity of the bicycle when let go.
- Ground surface that the accident simulation is conducted on.
- Object where the bicycle is pushed into. These can also be forces on the bicycle without an object, for example: pushing the bicycle to the side or rotating the frame around the z-axis after letting the bicycle go.
- Position of the phone on the bicycle. Saddle indicates that the measurements during this test run are done by a smartphone mounted to the saddle pole as shown in figure 2.2.
- Frequency of the IMU measurements.
- Phone brand and model. Measurements are done by either a Lenovo K6 or Nokia 3.4. The first also has a magnetometer, from which the values are also logged and the latter has a GPS receiver.
- IMU measurements. Indicate whether the accelerations and angular velocities during these test runs are logged.
- Magnetometer measurements. Indicate whether the magnetometer values are logged. These are measurements of the magnetic field in three directions in μT and orientation values in degrees. Orientation consists of the azimuth (angle between the y-axis and the north pole), pitch and roll of the phone when lying flat.
- GPS indication. Indicates whether the GPS measurements are usable.
- Filmed indication. Some test runs have been filmed to be reproduced.
- Usage of test run. This explains for what the test run is being used. With:
 - PE = preliminary experiments
 - VC = validation classifiers
 - TC = training classifiers
 - VS = validation system

name	velocity	ground	object	position	freq.	phone	IMU	Mag.	GPS	filmed	Used
Fall_grass_v0_1	0	grass	none	left	111	Lenovo K6	Yes	Yes	No	no	PE
Fall_grass_v0_2	0	grass	none	left	111	Lenovo K6	Yes	Yes	No	no	PE
Fall_grass_v0_3	0	grass	none	left	111	Lenovo K6	Yes	Yes	No	no	PE
Fall_grass_v0_4	0	grass	none	left	111	Lenovo K6	Yes	Yes	No	no	PE
Fall_stone_v0_1	0	stone	none	left	111	Lenovo K6	Yes	Yes	No	no	PE
Fall_grass_v11_1	11	grass	none	left	111	Lenovo K6	Yes	Yes	No	no	PE
Fall_grass_v12_1	12	grass	none	left	111	Lenovo K6	Yes	Yes	No	no	PE
val10Hz_v0	0	grass	none	left	10	Lenovo K6	Yes	Yes	No	no	PE
val50Hz_v0	0	grass	none	left	50	Lenovo K6	Yes	Yes	No	no	PE
pushval10Hz_v12_metduw	12	grass	side push	left	10	Lenovo K6	Yes	Yes	No	no	PE
val50Hz_v11_metduw	11	grass	side push	left	50	Lenovo K6	Yes	Yes	No	no	PE
F1	0	grass	none	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
F2	5	grass	none	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
F3	12	grass	none	saddle	111	Lenovo K6	Yes	Yes	No	no	VC
F4	22	grass	none	saddle	111	Lenovo K6	Yes	Yes	No	yes	VC
F5	22	grass	none	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
P1	0	grass	none	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
P2	5	grass	spush side	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
P3	12	grass	spush side	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
P4	22	grass	spush side	saddle	111	Lenovo K6	Yes	Yes	No	yes	VC
P5	22	grass	spush side	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
C1	5	grass	curb (90deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	VC
C2	12	grass	curb (90deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
C3	22	grass	curb (90deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
C4	22	grass	curb (90deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
C5	22	grass	curb (90deg)	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
S1	5	grass	curb (45deg)	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
S2	12	grass	curb (45deg)	saddle	111	Lenovo K6	Yes	Yes	No	no	VC
S3	22	grass	curb (45deg)	saddle	111	Lenovo K6	Yes	Yes	No	no	VC
S4	22	grass	curb (45deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	VC
G1	5	grass	curb (15deg)	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
G2	12	grass	curb (15deg)	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
G3	55	grass	curb (15deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
G4	55	grass	curb (15deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
W1	5	grass	wall (90deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC

name	velocity	ground	object	position	freq.	phone	IMU	Mag.	GPS	filmed	Used
W2	12	grass	wall (90deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	VC
W3	22	grass	wall (90deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
W4	22	grass	wall (90deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
H1	5	grass	wall (45deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
H2	12	grass	wall (45deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
H3	22	grass	wall (45deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
H4	22	grass	wall (45deg)	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
T1	5	grass	pull steer	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
T2	12	grass	pull steer	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
T3	22	grass	pull steer	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
T4	22	grass	pull steer	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
Q1	5	grass	pull frame	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
Q2	12	grass	pull frame	saddle	111	Lenovo K6	Yes	Yes	No	no	VC
Q3	22	grass	pull frame	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
Q4	22	grass	pull frame	saddle	111	Lenovo K6	Yes	Yes	No	yes	VC
R1		grass	unknown	saddle	111	Lenovo K6	Yes	Yes	No	yes	other
R3		grass	unknown	saddle	111	Lenovo K6	Yes	Yes	No	no	other
RH1		grass	unknown	saddle	111	Lenovo K6	Yes	Yes	No	yes	other
RH2		grass	unknown	saddle	111	Lenovo K6	Yes	Yes	No	yes	other
RH3		grass	unknown	saddle	111	Lenovo K6	Yes	Yes	No	yes	other
D1	5	stone snow	none	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D2	12	stone snow	none	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D3	20	stone snow	none	saddle	111	Lenovo K6	Yes	Yes	No	no	VC
D4	20	stone snow	none	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D5	5	grass snow	push side	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D6	12	grass snow	push side	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D7	20	grass snow	push side	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D8	20	grass snow	push side	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D9	5	stone snow	push side	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D10	12	stone snow	push side	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D11	20	stone snow	push side	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D12	20	snow	unknown	saddle	111	Lenovo K6	Yes	Yes	No	yes	TC
D13	5	stone snow	rotate frame	saddle	111	Lenovo K6	Yes	Yes	No	no	VC
D14	12	stone snow	rotate frame	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
D15	20	stone snow	rotate frame	saddle	111	Lenovo K6	Yes	Yes	No	no	TC
R4	?	stone snow	push side	saddle	111	Lenovo K6	Yes	Yes	No	no	other
R5	?	stone snow	push side	saddle	111	Lenovo K6	Yes	Yes	No	no	other



Visualisation of various test runs

The following figures show a visualisation of some explanatory test runs. The figures show the accelerations and angular velocities during the run in three directions and the total values. From these signals, 10 features are calculated and shown in the lower right corner. The signals resulting from the magnetometer are shown in the upper right corner, these are not used in the accident detection algorithm but are logged in case this data-set will be used for other research. The test run 'cyclingfall' does not have magnetometer values, but does have GPS data which is shown in a map in the upper right corner.

It can be seen that the total acceleration produces high peaks during normal cycling in the 'centrum' test run. The peaks in the total acceleration during the accidents have comparable values. Thus it can already be concluded that a threshold on this signal would not be sufficient for an accident detection algorithm. The angular velocities during the accidents are significantly higher than during normal cycling.

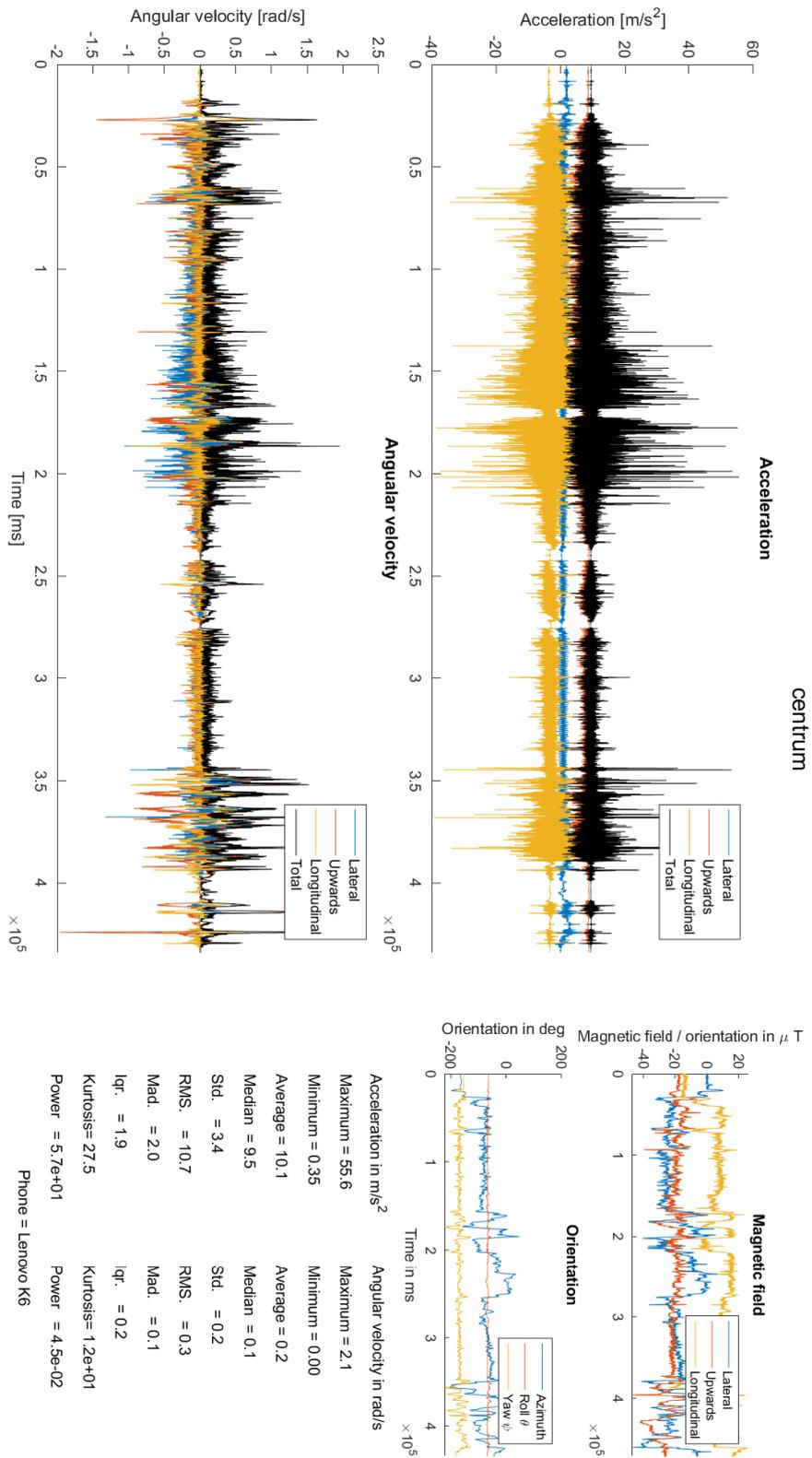


Figure C.1: Visualisation of a test run in the centre of Delft which includes normal cycling and various edge cases.

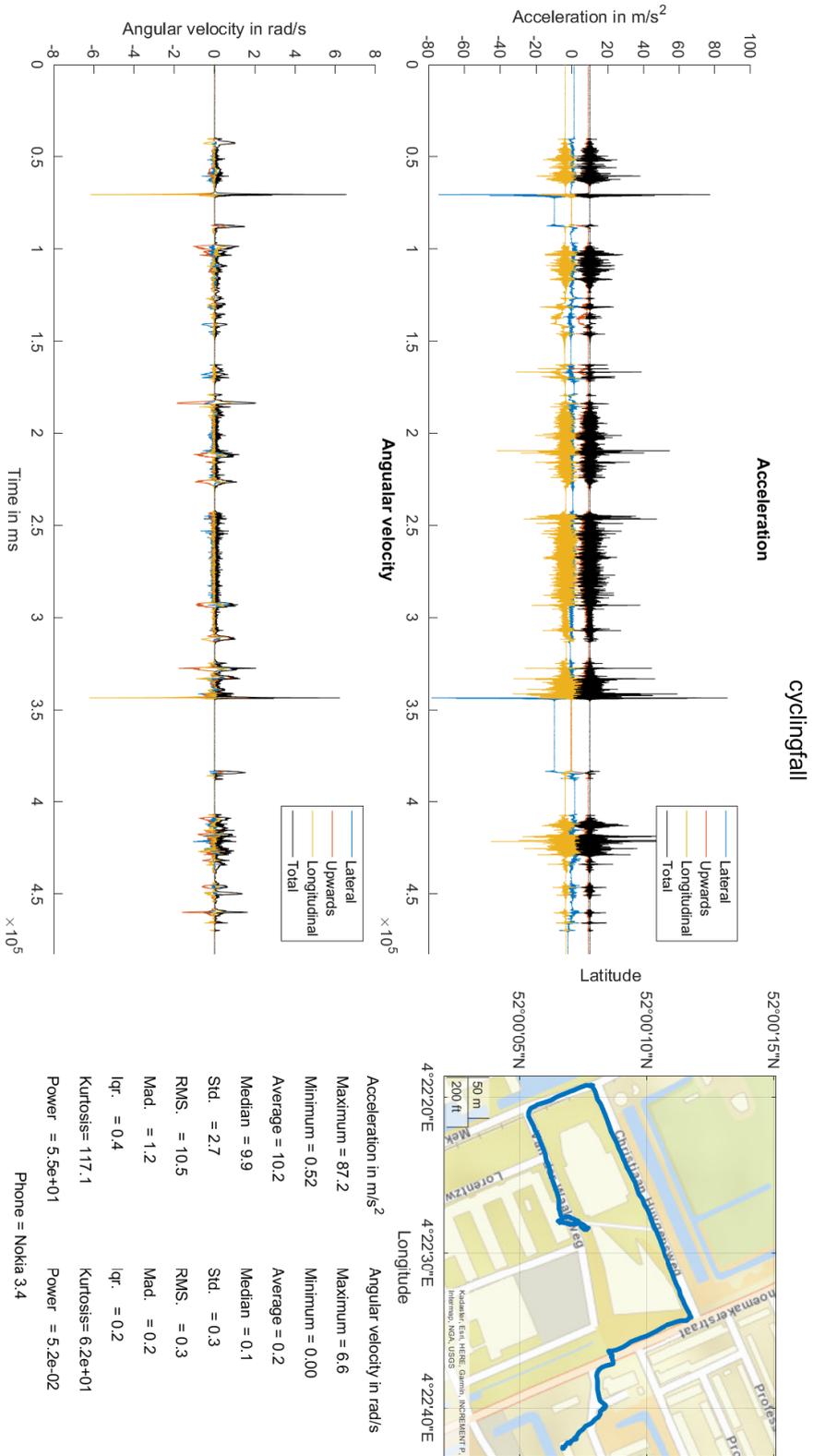


Figure C.2: Visualisation of a test run which includes a test run which includes an accident.

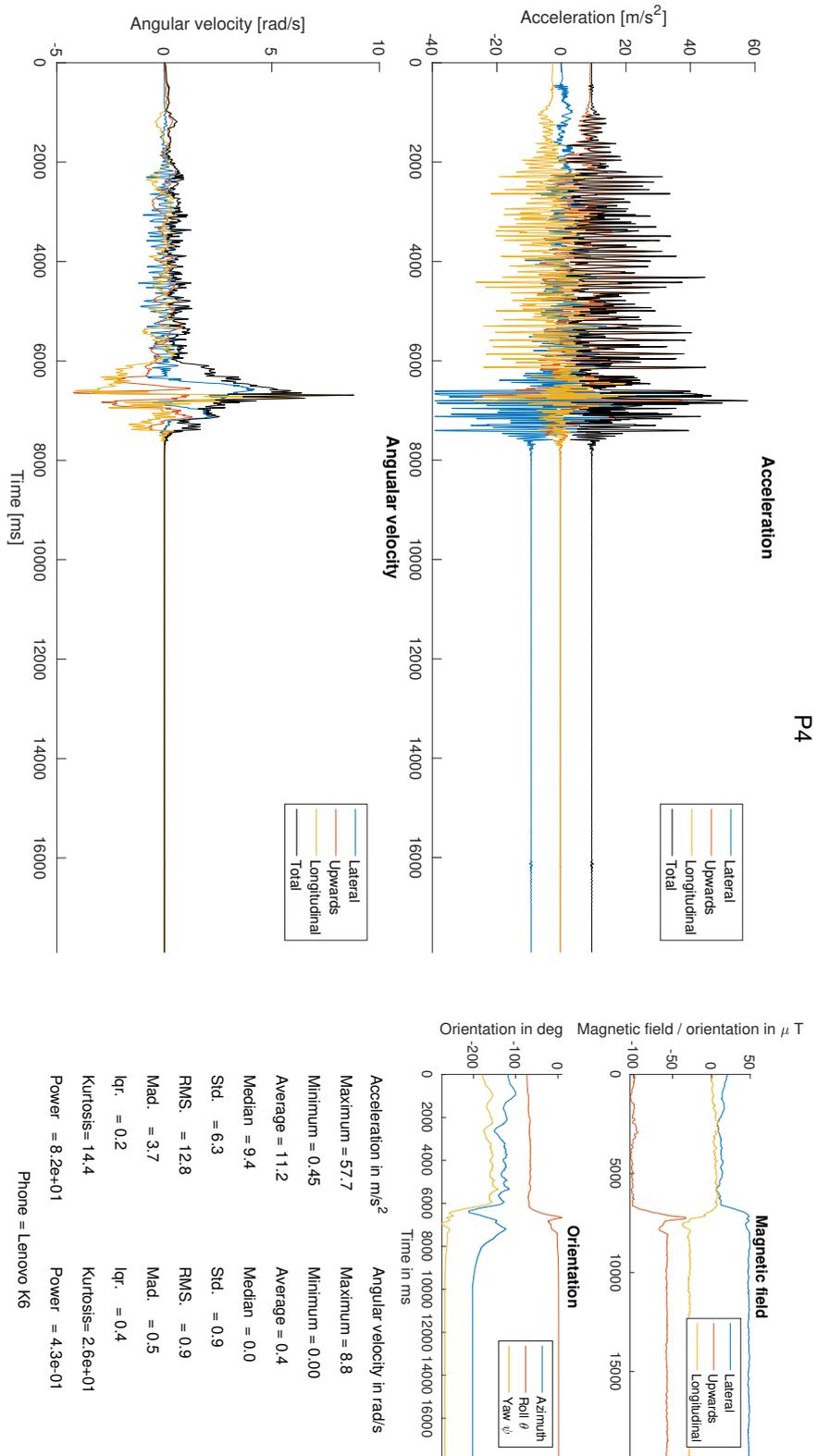


Figure C.3: Visualisation of an accident test run P4.

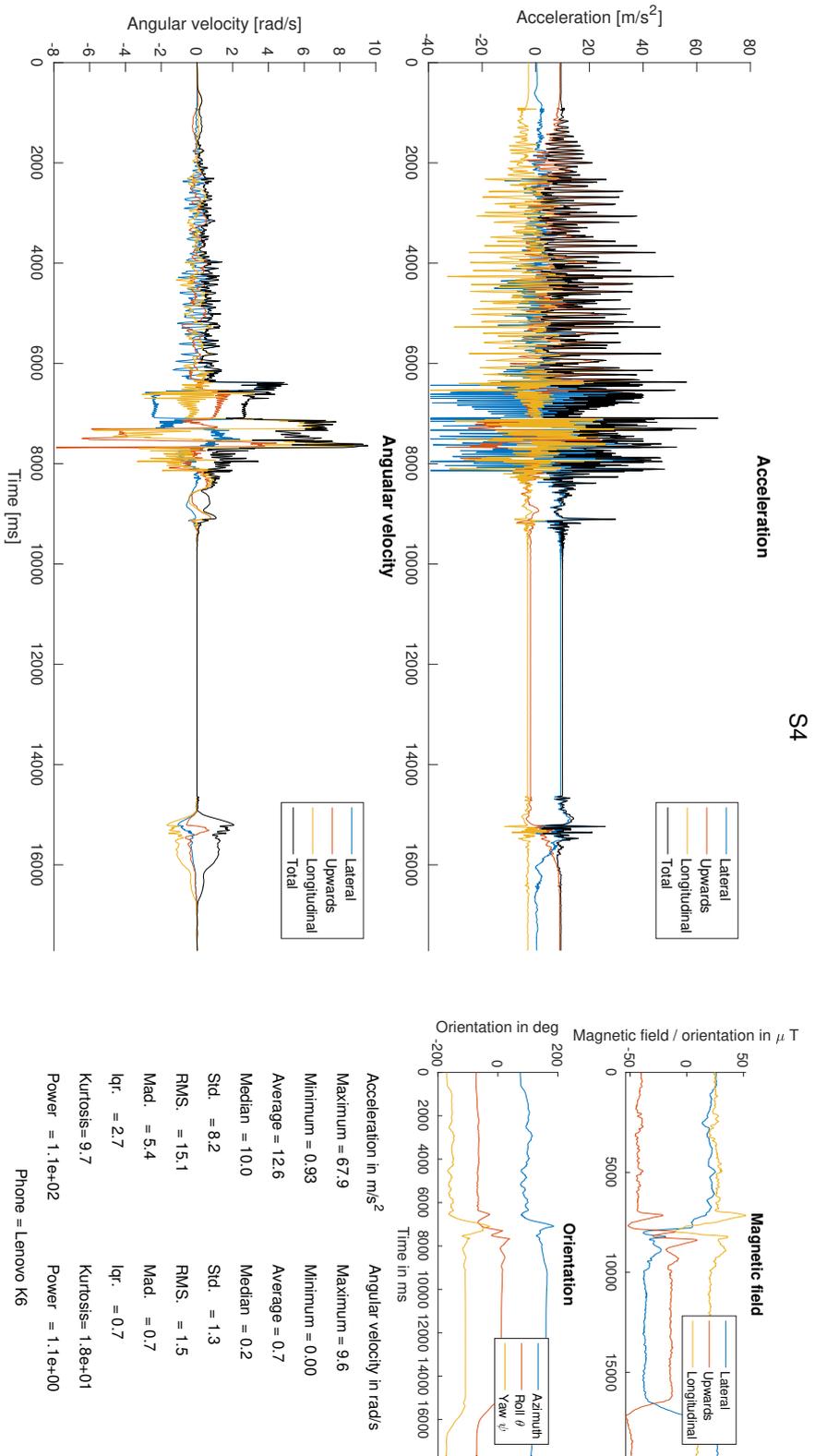


Figure C.4: Visualisation of an accident test run S4.

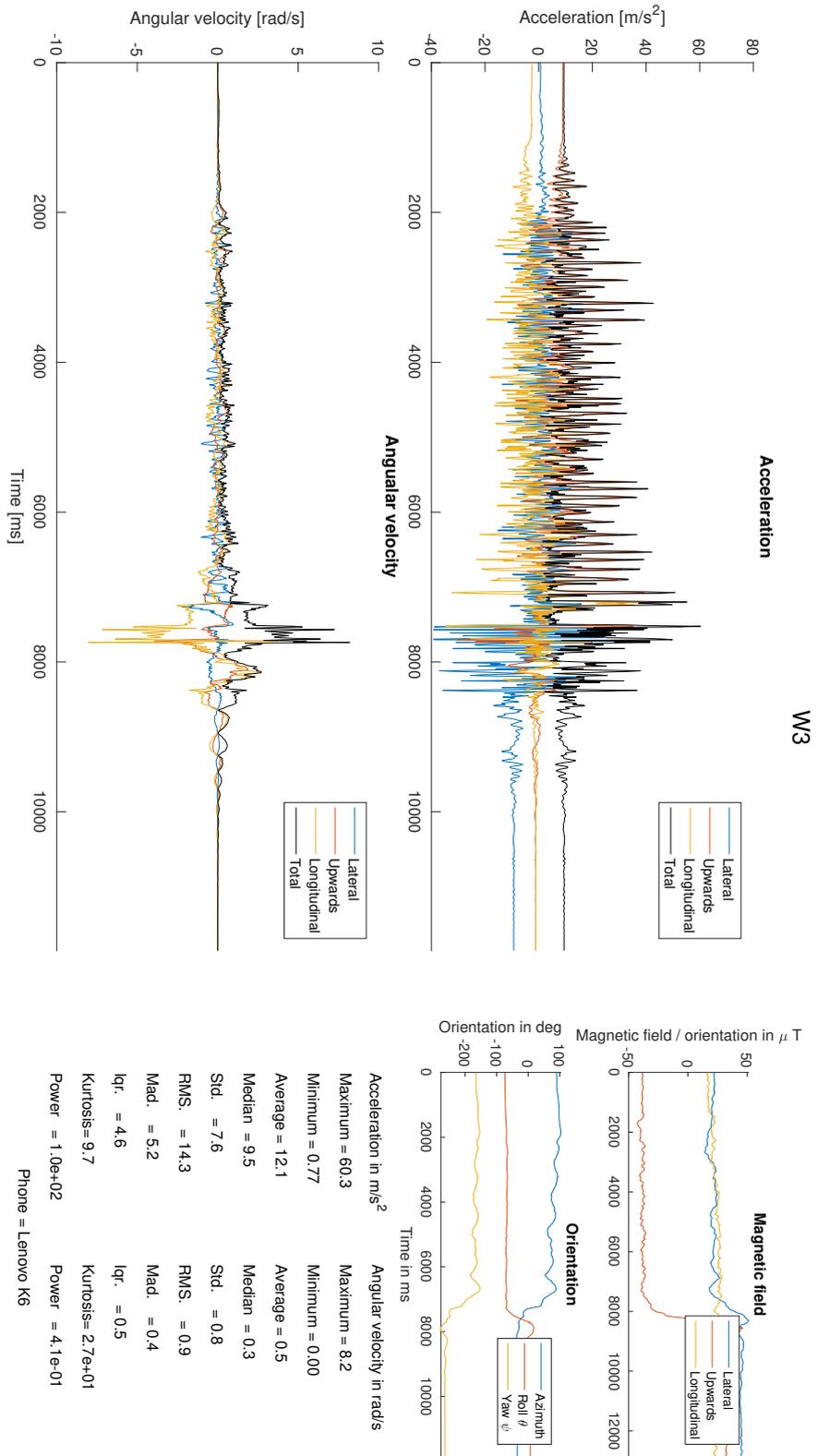


Figure C.5: Visualisation of an accident test run W3.

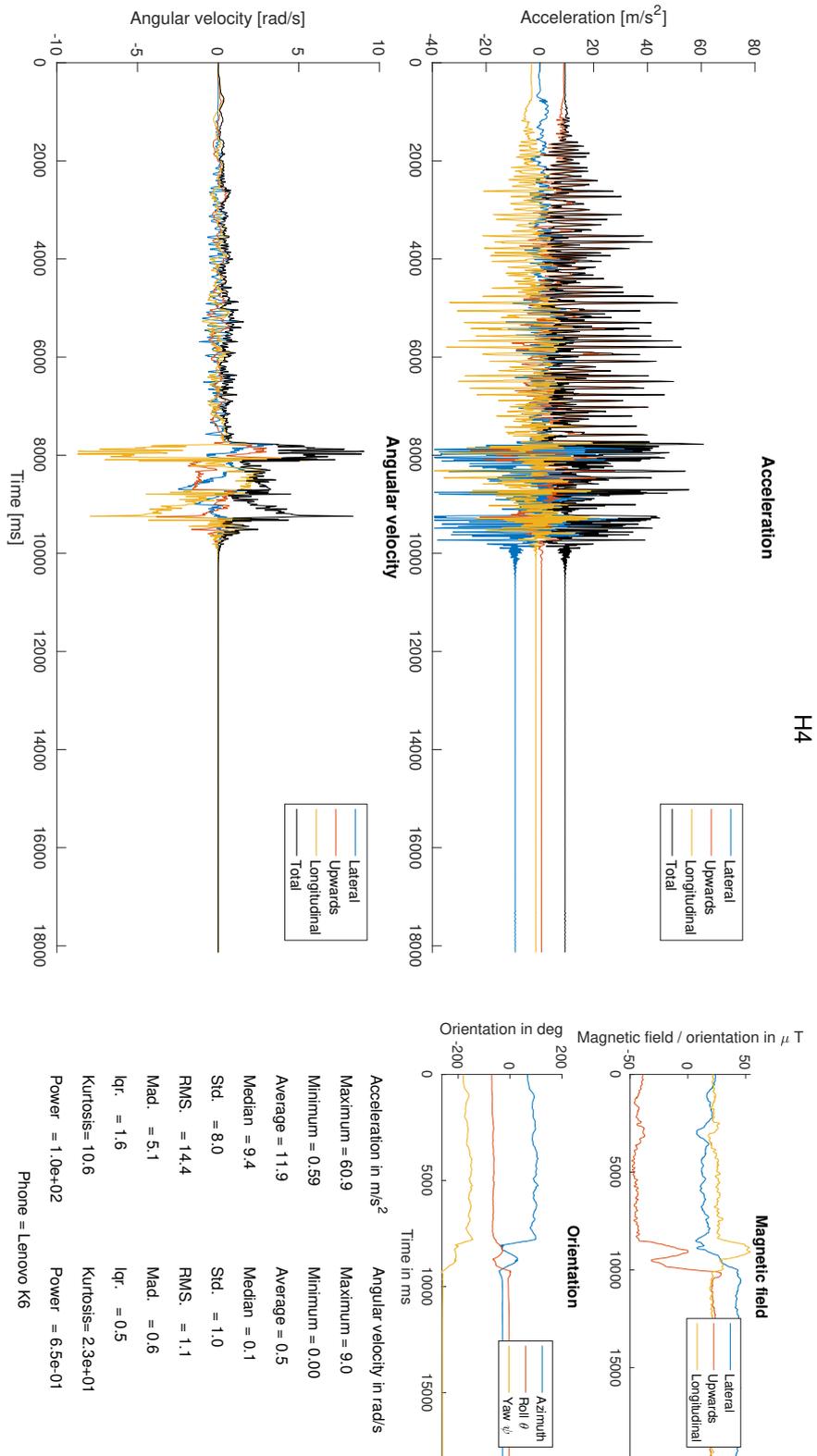


Figure C.6: Visualisation of an accident test run H4.

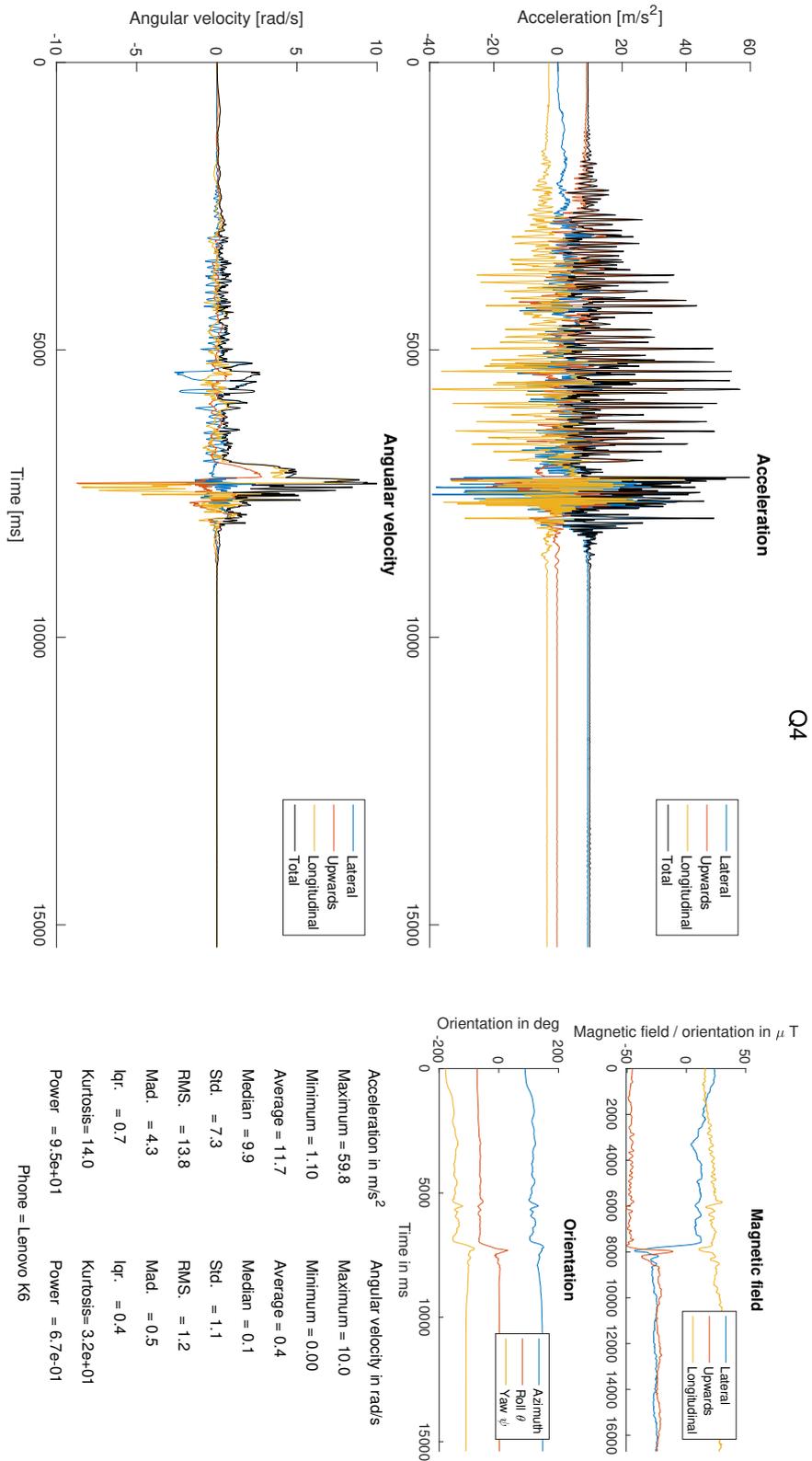


Figure C.7: Visualisation of an accident test run Q4.

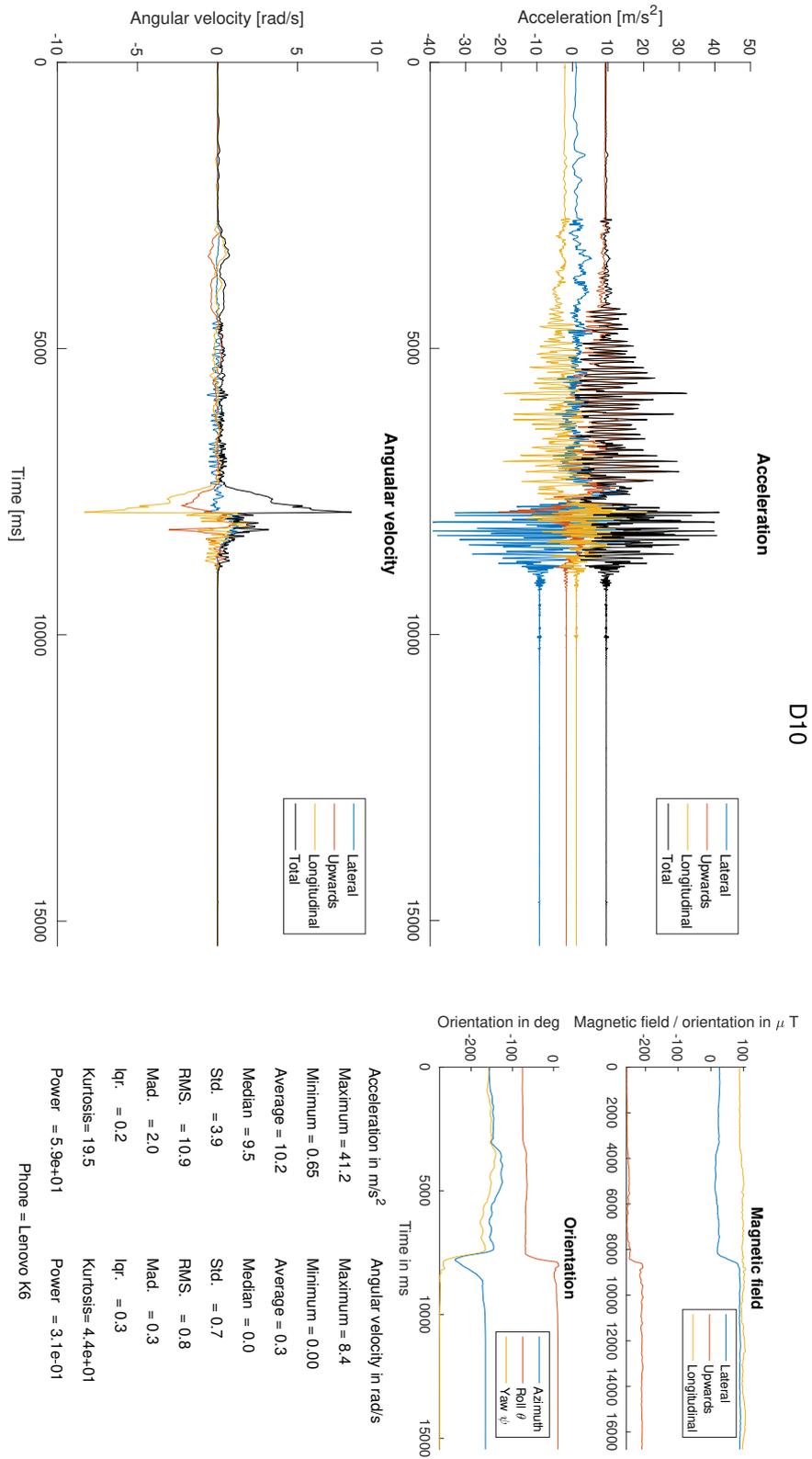
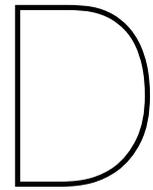


Figure C.8: Visualisation of an accident test run D10.



Briefing Document for Minimum Viable Product

A briefing document is written to inform bicycle manufacturers like Royal Gazelle on how to implement an accident detection system into existing 'connected' bicycles. This document is as follows:

Brief: Implementing a Minimum Viable Accident Detection on Connected E-bikes

The purpose of this brief is to provide manufacturers of connected e-bikes with an update regarding implementing the new integrated accident detection system.

Current status

The prototype system is trained and validated using an external sensor setup mounted under the saddle of the bicycle. Data is collected during 71 artificial accidents and 54 hours of normal cycling. This setup streams the IMU based accelerations and angular velocity as well as GNSS/GPS based longitude, latitude and velocity using UDP. A laptop reads this data stream and runs a classification algorithm every 3 seconds. Once an accident is detected, an e-mail is send with the location of the bicycle.

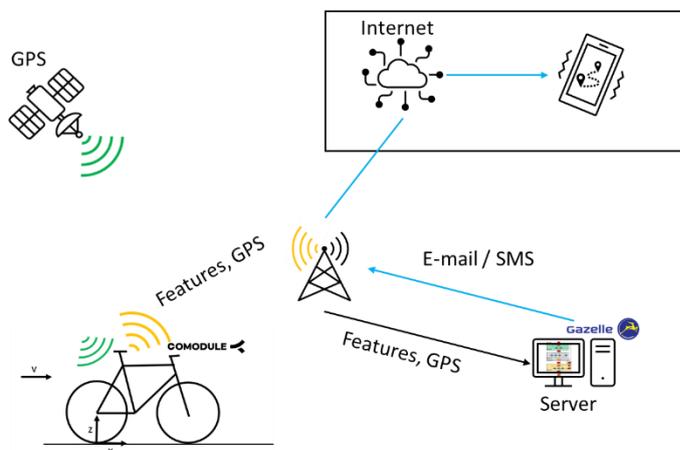


Figure 1 Minimum viable accident detection system

A minimum viable accident detection system would consist of: (1) An IoT module capable of sampling acceleration and angular velocity with 100Hz and GPS location with 1Hz, calculating simple features of 3 second samples and send these over the cloud with a GSM connection. (2) A server that receives these features and runs a classification algorithm. (3) A E-mail or SMS service to send the last known location to predefined contacts.

In the minimum viable product, the features of the IMU data will be calculated on the IoT module. These features will be added to the existing outgoing data stream which is send every 3 seconds to a server. This server receives this data stream and runs the accident detection algorithm.

Key minimum requirements

Data collection in embedded system

- IMU data should be measured with a frequency of 100 Hz and minimum resolution of 0.0001 m/s^2 and 0.0001 rad/s.
- GPS data should be measured with a frequency of 1Hz and minimum resolution of 0.0001 deg. and 0.01 m/s.
- The wheel speed should be measured with a frequency of 100Hz and a minimum resolution of 0.1 m/s .

Feature extraction in embedded system (approximately 1800 FLOPs)

This need to store 1315 floating numbers for the calculation, for a 32-bit system this will be 5.26KB.

- Features of the IMU and GPS data should be calculated for a sample time of 3 sec.
- Mean absolute deviation of acceleration in lateral direction (a_{lat}).

$$mad(a_{lat}) = \frac{1}{n} \sum_{i=1}^n |a_{lat_i} - \overline{a_{lat}}|, \quad \overline{a_{lat}} = \frac{1}{n} \sum_{i=1}^n a_{lat_i}$$

- Mean absolute deviation of total angular velocity (R_{tot}).

$$mad(R_{tot}) = \frac{1}{n} \sum_{i=1}^n |R_{tot_i} - \overline{R_{tot}}|, \quad \overline{R_{tot}} = \frac{1}{n} \sum_{i=1}^n R_{tot_i},$$

$$R_{tot} = \sqrt{R_x^2 + R_y^2 + R_z^2}$$

- Standard deviation acceleration in lateral direction (a_{lat}).

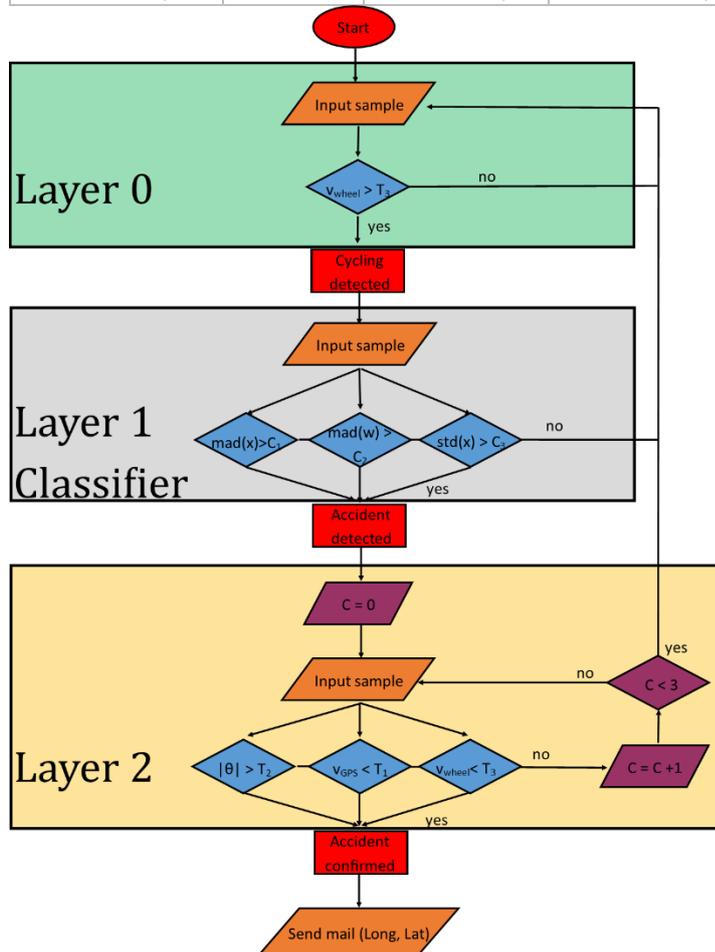
$$std(a_{lat}) = \sqrt{\frac{\sum_{i=1}^n (a_{lat_i} - \overline{a_{lat}})^2}{n - 1}}$$

- Mean lean angle with 0 is upright.
- Compute the mean longitude, latitude and speed of the 3 GNSS/GPS measurements.
- Calculate the mean velocity from wheel speed.

Data classification in the cloud

- Use the proposed 3 layer algorithm as shown in the flow diagram and tune thresholds starting with:

T1 = 1.38 m/s	T2 = 60°	T3 = 1.5 m/s	C1 = 4.25 m/s ²	C2 = 0.65 rad/s	C3 = 6 m/s ²
---------------	----------	--------------	----------------------------	-----------------	-------------------------



Algorithm

An example threshold based classifier works as follows:

```
FUNCTION classifier INPUT features
    IF mad lateral acceleration is larger than C1 OR mad total angular velocity is larger than C2
        OR std lateral acceleration is larger than C3
        RETURN accident
    ELSE
        RETURN cycling
    ENDIF
ENDFUNCTION
```

This threshold based classifier uses three features which characterize the IMU signals. When one of these features exceeds a certain threshold, the classifier will tag that 3 sec. sample as an accident. When they are all below their respective thresholds, the sample is classified as normal cycling.

The accident detection algorithm uses this classifier and works as follows:

```
SET T1,T2,T3,C1,C2,C3
INIT classifier
WHILE bike is turned on
    INPUT new features
    IF wheel speed is larger than T3 THEN
        % Cycling is detected
        INPUT update features
        CALL classifier RETURNING class
        IF class is accident THEN
            % Accident is detected
            SET loop is 1
            WHILE loop is less than 4 do
                INPUT update features
                COMPUTE absolute lean angle
                IF absolute lean is larger than T2 OR wheel speed is smaller than T3
                    OR GPS speed is smaller than T1 THEN
                        % Accident is confirmed
                        OUTPUT warning with longitude and latitude
                ENDIF
                ADD one to loop
            ENDWHILE
        ENDIF
    ENDIF
ENDWHILE
```

Layer 0 checks whether the bicycle is moving, this way an accident will for example not be detected when the bicycle fell over during standstill.

Layer 1 contains the classifier which returns either accident or cycling as a class.

Layer 2 will filter out any false positives that might occur during any harsh riding manoeuvres, which are directly followed by further cycling. This layer is looped three times. The bicycle must be laying on its side, or have a low velocity within 9 seconds for an accident to be confirmed. The GNSS/GPS velocity is also included since the wheel can still be turning after an accident when the bicycle is lying flat.

The performance of a prototype system with this classifier is validated on 11 simulated accident test runs and 11 hours of normal and harsh cycling, including edge cases which resulted in:

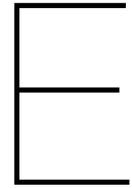
Confusion matrix Of the system		Sample from test runs		27% Precision
		Accident	Cycling	90.9% Sensitivity
Accident detected	Yes	TP = 10	FP = 27	99.85% Specificity
	No	FN = 1	TN = 18143	

Recommendations

The precision and sensitivity can be tuned with the thresholds. Increasing these will result in less false warnings, but more missed accidents.

Add more features to the classifier. A really useful pair is: maximum total angular velocity and the minimum angular velocity in longitudinal direction. However, you should not exclusively rely on these, since these values could be outliers. The use of angular velocity features makes the system independent of sensor placement.

Add a countdown and a button to the system. This countdown will start when an accident is confirmed by the algorithm. The emergency contacts will be informed when the user is not able to stop the countdown by pressing the button.



Thresholds

The threshold classifiers are developed using the method explained in section 2.5. This method compares the features of accident samples with features of cycling samples, which are all displayed in E.3. These figures show that 16 features have a clear distinction between cycling and accidents, these are listed in E.1. Then the effectiveness of these thresholds is determined using accident samples with a -2 seconds time shift. This means that the accident time series is cut in such a way that the sample which contains the accident only covers half of the accident. The effectiveness of the thresholds is summarized in E.2.

E.1. Useful thresholds

The list of useful thresholds for 3 second samples is:

1. Average acceleration in vertical direction.
2. Energy of acceleration in lateral direction.
3. Energy of angular velocity in longitudinal direction.
4. Interquartile range of acceleration in lateral direction.
5. Maximum acceleration in lateral direction.
6. Maximum total angular velocity.
7. Mean absolute deviation of acceleration in lateral direction.
8. Mean absolute deviation of total angular velocity.
9. Median acceleration in vertical direction.
10. Minimum angular velocity in longitudinal direction.
11. Total average power of acceleration in lateral direction.
12. Total average power of angular velocity in longitudinal direction.
13. Root mean square of acceleration in lateral direction.
14. Root mean square of angular velocity in longitudinal direction.
15. Standard deviation of acceleration in lateral direction.
16. Standard deviation of total angular velocity.

E.2. Threshold effectiveness

Table E.1 shows which threshold is passed for each accident. The samples of these accidents are subjected to a -2 second time shift. This is used to determine the effectiveness of a single threshold. A 1 means that this accident is detectable when using this threshold and 0 means that it is not. Table E.1 shows that collision C1 and skid T2 can not be detected with any thresholds for this time shift. The effectiveness of a single threshold can be determined by the amount of 1's in the corresponding column. For example: threshold 1 can only detect P1 and P3, where threshold 14 can detect most accidents. This table can be used to find combinations of thresholds that together can detect the most accidents, which resulted in classifier T1 and T2.

	Threshold	1	2	3	4	5	6	7	8	9	10	11	12	13	14
fall	F1	0	0	0	0	1	0	1	0	1	0	1	0	0	1
fall	F2	0	0	0	0	1	0	0	0	1	0	0	0	0	0
fall	F3	0	1	1	1	1	1	1	1	1	1	1	1	1	1
fall	F4	0	1	1	1	0	1	1	1	1	1	1	1	1	1
fall	F5	0	1	1	1	1	1	1	1	1	1	1	1	1	1
fall	P1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
fall	P2	0	1	1	1	1	0	1	0	1	1	1	1	1	1
fall	P3	1	1	1	1	1	1	1	1	1	1	1	1	1	1
fall	P4	0	0	0	0	1	1	0	0	1	0	0	0	0	0
fall	P5	0	1	1	1	1	1	1	0	0	1	1	1	1	1
collision	C1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
collision	C2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
collision	C3	0	1	1	1	1	1	1	1	0	1	1	1	1	1
collision	C4	0	0	0	1	1	1	1	1	0	0	0	0	1	1
collision	C5	0	1	1	1	1	1	1	1	1	1	1	1	1	1
collision	S1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
collision	S2	0	0	0	0	1	0	1	1	0	0	0	0	0	1
collision	S3	0	1	0	1	1	1	1	1	1	1	0	1	1	1
collision	S4	0	1	1	1	1	1	1	1	1	1	1	1	1	1
collision	G1	0	1	1	0	1	0	1	0	1	1	1	1	1	1
collision	G2	0	0	1	1	1	0	1	0	1	0	1	0	1	1
collision	G3	0	1	1	1	1	1	1	0	1	1	1	1	1	1
collision	G4	0	1	1	1	1	1	1	1	1	1	1	1	1	1
collision	W1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
collision	W2	0	0	0	0	0	0	0	0	1	0	0	0	0	0
collision	W3	0	1	1	1	1	1	1	0	1	1	1	1	1	1
collision	W4	0	1	1	1	1	1	1	1	1	1	1	1	1	1
collision	H1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
collision	H2	0	0	0	1	1	0	0	0	1	0	0	0	1	0
collision	H3	0	1	1	1	1	1	1	1	1	1	1	1	1	1
collision	H3	0	1	1	1	1	1	1	1	1	1	1	1	1	1
skid	T1	0	0	1	0	1	0	1	0	1	0	1	0	1	1
skid	T2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
skid	T3	0	0	1	0	1	1	1	0	0	0	1	0	0	1
skid	T4	0	1	1	1	1	1	1	1	0	1	1	1	1	1
skid	Q1	0	1	1	1	1	0	1	0	1	1	1	1	1	1
skid	Q2	0	1	1	1	1	1	1	0	1	1	1	1	1	1
skid	Q3	0	1	1	1	1	1	1	1	1	1	1	1	1	1
skid	Q4	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Table E.1: The thresholds that are passed by each accident. A passed threshold is indicated with a 1.

E.3. Sample distribution for each feature

The following figures show the distribution of samples for all features. The features with a clear distinction between accidents and normal cycling are used to find thresholds for the minimum viable product.

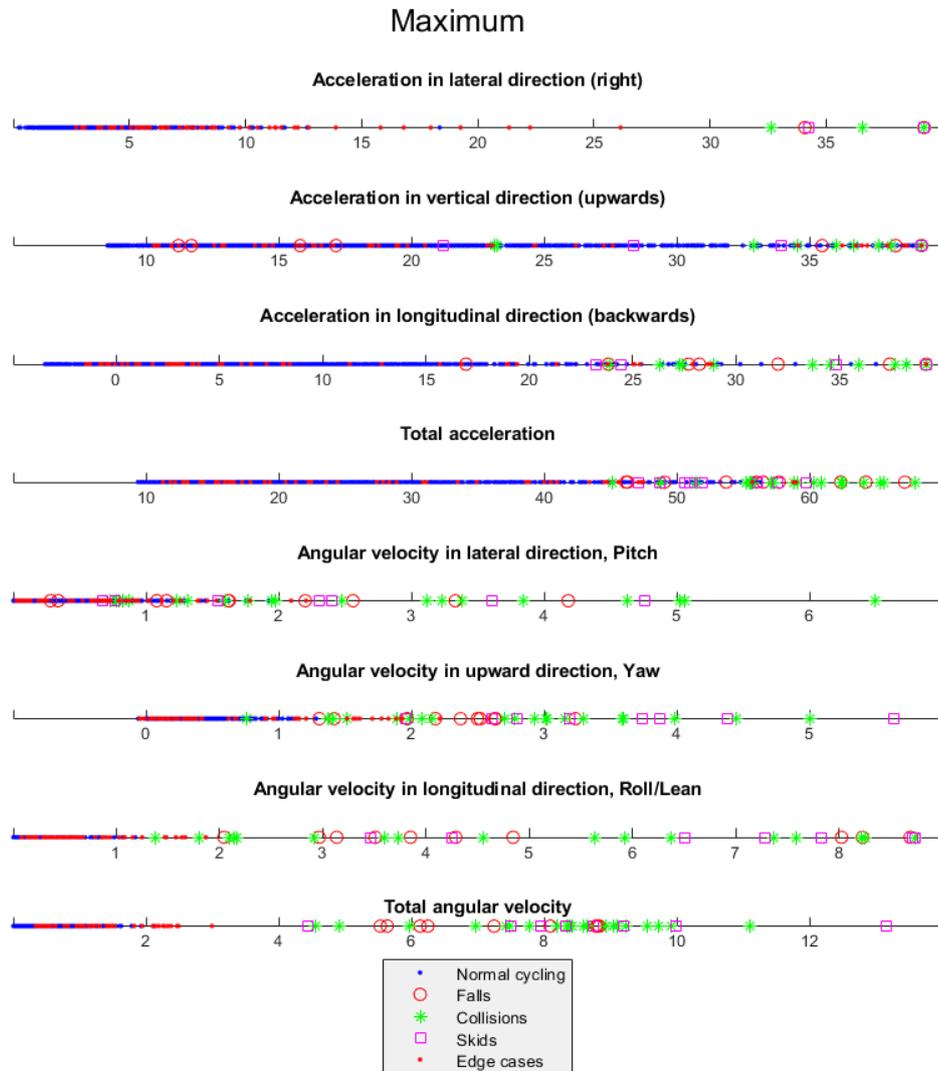


Figure E.1: Maximum of accident and cycling features for 8 signals.

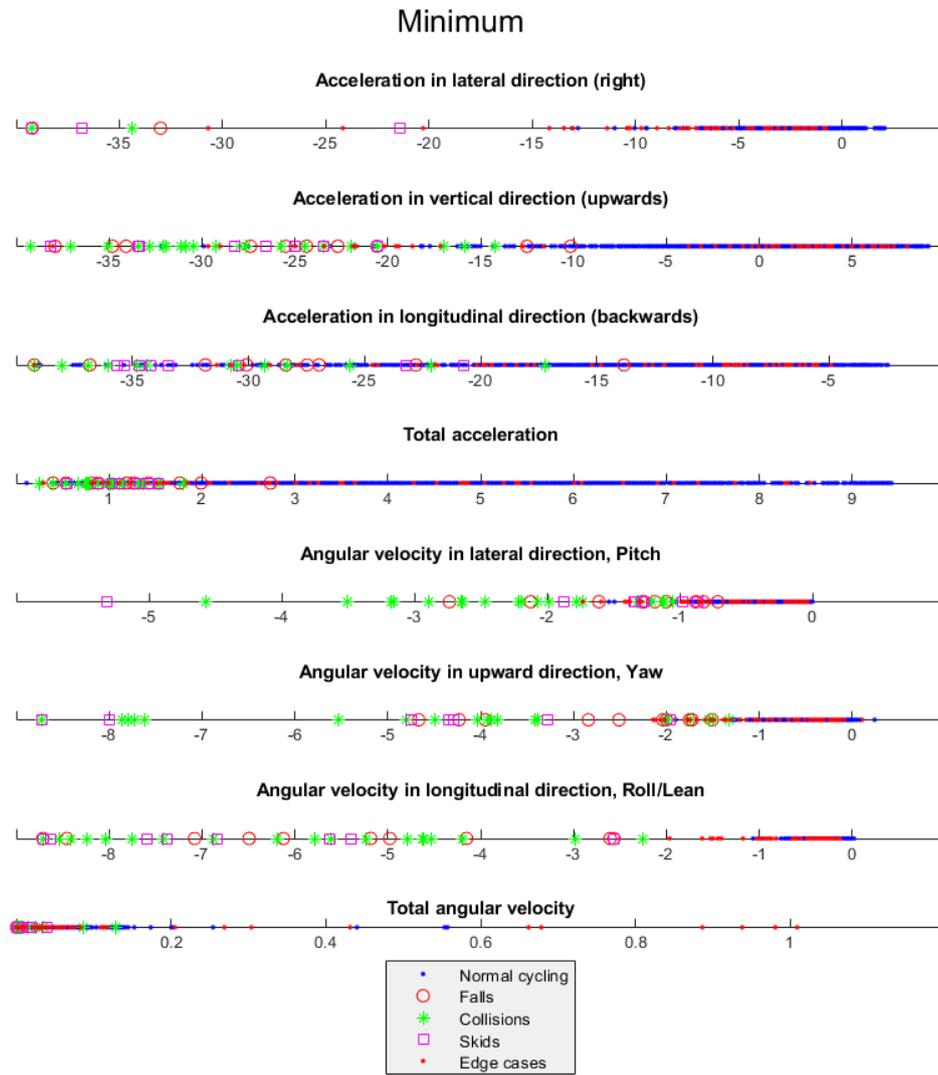


Figure E.2: Minimum of accident and cycling features for 8 signals.

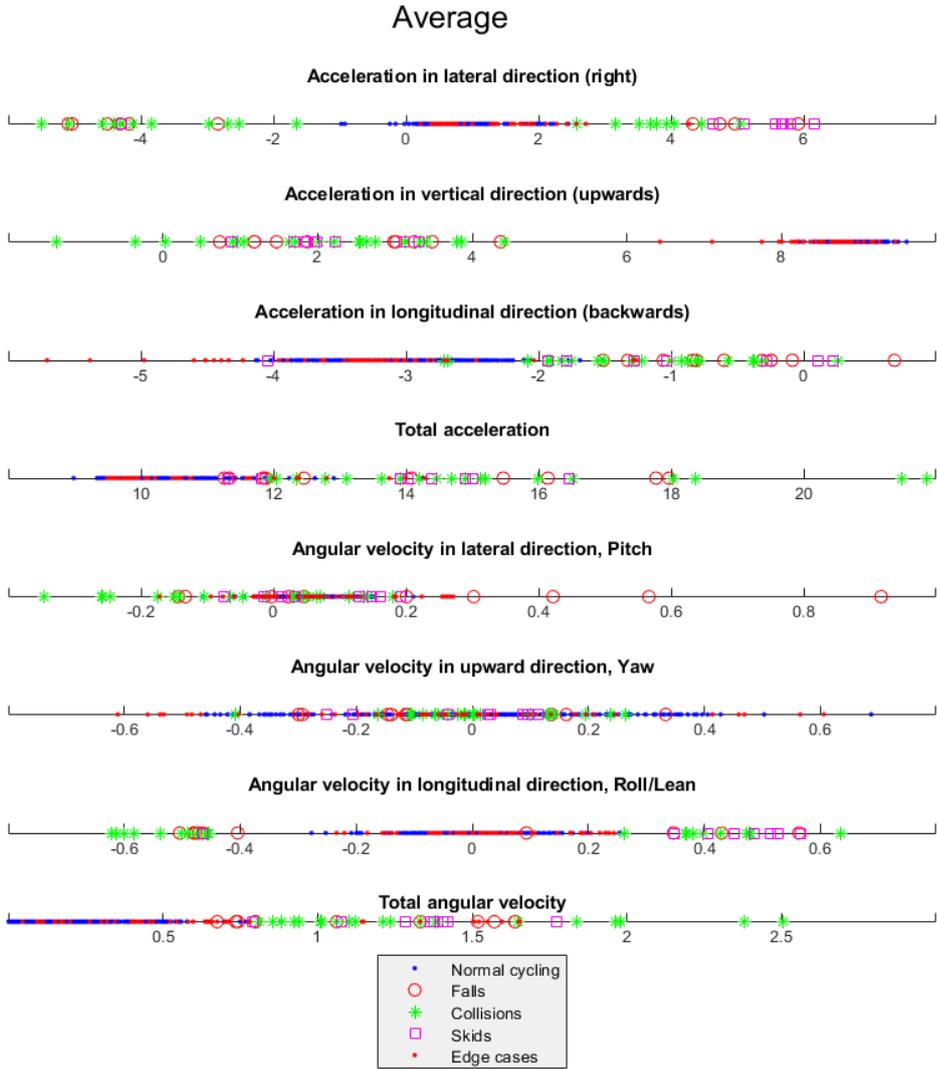


Figure E.3: Average of accident and cycling features for 8 signals.

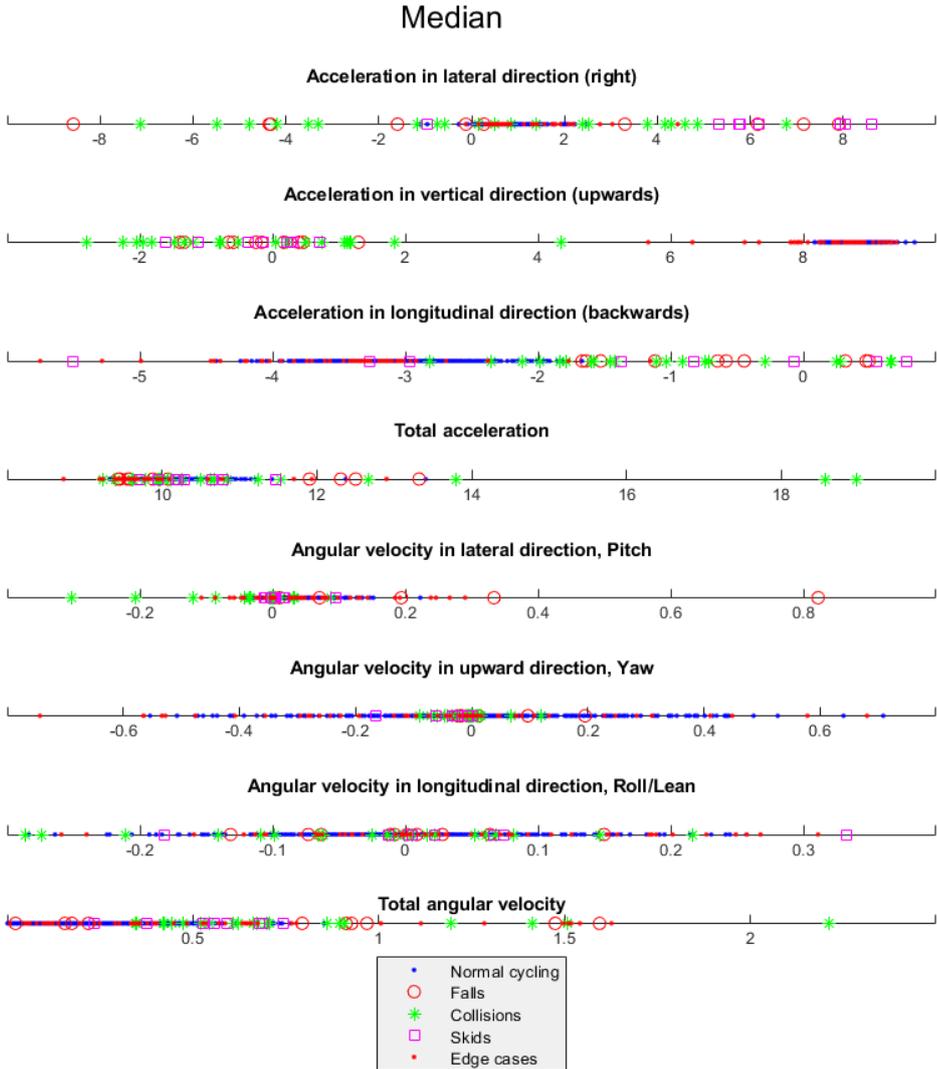


Figure E.4: Median of accident and cycling features for 8 signals.

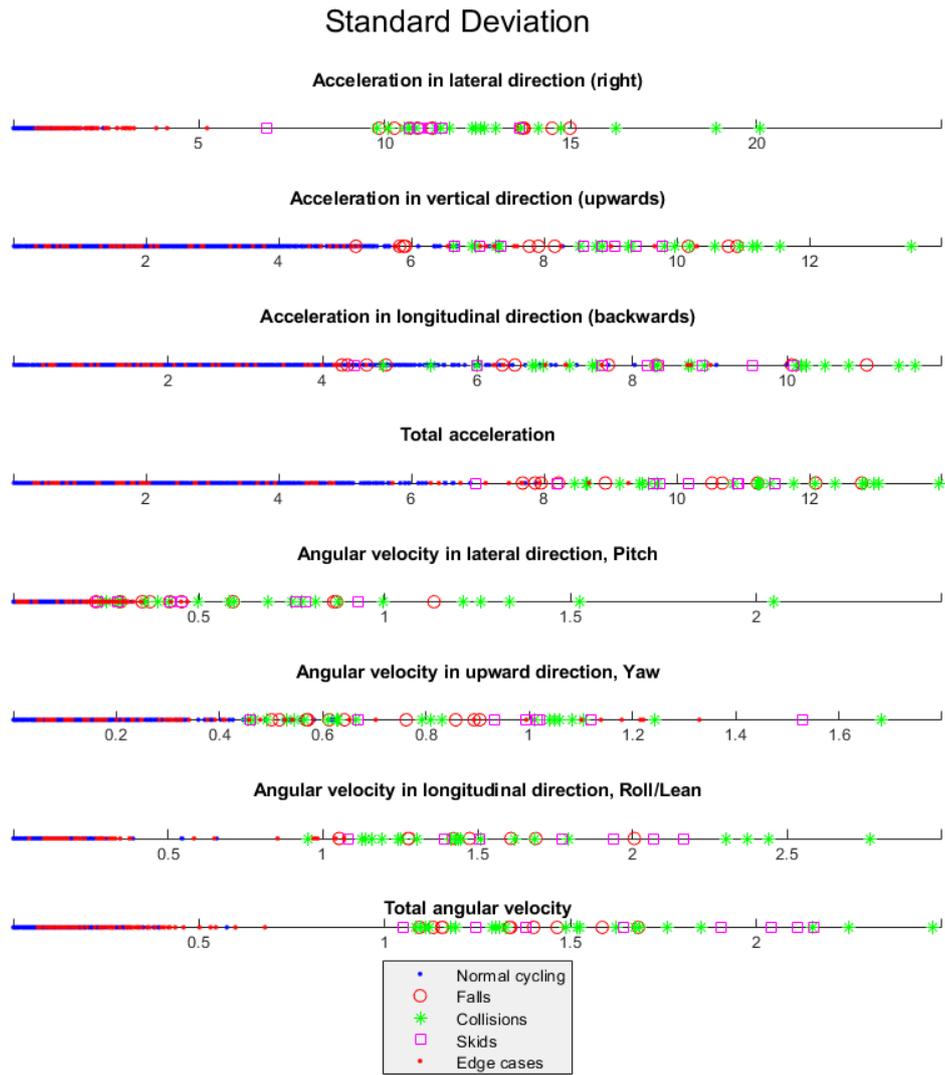


Figure E.5: Standard deviation of accident and cycling features for 8 signals.

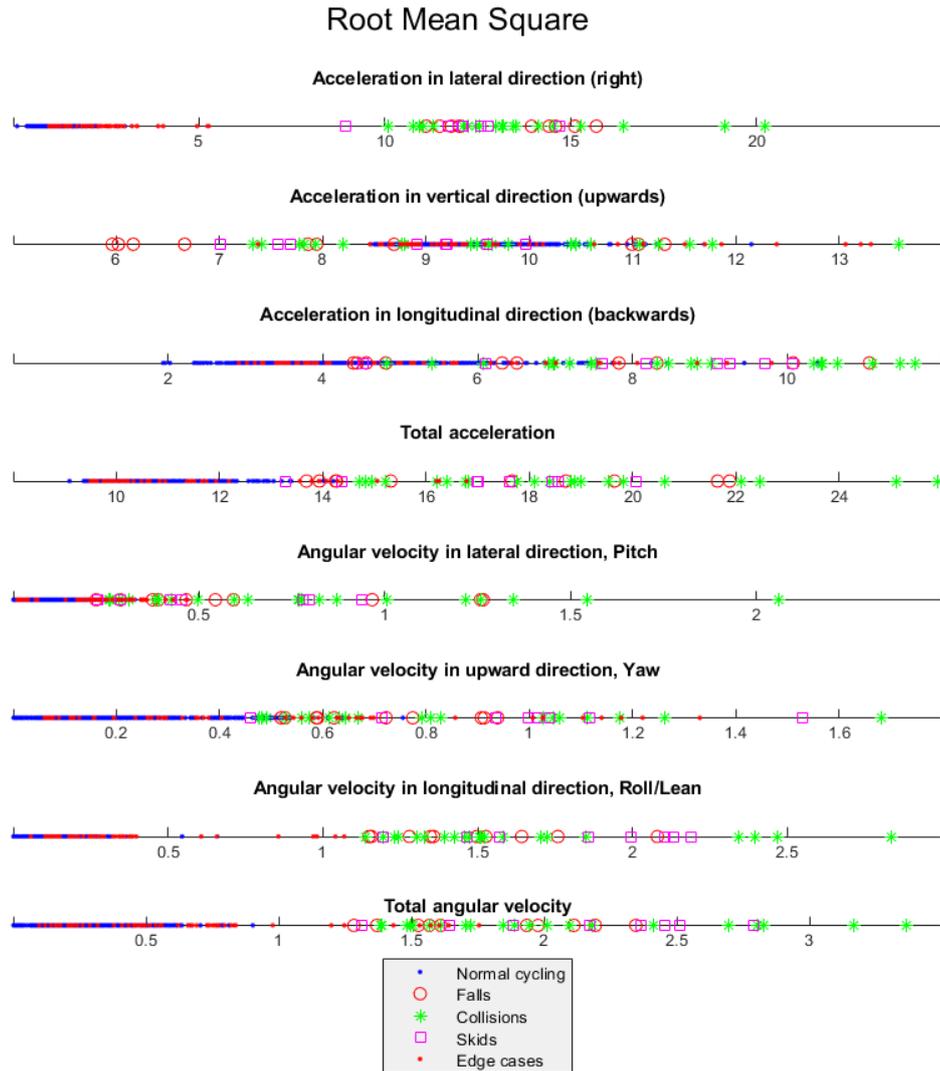


Figure E.6: Root mean square of accident and cycling features for 8 signals.

Mean Absolute Deviation

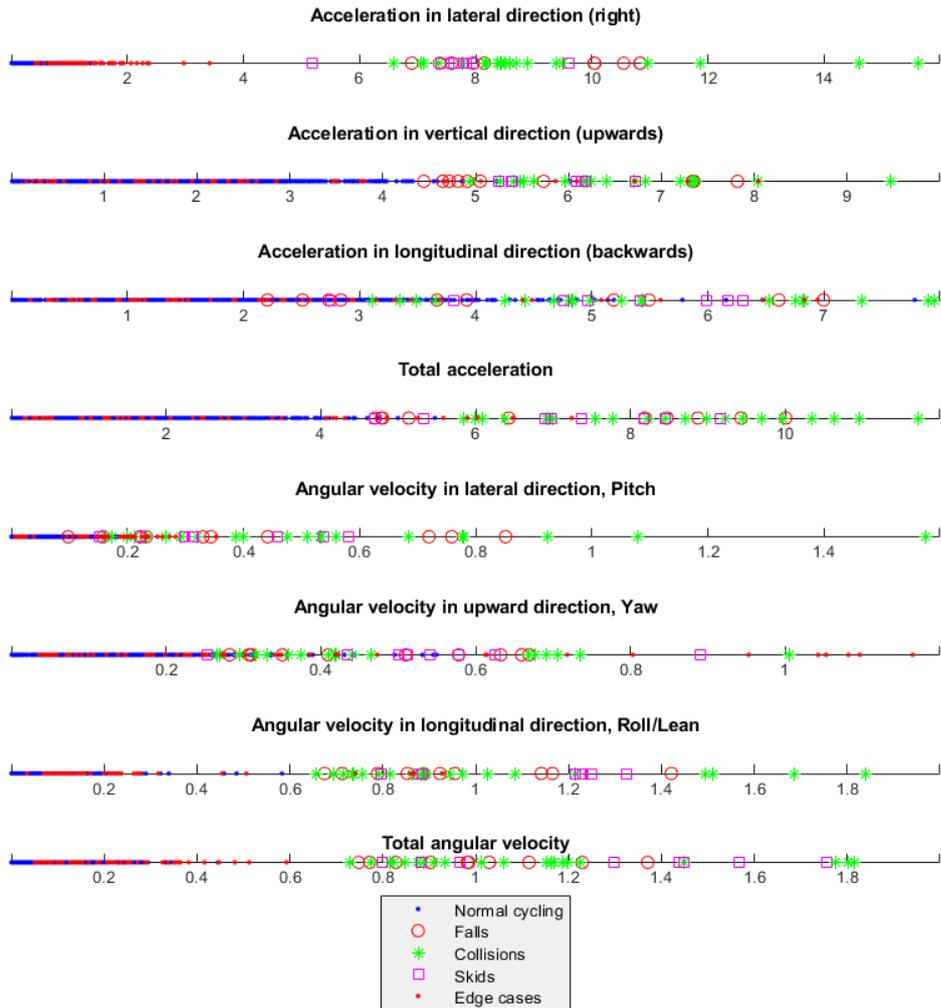


Figure E.7: Mean absolute deviation of accident and cycling features for 8 signals.

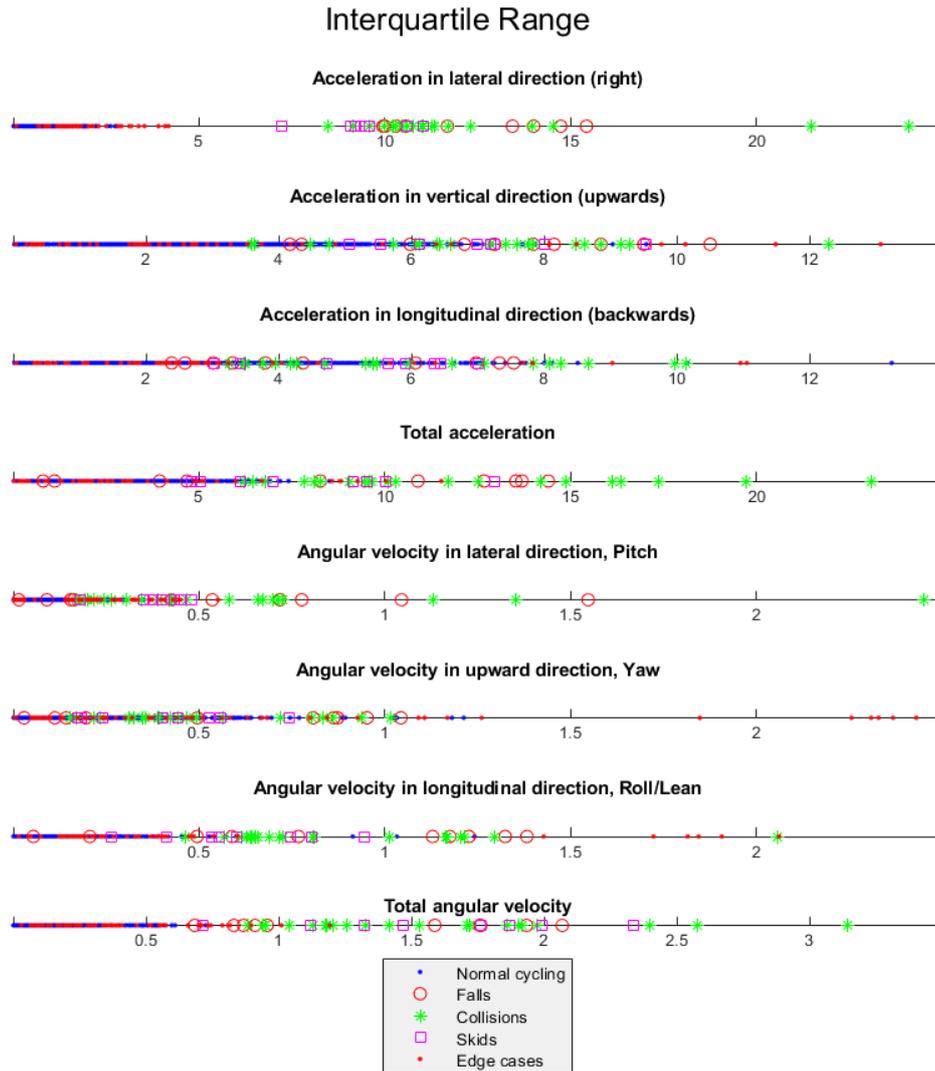


Figure E.8: Interquartile range of accident and cycling features for 8 signals.

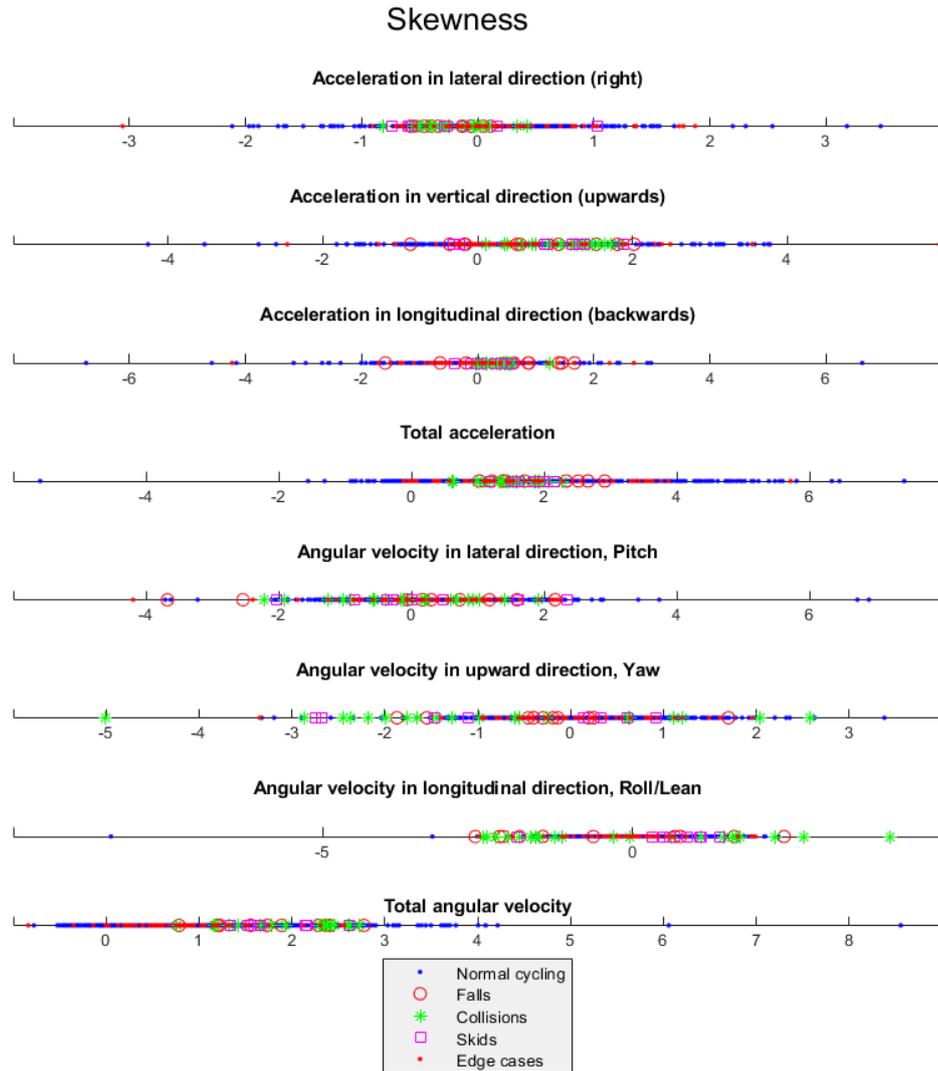


Figure E.9: Skewness of accident and cycling features for 8 signals.

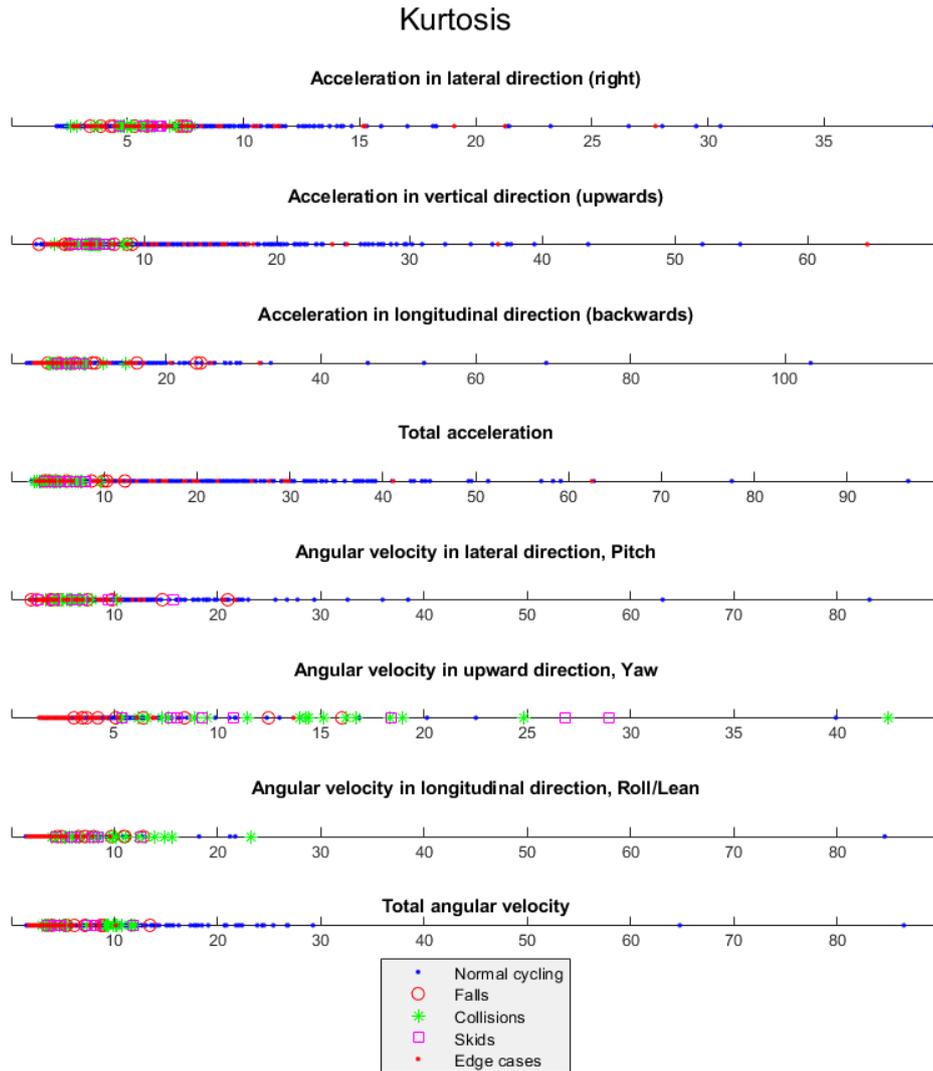


Figure E.10: Kurtosis of accident and cycling features for 8 signals.

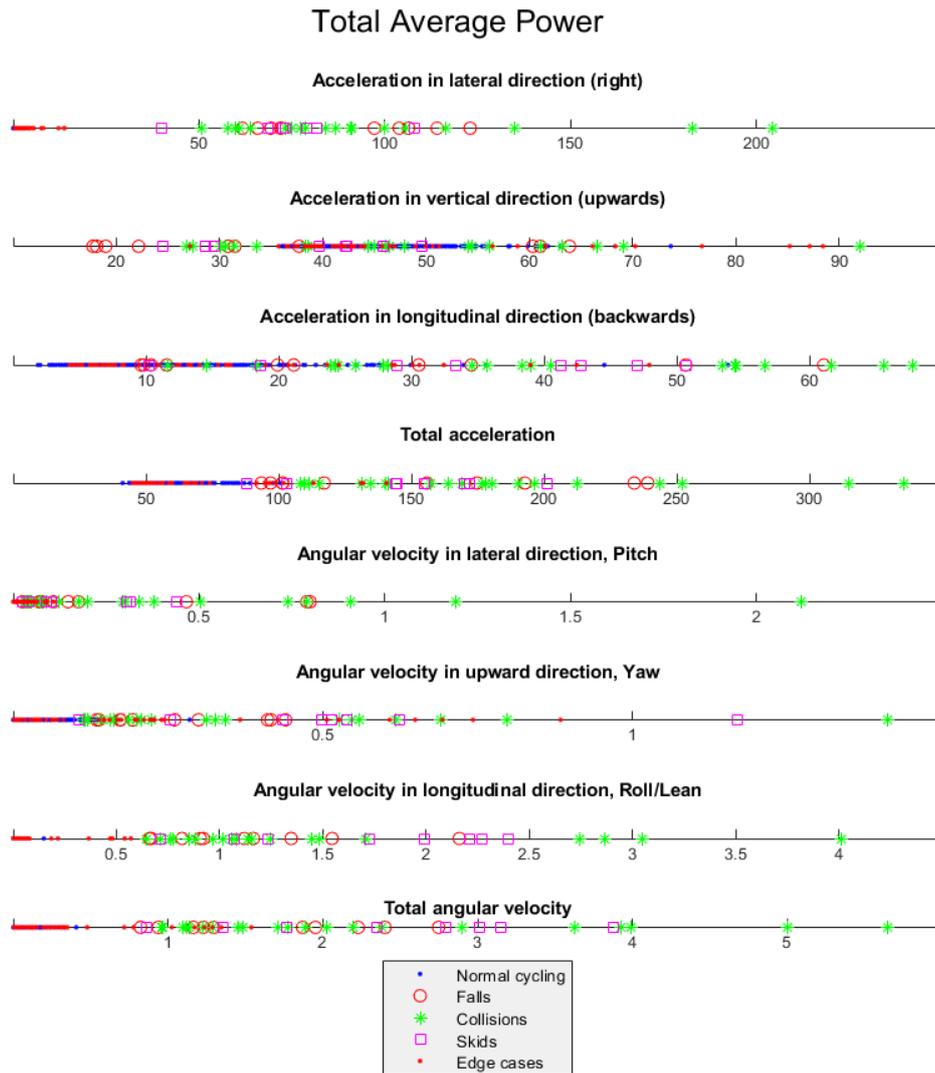


Figure E.11: Power of accident and cycling features for 8 signals.

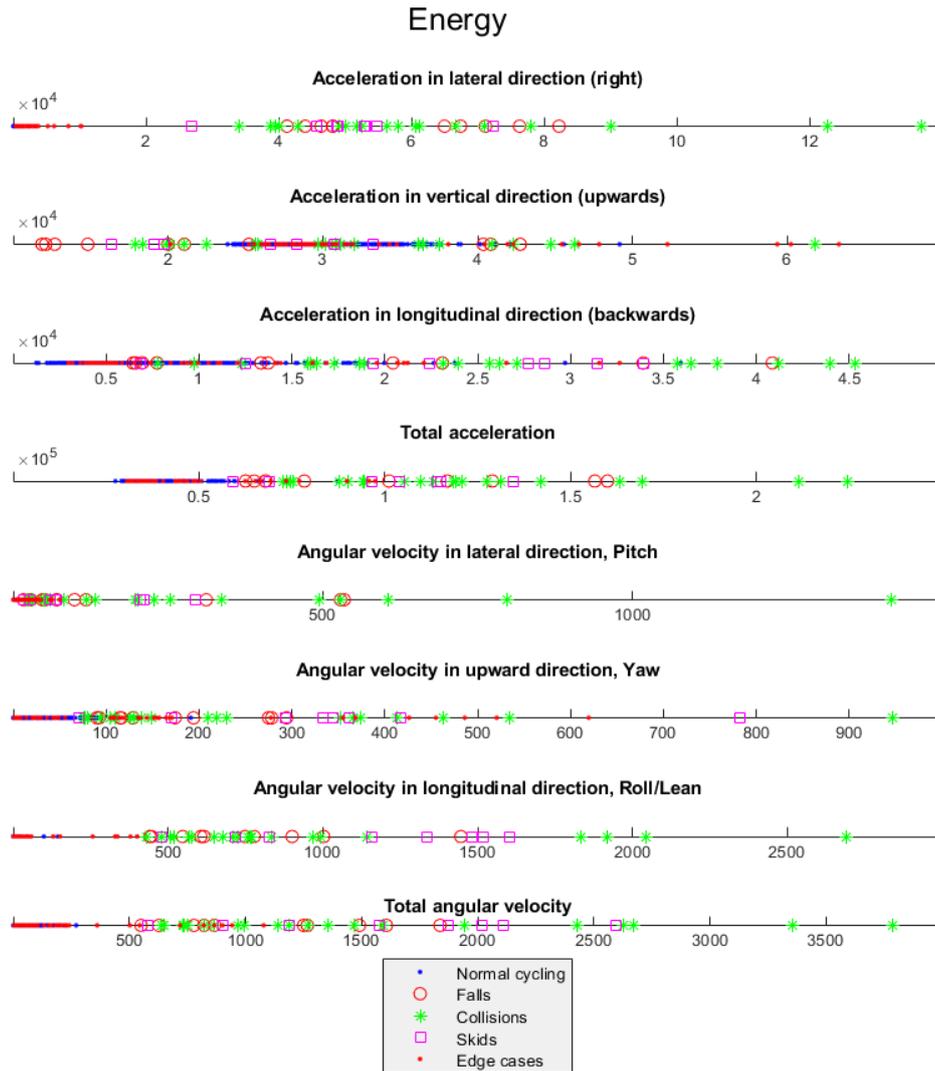
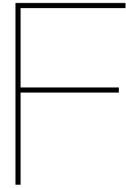


Figure E.12: Energy of accident and cycling features for 8 signals.



Performance of all validated algorithms

A total of 81 different machine learning classifiers are trained on 80% of the accident data and 25% of the cycling data. These are trained on 6, 12, and 2 dimensional training data that resulted from the PCA. Three amounts of cost parameters are tried: 20, 50, and 100. To find out which classifier performs the best, they are all (including T1 and T2) substituted in layer 1 of the algorithm. The performance of the individual algorithms is compared using 20% of accident data and 25% of normal cycling. The resulting performance metrics and values of the confusion matrices can be found in the following tables. The algorithm performs the best with a KNN classifier that works with 6 dimensional data and is trained using a cost parameter of 20.

PCA Components:	6											
Cost function:	[0,1;20,0]											
Classifier:	T1	T2	SVMG	SVML	SVMP	DT	DTPCA	KNN	KNNPCA	NB	NBPCA	
TP	10	10	11	11	11	10	10	11	10	9	11	
FP	6	27	6	12	15	21	27	9	3	30	15	
FN	1	1	0	0	0	1	1	0	1	2	0	
TN	18164	18143	18164	18158	18155	18149	18143	18161	18167	18140	18155	
Precision:	0,625	0,27	0,647	0,478	0,423	0,323	0,27	0,55	0,769	0,231	0,423	
Sensitivity:	0,909	0,909	1	1	1	0,909	0,909	1	0,909	0,818	1	
Specificity:	1	0,999	1	0,999	0,999	0,999	0,999	1	1	0,998	0,999	

PCA Components:	6										
Cost function:	[0,1;50,0]										
Classifier:	T1	T2	SVMG	SVML	SVMP	DT	DTPCA	KNN	KNNPCA	NB	NBPCA
TP	10	10	11	11	11	10	11	11	10	11	11
FP	6	27	21	30	27	36	66	37	12	141	23
FN	1	1	0	0	0	1	0	0	1	0	0
TN	18164	18143	18149	18140	18143	18134	18104	18133	18158	18029	18147
Precision:	0,625	0,27	0,344	0,268	0,289	0,217	0,143	0,229	0,455	0,072	0,324
Sensitivity:	0,909	0,909	1	1	1	0,909	1	1	0,909	1	1
Specificity:	1	0,999	0,999	0,998	0,999	0,998	0,996	0,998	0,999	0,992	0,999

PCA Components:	6										
Cost function:	[0,1;100,0]										
Classifier:	T1	T2	SVMG	SVML	SVMP	DT	DTPCA	KNN	KNNPCA	NB	NBPCA
TP	10	10	11	11	11	10	11	10	10	11	11
FP	6	27	42	30	38	12	32	15	6	141	129
FN	1	1	0	0	0	1	0	1	1	0	0
TN	18168	18147	18132	18144	18136	18162	18142	18159	18168	18033	18045
Precision:	0,625	0,27	0,208	0,268	0,224	0,455	0,256	0,4	0,625	0,072	0,079
Sensitivity:	0,909	0,909	1	1	1	0,909	1	0,909	0,909	1	1
Specificity:	1	0,999	0,998	0,998	0,998	0,999	0,998	0,999	1	0,992	0,993

PCA Components:	2										
Cost function:	[0,1;20,0]										
Classifier:	T1	T2	SVMG	SVML	SVMP	DT	DTPCA	KNN	KNNPCA	NB	NBPCA
TP	10	10	11	11	11	10	10	11	11	11	11
FP	6	27	15	15	9	21	9	6	12	141	43
FN	1	1	0	0	0	1	1	0	0	0	0
TN	18167	18146	18158	18158	18164	18152	18164	18167	18161	18032	18130
Precision:	0,625	0,27	0,423	0,423	0,55	0,323	0,526	0,647	0,478	0,072	0,204
Sensitivity:	0,909	0,909	1	1	1	0,909	0,909	1	1	1	1
Specificity:	1	0,999	0,999	0,999	1	0,999	1	1	0,999	0,992	0,998

PCA Components:	2										
Cost function:	[0,1;50,0]										
Classifier:	T1	T2	SVMG	SVML	SVMP	DT	DTPCA	KNN	KNNPCA	NB	NBPCA
TP	10	10	11	11	11	10	11	11	11	11	11
FP	6	27	6	24	21	36	35	22	12	141	46
FN	1	1	0	0	0	1	0	0	0	0	0
TN	18167	18146	18167	18149	18152	18137	18138	18151	18161	18032	18127
Precision:	0,625	0,27	0,647	0,314	0,344	0,217	0,239	0,333	0,478	0,072	0,193
Sensitivity:	0,909	0,909	1	1	1	0,909	1	1	1	1	1
Specificity:	1	0,999	1	0,999	0,999	0,998	0,998	0,999	0,999	0,992	0,997

PCA Components:	2										
Cost function:	[0,1;100,0]										
Classifier:	T1	T2	SVMG	SVML	SVMP	DT	DTPCA	KNN	KNNPCA	NB	NBPCA
TP	10	10	11	11	11	10	11	11	11	11	11
FP	6	27	25	27	15	12	32	46	15	141	55
FN	1	1	0	0	0	1	0	0	0	0	0
TN	18167	18146	18148	18146	18158	18161	18141	18127	18158	18032	18118
Precision:	0,625	0,27	0,306	0,289	0,423	0,455	0,256	0,193	0,423	0,072	0,167
Sensitivity:	0,909	0,909	1	1	1	0,909	1	1	1	1	1
Specificity:	1	0,999	0,999	0,999	0,999	0,999	0,998	0,997	0,999	0,992	0,997

PCA Components:	12										
Cost function:	[0,1;20,0]										
Classifier:	T1	T2	SVMG	SVML	SVMP	DT	DTPCA	KNN	KNNPCA	NB	NBPCA
TP	10	10	10	10	10	10	11	11	11	9	10
FP	6	27	41	52	44	21	85	22	57	27	52
FN	1	1	1	1	1	1	0	0	0	2	1
TN	18166	18145	18131	18120	18128	18151	18087	18150	18115	18145	18120
Precision:	0,625	0,27	0,196	0,161	0,185	0,323	0,115	0,333	0,162	0,25	0,161
Sensitivity:	0,909	0,909	0,909	0,909	0,909	0,909	1	1	1	0,818	0,909
Specificity:	1	0,999	0,998	0,997	0,998	0,999	0,995	0,999	0,997	0,999	0,997

PCA Components:	12										
Cost function:	[0,1;50,0]										
Classifier:	T1	T2	SVMG	SVML	SVMP	DT	DTPCA	KNN	KNNPCA	NB	NBPCA
TP	10	10	11	10	10	10	11	11	11	9	10
FP	6	27	111	83	83	36	85	16	167	31	109
FN	1	1	0	1	1	1	0	0	0	2	1
TN	18171	18150	18066	18094	18094	18141	18092	18161	18010	18146	18068
Precision:	0,625	0,27	0,09	0,108	0,108	0,217	0,115	0,407	0,062	0,225	0,084
Sensitivity:	0,909	0,909	1	0,909	0,909	0,909	1	1	1	0,818	0,909
Specificity:	1	0,999	0,994	0,995	0,995	0,998	0,995	0,999	0,991	0,998	0,994

PCA Components:	12										
Cost function:	[0,1;100,0]										
Classifier:	T1	T2	SVMG	SVML	SVMP	DT	DTPCA	KNN	KNNPCA	NB	NBPCA
TP	10	10	11	11	11	10	11	10	11	11	10
FP	6	27	153	143	156	12	196	12	189	141	129
FN	1	1	0	0	0	1	0	1	0	0	1
TN	18172	18151	18025	18035	18022	18166	17982	18166	17989	18037	18049
Precision:	0,625	0,27	0,067	0,071	0,066	0,455	0,053	0,455	0,055	0,072	0,072
Sensitivity:	0,909	0,909	1	1	1	0,909	1	0,909	1	1	0,909
Specificity:	1	0,999	0,992	0,992	0,991	0,999	0,989	0,999	0,99	0,992	0,993