Employing Sensor and Service Fusion to Assess Driving Performance

by

Seyed Vahid Hosseinioun

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The remarkable increase in the use of sensors in our daily lives has provided an increasing number of opportunities for decision-making and automation of events. Opportunities for decision-making have further risen with the advent of smart technology and the omnipresence of sensors. Various methods have been devised to detect different events in a driving environment using smart-phones as they provide two main advantages: they remove the need to have dedicated hardware in vehicles and they are widely accessible.

Rewarding safe driving has always been an important issue for insurance companies. With this intention, they are now moving toward implementing plans that consider current driving usage (Usage-based-drive plans) in contrast with traditional history-based-plans. The detection of driving events is important in insurance telematics for this purpose. Events such as acceleration and turning are good examples of important information. The sensors are capable of detecting whether a car is accelerating or braking, while through fusing services we can detect other events like speeding or the occurrence of a severe weather phenomenon that can affect driving.

This thesis aims to look at the telematics from a new angle that employs smart-phones as the sensing platform. We proposed a new hybrid classification algorithm that detects acceleration-based events with an F_1-score of 0.9304 and turn events with an F_1-score of 0.9038. We further performed a case study on measuring the performance of driving utilizing various measures. This index can be used by a wide range of benefactors such as the insurance and transportation industries.

keywords: auto-insurance telematics, mobile media, driving assessment, intelligent driving, sensory event detection.

Acknowledgements

I would like to express my sincere gratitude to Dr. Abdulmotaleb El-Saddik for his exemplary supervision of this research work and for his trust, continuous support, and guidance.

Likewise, special and sincere thanks goes to Dr. Hussein Al Osman, my co-supervisor, for the invaluable assistance, friendship and continuous supply of feedback and discussions he provided throughout my research in my Masters studies.

I would also like to greatly thank my parents, Mojtaba Hosseinioun and Mansoureh Meghdadian, for their inspiration, understanding, patience and endless support. You made this possible.

Finally, I would like to convey my appreciation to Mr. Kazi Masudul Alam, Mr. Oscar Mou, Mr. Juan Arteaga and to my other colleagues at the MCRLab for their positive and valuable input, contributions, and ideas throughout this work.

Dedication

I dedicate this to work to my parents whom I love very much. Without them and their non-stop support, I would have not been able to make this work possible.

I also dedicate this work to my greater family-friends who were there with me when I needed them the most. I also dedicate this work to the people of Iran who helped me and supported my efforts and education to be able to make it here.

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

API	Application Program Interface
CSV	Comma Separated Values
DPI	Driving Performance Index
FFT	Fast Fourier Transform
FoR	Frame of Reference
GPS	Global Positioning System
JSON	JavaScript Object Notation
MEMS	Micro Electro-Mechanical Systems
OBD	On-Board Diagnostics
OS	Operating System
PAYD	Pay as You Drive
PMM	Performance Measure Monitoring
SDK	Software Development Kit
SVM	Support Vector Machine
UBI	Usage Based Insurance

Chapter 1

Introduction

This work will discuss the role in which sensors and services are capable in monitoring vehicles and the process of driving. In this introduction, first, the problem this research will seek to solve is identified; then, the motivation behind solving this problem is discussed, and finally the importance of the proposed driving assessment mechanism is described..

1.1 Problem Statement

Vehicle insurance is required by law in almost every corner of the world. Rewarding safe driving has always been an important aspect for insurance companies, and they have had many different ways to approach that. These days, the auto-insurance industry is coming to the realization that the current methods used to calculate premiums do not consider equity and can be further improved [29]. The reason is that the these methods are preoccupied with the history of the driver and do not consider the quality of driving at the present time [14]. As a result, alternative insurance plans are gaining ground, namely pay-as-you-drive (PAYD)—more generally known as usage-based insurance (UBI) plans [14].

In auto-insurance, for plans that use PAYD[14] and UBI[38], the premium of a user is calculated on the basis of the distance the insured vehicle travels routinely and data collected to assess how well said user is driving. The data collected include the times of day users are on the road, the way users accelerate and brake, the vehicle type, and measures of location. Therefore, this enables the insurance providers to recalculate the user's premiums and benefits regularly in terms of how they drive.

This method differs drastically from traditional insurance practices, where companies award deductions to drivers demonstrating certain behaviour;–no-claim bonuses, for example, for drivers who have not made a claim on their insurance for a given length of time.

Within the defined scopes, the problem of how driving events can be sensed, measured, and used to give a metric is what this thesis aims to resolve.

The new generation of smart-phones, with their powerful processing and sensing capabilities, are increasingly regarded as an alternative to hand-held computers [15]. For the

reasons mentioned above, when it comes to sensing, monitoring, and understanding the process of driving , smart-phones can be an important source for gathering data. Smart-phones provide several advantages for in-vehicle sensing, as follows:

1. They remove the need to have dedicated sensors. This will help in reducing costs associated with installing separate black-boxes for in-vehicle sensing scenarios.
2. They are already widely available and are being used ubiquitously. There is no need or concern to create a new infrastructure that is approved by the public.
3. They have a pre-existing architecture. There are no concerns related to creating a new sensing and communication architecture.

1.2 Motivation

Based on the reasons mentioned above, smart-phones, can be the solution to the problem of in-vehicle driving assessment. By using the large set of sensors in modern smart-phones and fusing them with other services (e.g. web-services), we would provide an elaborate answer to solve this problem. By employing mobile-sensing for this purpose, we can help both the insurance companies and their clients.

Monitoring the task of driving, namely by detecting events related to the quality of driving, is of utmost importance. This analysis is the decisive factor to first, determine the premium of the user, and second, the level of trust and risk that is associated with a particular user of the system.

Furthermore, the need for constant monitoring of drivers is not just restricted to the insurance industry; road transportation industry can also benefit substantially. The reason being that transportation companies are interested in this task as they can gain crucial information about their employees from monitoring and assessing their driving behaviours.

These event-related data chunks can be of great value to many other parties, such as urban planners, by providing them with valuable information on patterns of driving and the potential risks that are involved with specific roads and different vehicle types. By the same token, network transportation companies such as Uber, Lyft, and Summon are potentially interested in ensuring their drivers are driving responsibly to the satisfaction of their customers.

At the same time, drivers would benefit from having more information on road conditions, traffic, and surrounding vehicles. It has been demonstrated in Wouters (2000) that providing driving feedback and assessment records to drivers has reduced the rate of accidents around 20% on average. [49]

According to the Victoria Transport Policy Institute in British Columbia, widespread adoption of telematics has other potential benefits [8]:

1. Reductions in traffic, accidents, and cost of roads and parking.

2. By exploring alternative approaches to driving such as biking, walking, or using public transportation, healthier lifestyles are promoted. Furthermore, people will be driving less, resulting in the reduction of pollution and energy use.

3. Opportunities for urban planners to explore other land-use objectives. These could be increasing or decreasing the number of lanes, expanding the road in a busy area, or even using roads not travelled as often for other purposes such as parking places or recreational uses.

To solve this problem, we propose a system to resolve the problems associated with driving assessment as described above by employing a fusion of services and sensors from smart-phones. This process would lead to an assessment of driving quality that can be used for a wide range of benefactors, mainly the insurance sector, the transportation industry, and drivers.

1.3 Research Methodology

In this work, we adopted the design research methodology. Design is defined as a process that aims to identify a specific need and develop a product to solve that need [13]. This process applies wisdom from engineering and science, and if not able to find the required knowledge, it makes necessary assumptions to minimize the risks and pursue the process [13].

Design research's goal is to gain an understanding of the process of design with its complexity and to furthermore increase the development and validation of knowledge, methods, and tools to improve the current situation in design [13]. The design research methodology is a set of supporting methods to be employed as a framework for doing design research.

In this type of methodology, the objective of the research is to solve problems and contribute by creating a design process that will help in using and implementing your work in related industries. In other words, this research is performed for the sake of designing a practical solution to common problems in a wide variety of industries related to traffic. Namely, in this work, insurance and transportation are the target industries that would hugely benefit from this research.

1.4 Contributions

As the section above demonstrates, the contributions made in this thesis are the following:

- Establishing a new classification algorithm that detects in-vehicle acceleration-related events;–namely decelerations, accelerations, and turns with improved accuracy in comparison to previous works.
- Employing data from detected events and service data in a progressive case study to measure driving performance.

The evaluations made in the present research demonstrate that the results have a higher precision in comparison to similar works. This work compiled a system that records and provides feedback about these detected events as a service.

To carefully assess the quality of driving, we present, as an example in a case study, a driving performance index calculated using the measured driving events. This index measurement is based on that which the current literature and reports from the insurance industry and other relevant parties concerned with traffic identify as important measures for driving assessment.

Chapter 2

Background and Related Works

In this chapter, detailed background information on the necessary components of our proposed framework will be given. Since the framework employs sensors, services, machine learning, and the recalibration of values, the background information will be introduced in the first section.

In the second section of this chapter, there will be a review of some of the current literature on using sensors, mainly from smart-phones, to detect driving events. Further along in this chapter, an analysis of these works is given to assess the driving habits of their users.

The related works section will survey the relevant papers in the field. We will classify these works based on whether they focus on detecting events using sensors only or by additionally relying on available services; furthermore, we will consider whether the objective of these papers was driving assessment or traffic anomaly detection.

2.1 Background

For the purpose of having a common terminology, many of the symbols used in this section are described in Table 2.1.

2.1.1 Sensors

Before detecting any sort of event, having an understanding of the method of capturing data and processing such information is required. Since the implementation of this framework was effected in Android, the discussion is based on sensing with Android phones. This knowledge of sensors is important when selecting the corresponding sensors for input data.

In the next section, all the important sensors of the phone that were employed in the detection of events are introduced. The sensor and sensing concepts described here should be similar in other mobile device architectures such as iPhones or BlackBerry with minor differences..

Symbol	Unit	Description
a	m/s^2	Acceleration
al	m/s^2	Linear acceleration
g	m/s^2	Gravity = 9.81
v_t	m/s	Current speed (Velocity) of the vehicle in time t
v_l	m/s	Speed limit
α_S	degree °	Azimuth of the orientation sensor
β	degree °	Roll
γ	degree °	Pitch
ϕ	degree °	Geographical latitude
λ	degree °	Geographical longitude
A	m	Geographical altitude
α_L	degree °	Bearing (Azimuth) of the location services
Δt	seconds	Overall time period of event

Table 2.1: Sensing and Services Symbol and Descriptions Table

1) Measuring Motion with Sensors

Accelerometer sensor: This sensor measures the acceleration that is applied to the device, including the force of gravity [5]. Acceleration in smart-phones is represented for any sensor event by a three-dimensional vector based on three device coordinates (see Fig. 2.1 for symbols and descriptions).

1. a_x is acceleration value plus g_x on the x-axis
2. a_y is acceleration value plus g_y on the y-axis
3. a_z is acceleration value plus g_z on the z-axis

For example, when the device lies on a flat surface such as a table, the acceleration value of the z-axis Acc_z is $\sim +9.81$. This figure corresponds to the acceleration of the device ($0\ m/s^2$) minus the force of gravity ($-9.81 m/s^2$) [5].

Linear Acceleration sensor: This is in fact a synthetic sensor [33]. Synthetic sensors like the linear acceleration sensor and the gravity sensor, are Software-based sensors that derive their data from one or more of the hardware-based sensors with some modifications [7].

The linear acceleration values correspond to a three-dimensional vector, showing forces measured by the accelerometer that are not caused by gravity along each device axis [5]. They are the result of the forces shown in Eq. (2.1) acting on the phone:

$$\vec{al} = \vec{a} - \vec{g}\ [5] \tag{2.1}$$

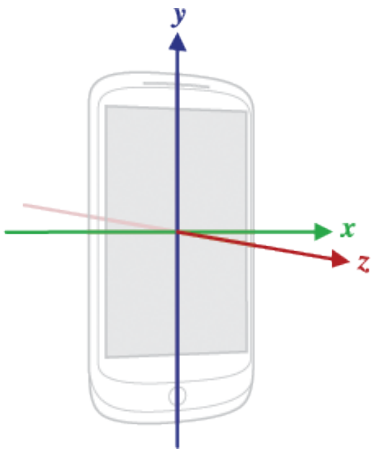

Figure 2.1: Device Coordinate system used by the Android Sensor Framework[7]

Gravity sensor: This is also a synthetic sensor [33]. The gravity values correspond to a three-dimensional vector indicating the direction and magnitude of gravity. It shows forces that are caused only by gravity. Units are in m/s^2. The coordinate system is the same as that used by the acceleration sensor [7].

Gyroscope sensor: The gyroscope is a hardware-based sensor [33]. The values of this sensor correspond to the rate of rotation around the device's local three axes with the same co-ordinate as that outlined in Fig. 2.2. Values are in $radian/s$ units [5]. Usually these values are integrated over time to calculate rotation over a certain time stamp.

Given the evidence above, the best sensor to use for obtaining the acceleration measures of a car is Linear Acceleration, because data based on gravity are not of concern in this research [5]. Moreover, no filtering would be required to remove the effects of gravity on the original accelerometer signal values when processing the data.

2) Measuring Position and Orientation

Another important quality to measure is the orientation of the phone in regard to the earth's fixed orientation. The Android APIs provide a set of libraries that can be used to measure this quality.

Orientation Sensor: This is a synthetic sensor [33], which provides the phone orientation according to Euler angles with minor modifications (see Fig. 2.2 on page 8). Leonhard Euler introduced the three Euler angles to describe the orientation of a rigid body in a standard way. All three values are angles in degrees [5].

- **Azimuth** or $\boldsymbol{\alpha}$ is the angle between the magnetic north direction and the y-axis, around the z-axis (0 to 359). See Table 2.2 for the relation between the azimuth angle and geographical direction.

Azimuth angle value (From North)	**Direction**
0	North
45	North-East
90	East
135	South-East
180	South
225	South-West
270	West
315	North-West

Table 2.2: Table of True North azimuth values

- **Pitch** or β is the rotation around x-axis (-180 to 180), with positive values when the z-axis moves toward the y-axis.
- **Roll** or γ is the rotation around the y-axis (-90 to 90), increasing as the device moves clockwise. For historical reasons, the roll angle in the Android device is positive in the clockwise direction, while mathematically speaking, it should have been positive in the counter-clockwise direction.

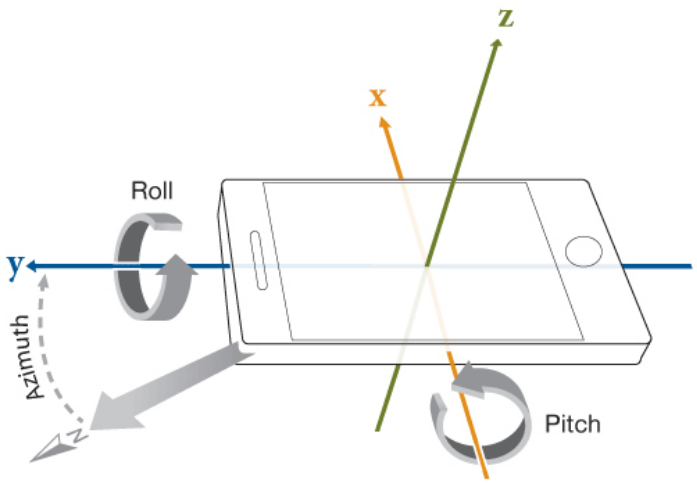

Figure 2.2: How phone orientation works in Android [1]

3) Direct Sensing from the vehicle

Almost all modern cars are equipped with reporting and self-diagnosis features. The On-Board Diagnostic (OBD) computer program for cars is used to provide a detailed report when any detected issues necessitating diagnosis arise. Fortunately, this technology can also be used for sensing capabilities, as this research has explored.

All vehicles manufactured in the United States since 1996 and in Europe since 2001 have implemented this standard. OBDs, through this port, give the car owner, repair technicians, and repair facilities the ability to diagnose potential issues with the vehicle.

Figure 2.3: A car's female OBD-II connector

Figure 2.4: A Bluetooth ELM327 OBD-II Scanner with male connector

The OBD-II provides the following data: real-time speed v_t, revolutions per minute or rpm, and other diagnostic indicators. This information is directly measured on the vehicle and therefore considered the golden standard in vehicular velocity sensing.

An interface is usually used for connectivity, with the most common being the ELM327, a programmed micro-controller that interfaces the car's on-board computer and network to a USB or Bluetooth interface. Bluetooth communication is ideal for this type of research and thus was used for the case scenario.

2.1.2 Services

This subsection outlines the background information on the services used throughout this work. Services provisioning models, such as Internet services, make resources and computing power available to the their users on demand [35]. For the purpose of driving assessment, three of these services were employed: location, roads, and weather providers. Android location services, OpenWeatherMap[1], and Here Maps[2] are what we used for the implementation of our framework. We shall briefly introduce them in the rest of this section.

1) Location Services

Android location services are a set of functionalities that gives the developer the capability to deal with a set of real-time data from the driving session. According to Android APIs documentation, these services give the developers the ability to do the following [3]:

- Query for a list of location providers in order to retrieve the last known user location.
- Register or unregister for the reception of updates on the user's current location along with other related information from location providers of the system in a periodic or non-periodic way.
- Register or unregister for the occurrence of a specific task should the device arrive within a given proximity.

Location updates data are of special interest. These data sets include basic geographical location information of longitude, latitude, and altitude. They also contain motion-related information such as speed and movement bearing. The last two will only be applicable in scenarios where the phone is being carried and moved around.

- **Geographic Coordinate Data:** The data, composed of longitude $\boldsymbol{\lambda_t}$ and latitude $\boldsymbol{\phi_t}$ at a given time, show geographical location.
- $\boldsymbol{A_t}$ or **Current Altitude:** Provides current device altitude in metres above the WGS 84 reference ellipsoid.
- $\boldsymbol{v_t}$: Shows instantaneous velocity with good accuracy. The values are in m/s.
- $\boldsymbol{\alpha_t}$ or **Current Bearing:** "Bearing" as used in navigation, usually refers to the direction of motion. Bearing is usually measured in degrees [11]. In Android, this value gives the horizontal direction of travel as an angle and is guaranteed to be in the range of (0.0, 360.0] degrees [4]. Technically, the bearing provided here has the same definition as that of the azimuth with one very important difference: unlike orientation sensor, the azimuth value introduced earlier is free of device-orientation and is dependant on the vehicle, and thus device, movement.

[1] Follow this link for more information: http://openweathermap.org/api
[2] Here mapping and location services, retrieved from https://developer.here.com/

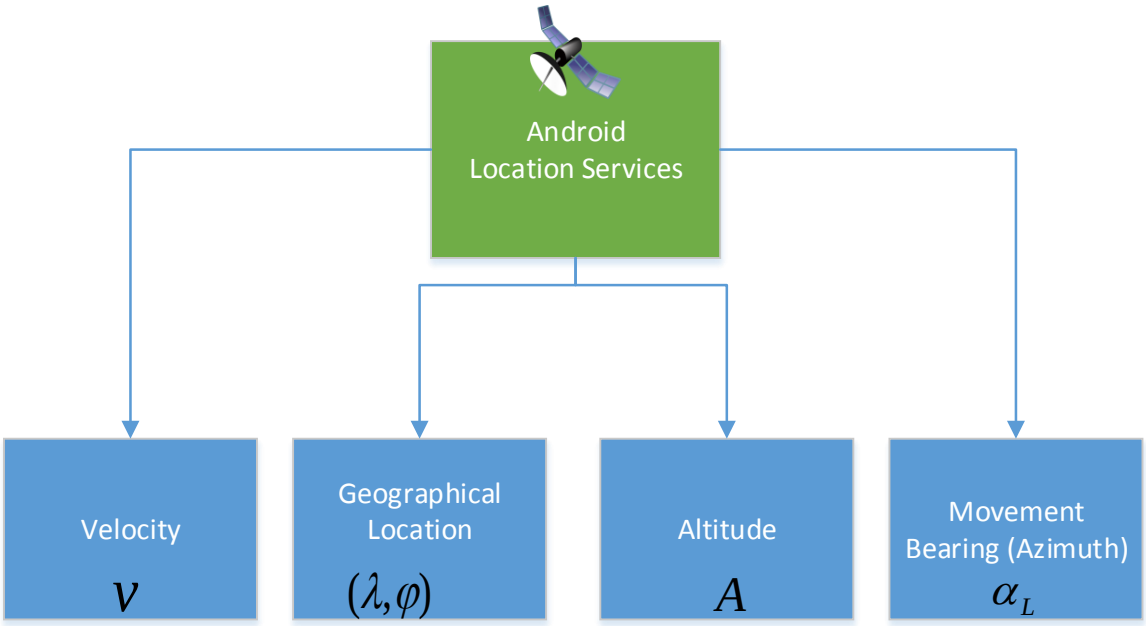

Figure 2.5: Android location services diagram

2) Weather Services

To acquire weather context data, we considered many weather web services with APIs such as Accuweather and OpenWeatherMap. The verdict was that, in comparison to other options, OpenWeatherMap because of its simplicity, easy interpretation, and more importantly being free, OpenWeatherMap was the API of choice.

OpenWeatherMap returns its data in JSON format and offers several different APIs for providing its weather information. Most importantly, weather data for a given location at the current time was the desired function. The set of data returning from the **current weather** API are the followings:

- **Geographic Coordinates Data:** These data are composed of longitude $\boldsymbol{\lambda_t}$ and latitude $\boldsymbol{\phi_t}$ that have already been provided as the input. The city, province, and country, on top of sunrise and sunset data, also fall within this subset of data.
- **Main Weather Data:** Shows your current, minimum, maximum, and temperature. The minimum and maximum temperatures are current temperatures anywhere in that area at the moment and in most cases are usually the same as current temperature.
  - The humidity expressed as a percentage of the total humidity of the air.
  - The pressure of the air.
- **Wind:** Provides the current wind speed $\boldsymbol{v_w}$ and its bearing $\boldsymbol{\alpha_w}$.

- **Snow:** Provides the snow volume ***Snw*** for the last 3 hours, in mm.
- **Rain:** Similarly it is the amount of precipitation falling as rain ***Per*** for the last 3 hours, in mm.
- **Clouds:** Provides the weather cloudiness, which is the percentage of cloud coverage or possibility of cloud cover ***Cld%***.

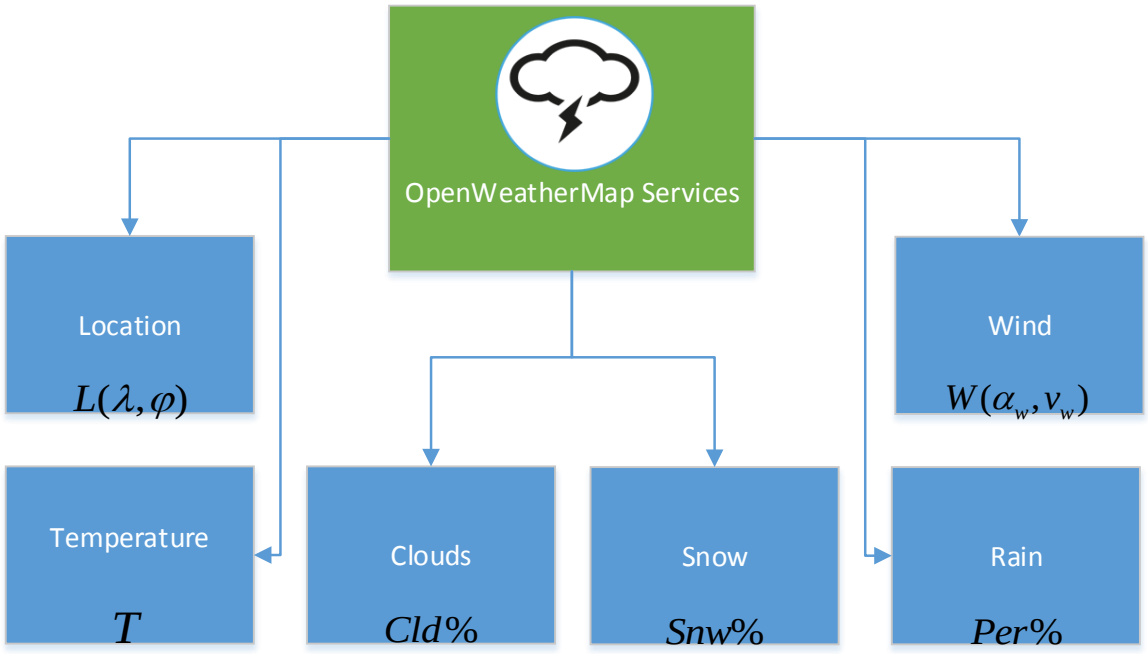

Figure 2.6: Here Map and Location Services Diagram. Here offers speed limit v_l data.

3) Speed Limit Services

Different providers were considered for acquiring these services. Google provides this functionality through its Google Maps Road API. However, it does not offer these services for free as it has been targeted for enterprise applications[3]. Other free alternatives were Wikispeedia the Open Speed Limit Database[4], and ITO Map[5], a global overlay map service showing hidden data layers within OpenStreetMap. Although both provide reasonably good support for the United States and the United Kingdom, none of these provide speed limit information for Canada.

We used Here, formerly NAVTEQ, from Nokia, which offers mapping and location data, including speed limit data offering them in JSON format. It provides the desired input given a possible geographic coordinate, such as for $L(\lambda, \phi)$, it provides the speed limit of that given location, v_l in m/s.

[3] Google Maps Roads API, retrieved from https://developers.google.com/maps/documentation/roads/
[4] The Open Speed Limit Database, retrieved from http://www.wikispeedia.org/
[5] Speed limits provider from OpenStreetMap, retrieved from http://www.itoworld.com/map/main

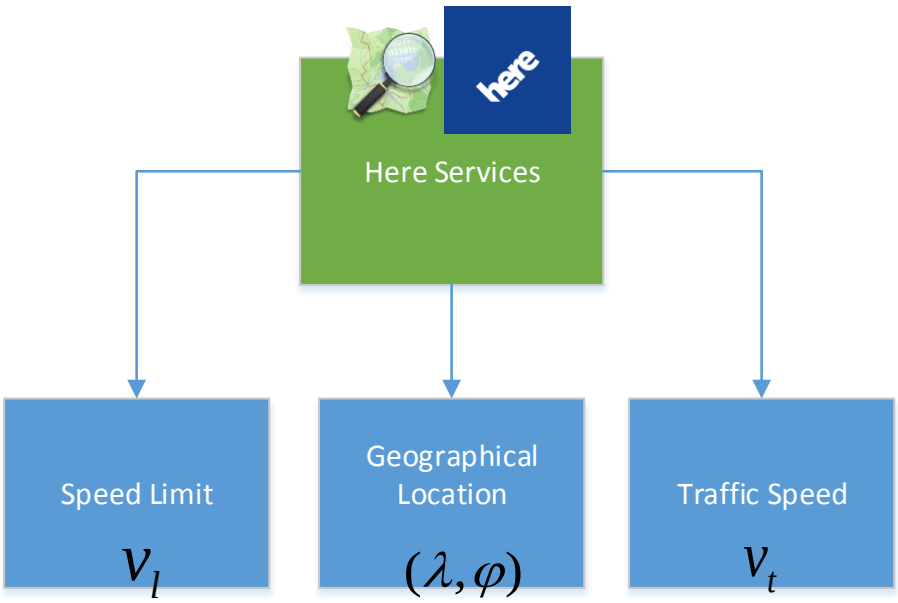

Figure 2.7: OpenWeatherMap Services Diagram

2.1.3 Correction of Sensor Values

While using smart-phone-based sensors, it is important to note that the orientation of the phone is arbitrary with respect to the direction of motion and that it is prone to repeated changes. This can negatively affect the data collected and render it useless unless a correction process is undertaken.

One possible solution to this problem is to fix the phone in a position that matches the orientation of the vehicle, which is inevitably the same as the motion direction. Unfortunately, this is not the case in real-life situations, and no such assumption should be made. Therefore, knowing that the phone orientation changes continuously, constant reorientation of the axes of the phone with respect to the vehicle will be necessary.

The Android documentation mentions that the values received from the following sensors correspond to the phone's coordinate system [7]:

- Acceleration
- Gravity and Linear Acceleration
- Gyroscope
- Geomagnetic Field

The coordinate system of the smart-phone is defined in relation to its device screen when it is held in its default orientation, as shown in Fig. 2.1 on page 7. As a result, the problem after receiving the raw acceleration values from the Linear Acceleration sensor is

the fact that the Android system assumes its values are recorded according to the axes of the phone, not the world around the device or the device-holder.

The proposed solution to this reorientation issue is introduced in the following section by first defining the rotation matrix and then explaining how it is employed in the solution.

Rotation Matrix: To reach a different coordinate system, a rotation matrix R is needed. Since the default orientation of the smart-phone is astatic and not entirely reliable, sensor values, which are based on the coordinate system of the phone, need to be converted to the coordinate system of the world. Doing so requires a rotation from the phone coordinate system to the global fixed frame of reference.

In order to do a rotation in three-dimensional Euclidean space it is required to do find a rotation matrix R that performs the rotation of the values for us [44].

1. Decomposing a $3 * 3$ rotation matrix, we have a general matrix formula that is composed of three separate Rotation formulas from three axes of x, y, and z.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = R_x R_y R_z \tag{2.2}$$

   The Euler angles used for rotation are the following (atan2 is the same arc tangent function, with quadrant checking consideration):

$$\theta_x = atan2(r_{32}, r_{33}) \tag{2.3}$$

$$\theta_y = atan2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \tag{2.4}$$

$$\theta_z = atan2(r_{21}, r_{11}) \tag{2.5}$$

2. Given the three Euler angles of θ_x, θ_y, and θ_z, the rotation matrix is composed as follows:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) \\ 0 & sin\theta_x & cos\theta_x \end{bmatrix} R_y = \begin{bmatrix} cos\theta_y & 0 & sin\theta_y \\ 0 & 1 & 0 \\ -sin\theta_y & 0 & cos\theta_y \end{bmatrix} R_z = \begin{bmatrix} cos\theta_z & sin\theta_z & 0 \\ sin\theta_z & cos\theta_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

3. Note that the Euler angles returned will be in the following ranges:

$$\theta_x : (-\pi, \pi)$$
$$\theta_y : (-\pi/2, \pi/2)$$
$$\theta_z : (-\pi, \pi)$$

If the angles are kept within these ranges, then the same angles on decomposition will be obtained. If, however, the input angles are outside of these ranges, the correct rotation matrix will still be obtained, but the decomposed values will be different from the original angles. As mentioned in the Android documentation [6], rotation Matrix R can be obtained to transfer a set of values from device orientation to global orientation. The orientation of the world is shown in Fig. 2.8 on page 15.

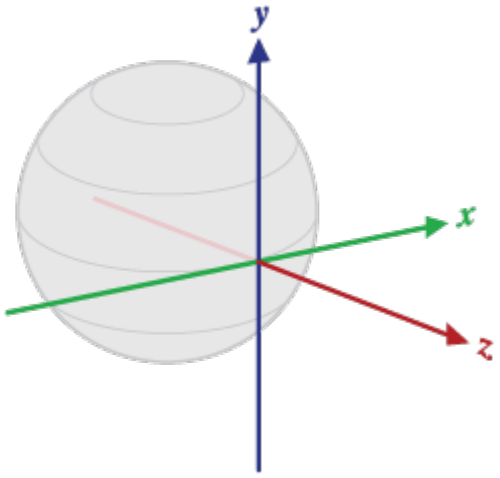

Figure 2.8: World Coordinate system used by the Android Sensor Framework [5]

Rotation of Sensor values to Vehicle FoR: In order to reach the Vehicle FoR, two sets of rotations are required. Firstly a rotation towards the earth fixed FoR is attempted. A set of sensor values are needed: those of the phone (in our case the acceleration values), gravity values and geomagnetic values. The Android documentation APIs already provide this feature through the getRotationMatrix function[6]. Through this function, the rotation matrix R_{pe} is acquired. Because of this, no detail will be given here.

Up to this point, values have been rotated to Earth FoR. Next, these values are transformed to vehicle FoR. The movement bearing α_t coming from the location service is the angle of rotation.

The earth's FoR in Android is based on magnetic north (not to be confused with real north). To find the actual rotation degree, first the direction of movement or bearing α_t of the car needs to be known.

In land navigation, bearing or azimuth is calculated in a clockwise direction starting from a reference direction [11]. If the reference direction is north (either true or magnetic), the bearing is referred to absolute bearing. There are two kinds of absolute bearing [11]:

1. True bearing is measured in relation to the fixed horizontal reference plane of true north–that is, using the direction toward the geographic North Pole as a reference point.

2. Magnetic bearing is measured in relation to magnetic north, using the direction toward the magnetic north pole as a reference.

The magnetic declination δ in Fig. 2.9 is the angle on the horizontal plane between magnetic north (the direction of the earth's magnetic field lines) and true north (the direction along a meridian towards the geographic north pole). This angle varies depending on one's position on the earth's surface, and also over time.

The Android geomagnetic services also provide us with the magnetic declination that is the declination of the horizontal component of the magnetic field from true north, in degrees (i.e. positive means the magnetic field is rotated east that much from true north) [2].

Therefore, the final rotation angle r_{ev} between magnetic north N_m and bearing is calculated from the following Eq. (2.7):

$$r_{ev} = \alpha_t - \delta \tag{2.7}$$

Given the rotation angle of r_{ev}, the rotation matrix R_{ev} is calculated as in Eq. (2.6)

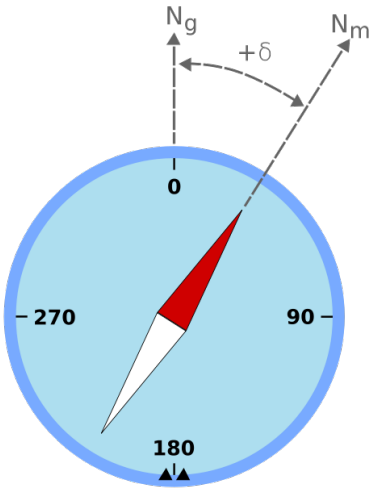

Figure 2.9: Magnetic declination angle δ between true north N_g and magnetic north N_m.

2.1.4 Classifying and Clustering

For detecting events on time series data, various Machine Learning techniques can be employed. In this field, data categorization is an important aspect of data analysis. In unsupervised categorizing also called *clustering*, no previous knowledge or labelled data for training is available [17]. On the other hand, in supervised categorizing, also known as classification, after training the classifier, first we try to map a set of input data vectors to a finite set of input class labels [17].

Two of these algorithms that are later employed in our work are discussed here: k-means clustering and SVM. The first is a clustering algorithm, while the second is a famous classification algorithm.

K-means Clustering: (or Lyold's algorithm as it is often called in computer science) is a square error-based categorizing algorithm. It uses a vector quantization method for cluster-analysis. The origins of this method are in signal processing. K-means clustering partitions the input data into k exclusive clusters [50]. The algorithm behind is an iterative refinement technique. It is described generally as follows:

If provided with a set of observed inputs $(x_1, x_2, \ldots, x_n)$, where each input is a d-dimensional real vector, using this algorithm helps in partitioning the n observations into k sets where $(k \leq n)$

$$S = \{S_1, S_2, \ldots, S_k\} \tag{2.8}$$

so that we have minimized the within-cluster sum of squares.

$$\underset{x}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2. \tag{2.9}$$

where μ_i is the mean of points in S_i.

The steps are the following [50]:

1. Create K partitions amongst your initial observations either randomly or based upon some prior knowledge. Calculate the cluster means set $M = \{M_1, M_2, \ldots, M_k\}$.
2. Assign every observation of the data set to its nearest cluster C_w

$$for(j = 1, \ldots, N, i \neq w, \text{ and } i = 1, \ldots, K)$$
$$x_j \in C_w, if||x_j - m_w|| < ||x_j - m_i||$$

.

3. Calculate the cluster means set again based on current partitions.
4. Repeat steps 2 and 3 until there is no change.

The k clusters of data are extracted.

SVM: Support Vector Machines, are a set of supervised learning models that are used for classification of data based on already labelled training data. The goal of this algorithm is to find an optimal clear margin between categories to be as wide as possible.

For example, in Linear SVM, given n labelled training data with the form of

$$I = \{(x_i, y_i)|x_i \in R^p, y_i \in \{-1, 1\}\},$$

when $i = 1, \ldots, n$ and $y_i \in \{1, -1\}$ are the class labels from the x. Each x_i belongs to a p-dimensional real vector. Our aim is to find the maximum-margin hyperplane that divides the points with different labels. Any hyperplane can be written as the set of points x that satisfies

$$\mathbf{w} \cdot \mathbf{x} - b = 0,$$

where . denotes the dot product and $\mathbf{w}$ the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

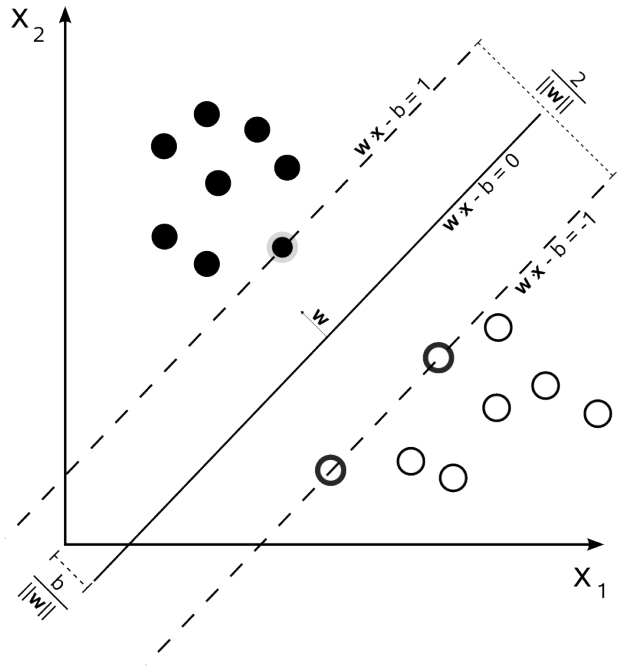

Figure 2.10: Figure showing the margin and the maximum separating hyperplane

In the case of linearly separable training data, two hyperplanes can be selected to separate the data in a way that there are no points between them, thus maximizing their distance. The bounded region by them is called the margin. These hyperplanes can be described by the equations

$$\mathbf{w} \cdot \mathbf{x} - b = 1 \text{ and } \mathbf{w} \cdot \mathbf{x} - b = -1.$$

The distance between the two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, to minimize $\|\mathbf{w}\|$. As we also have to prevent data points from falling into the margin, we add the following constraint: for each i either

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \qquad \text{for } \mathbf{x}_i \text{ of the first class or}$$

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \qquad \text{for } \mathbf{x}_i \text{ of the second.}$$

This can be rewritten as

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n.$$

We can put this together to get the optimization problem:

$$\text{Minimize (in } \mathbf{w}, b)\|\mathbf{w}\| \text{ subject to (for any } i = 1, \ldots, n) y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1.$$

2.1.5 Power Spectral Density

For the clustering and classification of data, finding a suitable feature on data chunks is necessary. For our use-case, we used power spectral density, or PSD. In signal processing, power spectral density is a notation related to periodic sequences. The power spectrum of time series data such as $x(t)$ explains how the data are distributed over the frequency components that $x(t)$ may be composed of. Intuitively, the spectrum decomposes the content of a signal or of a stochastic process into the different frequencies present in the signal and helps identify periodicities.

The energy of a signal over any finite support is always finite and proportional to the length of the support. Therefore we have:

$$\lim_{M \to \infty} \frac{1}{2M+1} \sum_{n=-M}^{M} |x[n]|^2. \tag{2.10}$$

This equation (Eq. (2.10)) represents the signals' average power in time and is also finite. A meaningful spectral representation is obtained for these signals by considering their *Power Spectral Density.* Since the energy of a power signal is finite over a finite length of time, the truncated Fourier Transform

$$X_M(e^{j\omega}) = \sum_{n=-M}^{M} x[n] e^{j\omega n} \tag{2.11}$$

exists, which is finite and its magnitude is the energy distribution of the signal over the time interval $[-M, M]$. The definition of power spectral density is

$$P(e^{j\omega}) = \lim_{M\to\infty}(\frac{1}{2M+1}|X_M((e^{j\omega}))|^2). \tag{2.12}$$

Hence, Eq. (2.12) represents the distribution of power in frequency with units of time over frequency.

Spectral Density Estimation: is the technique used to estimate the power spectral density of a signal from samples' values of the signal over time.

The method used in this work employed periodogram for the estimation of spectral density which is a special case of Welch's method. The periodogram for each window of data is calculated using FFT (fast Fourier transform) as:

$$\frac{1}{N}\sum_{i=0}^{N-1}|\text{fft}(x[i])|^2. \tag{2.13}$$

This equation (Eq. (2.13)) assumes that the frequency-domain FFT result is computed from the time-domain signal data and where N is the number of values in time-domain signal [41].

2.2 Related Work

Similar works that use mobile sensing have previously been proposed and implemented. Their main area of focus has been on traffic management, routing, planning, safety of vehicles and roadways, and emergency services [34, 18]. This section will provide a brief outline of some of these works, including a short summary of each work highlighting their importance and their relevance to our work.

2.2.1 Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smart-phones

Nericell (2008) proposed one of the earliest mobile-sensing solutions for monitoring traffic in cities [34]. It opened the way for others to work on monitoring the task of driving with sensors (with a focus on smart-phone sensors). In this work, the accelerometer, GPS, and the microphone were used to detect driving events such as braking, bumps, horning, and pot holes. For the acceleration sensor, a process of virtual reorientation took place before using the raw sensor values. Additionally, for the results, both well- and re-oriented data were evaluated.

The proposed system used simple threshold and heuristics-based algorithms to detect road bumps, brakes, and other related events. To further identify a noisy and chaotic traffic situation such as an unregulated intersection, the system also used a heuristic-based method to detect honking in the traffic.

As already mentioned, since the focus of this particular this system was on traffic monitoring, it did not provide any measurable indicator or tool in the work for the performance assessment of driving. Furthermore, no specific service was employed in the work, other than the use of a GPS receiver.

2.2.2 Wolverine : Traffic and Road Condition Estimation Using Smart-phone Sensors

Wolverine (2012) focused on road anomaly detection using machine learning [12]. It initially employed K-means clustering for finding the road bumps and brakes from accelerometer data and then used the labelled data for training an SVM classifier to detect brakes and bumps on new input data. The trained classifier was later used to detect events on new data as can be observed in Fig. 2.11. This work inspired us to come up with our idea of the detection of events using similar approaches.

Detection and general road monitoring were the main areas of the focus for this work; therefore, services and other context data were not considered, and driving assessment was not implemented.

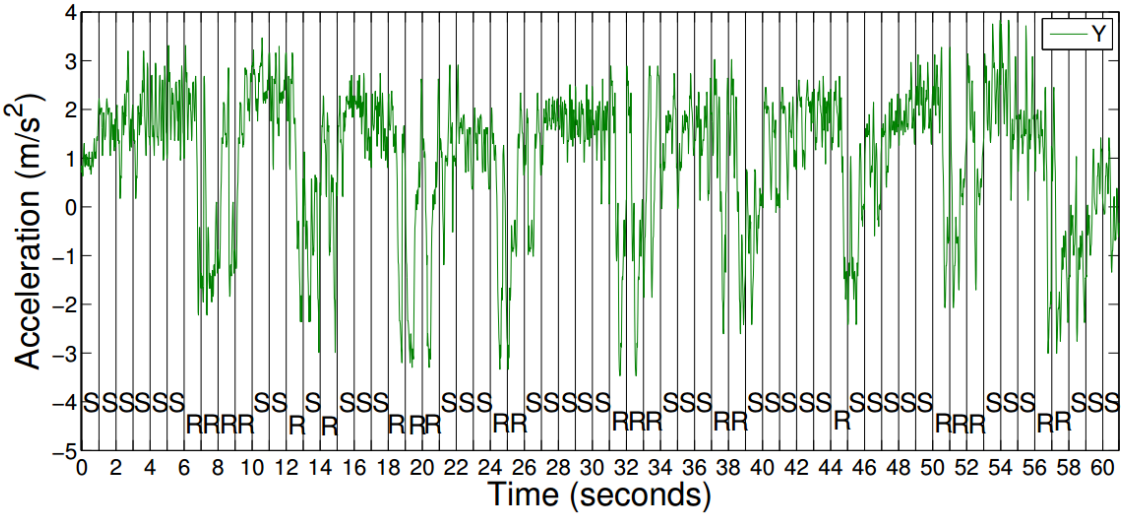

Figure 2.11: Classified Braking Events with SVM Generated Labels in Wolverine for re-oriented y-axis (S is smooth, while R is Braking) [12]

2.2.3 Driving Style Recognition Using a Smart-phone as a Sensor Platform

The MIROAD system developed by Johnson and Trivedi (2011) comes closer to tackling the problem proposed in our work by dividing drivers into two general groups. These

groupings were based on their driving style: aggressive and non-aggressive. This work mentions the fact that the leading cause of most fatal accidents in United States is the former style of driving, the aggressive driving style [26].

The system detected events and processed them within the smart-phone itself. Detected by this system were aggressive and non-aggressive turns, aggressive line changes, excessive speeding, and excessive accelerations. However no services other than the Location Service was used, nor was a driving assessment component provided. Additionally, dividing the characteristics of drivers into only two categories of aggressive and non-aggressive driving is an over-simplification in assessment and not very sophisticated in nature.

Their system used smart-phones in a fixed position as demonstrated in Fig. 2.12.

Figure 2.12: MIROAD system in mounted position [26]

2.2.4 Driving Behaviour Analysis with Smart-phones: Insights from a Controlled Field Study

Paefgen and Zhai (2012) assess driving behaviour from the in-vehicle's acceleration measurement while comparing their results with the car's OBD-II through correlation of the derived results [37]. Assuming that smart-phones are a potentially less reliable sensing platform because of their noises, they tried to evaluate them against the car's OBD-derived data.

To evaluate the study, they assessed driving behaviour based on the frequency of critical driving events in their framework as shown in Fig. 2.13, and they come to conclusion that mobile-based measurement tends to overestimate driving events. The critical events detected were acceleration, braking, and turns by employing threshold-based approaches.

Nonetheless, they focused only on the above-mentioned events using an accelerometer, gyroscope, and GPS sensors for the assessment, and they did not take any other context data or services into consideration.

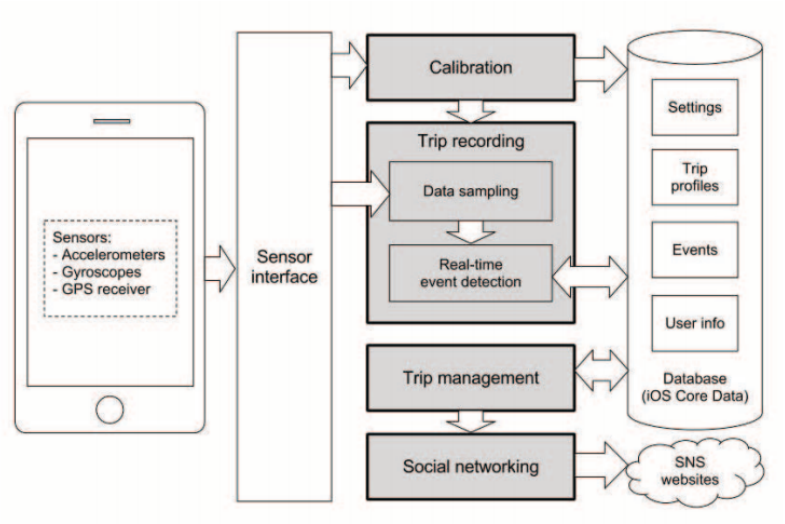

Figure 2.13: Application Overview for Driving Behaviour Analysis with Smartphones [37]

2.2.5 Driving Behaviour and Traffic Safety: An Acceleration-Based Safety Evaluation Procedure for Smart-phones

In another related work by Vaiana et al. (2014), the authors developed a prototype mobile application for analyzing driving quality and its assessment by matching the driving events to a model of being either aggressive or non-aggressive, which was later used for further assessment [45]. Events such as excessive speeding, sudden and intense braking or accelerations, intense turnings, and aggressive U-turns and lane changes were detected by using only GPS in this study, with no accelerometer being used in this work. Then it would send warnings when it is possible and convenient for users to correct their aggressive style of driving [45].

While a very good work in terms of assessing driving, the presented methodology has shortfalls as it has only considered longitudinal and lateral acceleration. No other contributing factors were investigated or employed.

2.2.6 WreckWatch: Automatic Traffic Accident Detection and Notification with Smart-phones

White et al. (2011) employed a similar methodology of threshold-based event detection to detect an accident by finding patterns on the phone sensors, mainly the accelerometer and other context data, to detect an accident and notify emergency services through the phone

(see Fig. 2.14), thus reducing the time between occurrence of an accident and dispatch of emergency responders [48]. The WreckWatch prototype is a client/server smart-phone application that acts as an accident/event data recorder by recording the related data as shown in Fig. 2.14. These data include the path, speed and forces of acceleration during the acceleration [48].

The paper is highly interesting for its novel approach to a new problem. Many parts of its architecture inspired us in giving us ideas for building our framework. However, it did not seek to solve the problem as we are trying to solve in our work.

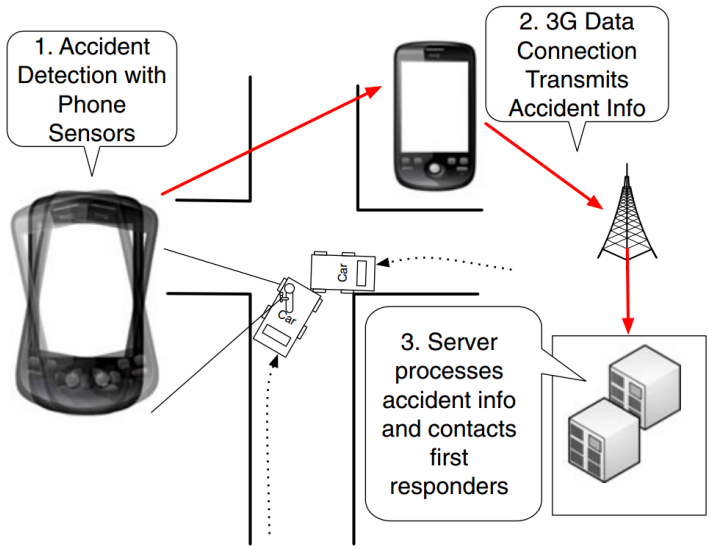

Figure 2.14: WreckWatch's smart-phone-based accident detection system System [48]

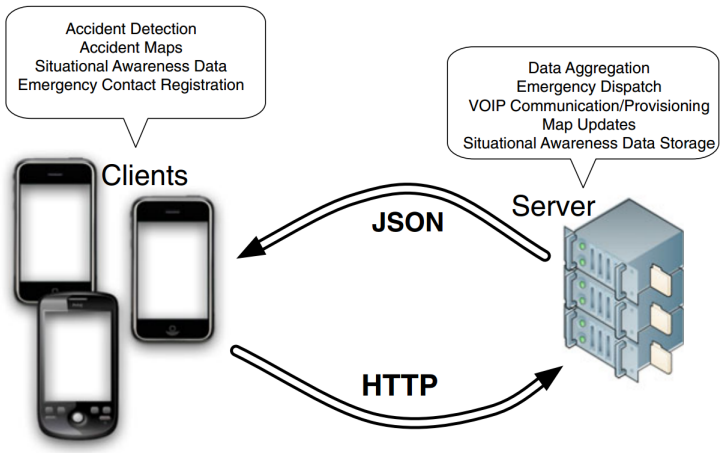

Figure 2.15: WreckWatch architecture diagram [48]

2.2.7 Safe Driving Using Mobile Phones

In their paper, Fazeen et al. (2012) used a road anomaly detection for classifying safe driving events [18]. The authors detected events from the three-axis accelerometer sensor of Android phones to record drivers' behaviour and road conditions. Further they took those data and then performed an analysis to provide useful information for the public and other affected parties [18].

The authors stressed that their work is more in line with the new strategy among car manufacturers, moving away from an emphasis on passive safety approaches such as air-bags and seatbelts to more active ones such as lane-departure warning systems and collision avoidance systems. A wider range of events, most notably lane changes and road classification system such as rough road, uneven road, and smooth road were classified and detected in their work, which made it unique amongst similar works [18].

2.3 Summary

Given the overview of the relative literature work, it can be observed that in most cases, the authors have employed a set of sensors to detect events that are relevant to their intended application. Most of these applications were: traffic monitoring, road anomaly analysis and driving assessment. Road anomaly analysis includes the detection of holes or uneven roads. Driving assessment is mainly recognized by judging driver behaviour as aggressive and non-aggressive.

Yet the main gap within similar works is the lack of a comprehensive performance measure that takes into account both sensors and services data. Driving performance measurement is more nuanced and cannot be effectively captured through a binary grouping of aggressive and non-aggressive driving habits. There are more factors to consider; therefore, we propose a comprehensive framework that not only considers these sensor-based events for measurement, but takes into account service and context data as well.

We employed machine learning techniques for event detection by considering different feature of data in comparison to mainly threshold-based approaches used in similar works. Machine learning has shown to be more adaptable and robust in dealing with errors or outliers [21, 12]. Our results demonstrate that our proposed algorithm has better accuracy with higher F_1-measure in comparison to similar works[34, 12].

Research has shown that weather data, time of driving, driving duration, and driver information all greatly influence performance. We aim to consider as many factors as we can for our assessment. The main benefit of our framework is its ability to be further extended by new measures if the need arises in the future.

Research	Baseline	Sensors	Approach	Evaluation Method	Application
Nericell [34]	GPS	Accelerometer Magnetometer	Threshold-basedheuristics	Cross correlation of false positive and negative rates	Traffic monitoring Road anomaly Detection
Wolwerine [12]	Human-based Tagging	Accelerometer Magnetometer GPS	K-means Clustering and SVM	Comparison between clustered data and classified data	Road anomaly Detection and Driving event detection
Driving behaviour analysis - ETH [37]	OBD	Accelerometer Magnetometer	Threshold based	Correlation of total events between Car and Phone IMU	Driving Assessment
MIROAD [26]	NA	Accelerometer Magnetometer Gyroscope	Threshold based	Correlation of total events between Car and Phone IMU	Assorting driving events to aggressive/non-aggressive
WreckWatch [48]	Publicly available crash acceleration data	Accelerometer Magnetometer	Threshold-based	Cross correlation of false positive and negative rates	Traffic Accident Detection
Driving Behaviour Safety [45]	GPS	GPS	Threshold-based longitudinal and lateral acceleration	Driving Test Data Analysis	Traffic Safety Aggressive vs. Non aggressive Driving

Table 2.3: Comparison of related works

Chapter 3

Proposed Methodology

Our main contributions in this work are devising an overall framework that detects driving events with a more accurate and robust classification algorithm and also proposing a performance index based on those events and other service data. This framework is composed of a driving event service and an application that uses that service and calculates the driving performance index or DPI, which was named PerfDrive. PerfDrive employs the sensors and services of smart-phones and the Internet to measure driving performance.

In this chapter, first, the architecture of the our proposed framework and its various components are formally introduced. That is followed by an introduction to the set of measures that PerfDrive application employs for detecting events. Then, a detailed description of the framework is provided, including all of its components and subsystems.

We will start with Event Services that contains an elaborate mechanism for driving events detection, which is composed of two components. The first component is the sensor and service fusion component, as can be seen from Fig. 3.1. The second component is the Event Detection component that classifies the data for the detection. Later in the chapter, is the driving performance index section, which covers the performance indicators that positively affect the driving assessment and also the methods that are used to detect and measure them. Finally, we will have a section dedicated to assessing the driving processing by introducing a Performance Index to measure that.

3.1 System Architecture

This section will first set out the overall architecture of the proposed methodology. The intention is to give a concrete overview of the whole architecture, which will provide a better understanding of the system. We will eventually focus on each component of the system and display its own architecture in a top-down order.

As can be seen in Fig. 3.1, the proposed system is composed of three main components, listed below:

1. Fusion component: The role of this component is to conduct a set of related processing on the data that is coming from the sensors and services so that they are ready to be used. We will talk in depth on this subject in this chapter, but the chapter contains many subsections including but not limited to syncing, signal cleaning, filtering, further processing, and finally fusing (with other signals and services).

2. Event Detection component: This component's goal is to detect the events that are of importance to the assessment of driving. We will be using a classification algorithm from machine learning for detection.

3. Assessment component: This component is central to the whole framework. For reaching an overall and reliable driving assessment, we devise a DPI scoring mechanism. This scoring and assessment subsystem will be thoroughly covered later in Section 3.4.

We found it helpful to display the framework in an overall algorithm as well. For this reason we provided alg. 1, that shows the step by step procedure associated with our framework. PerfDrive is the DPI calculator application.

Algorithm 1: MAIN, The Driving Assessment Process

Data: Real-Time Driving Sensor and Services Data
Result: The DPI (performance index) of the driving event

```
initialization of PerfDrive application
initialization of Fusion Component from Event Services
initialization of Event Detection Component from Event Services
while not finished driving do
    collection of sensory and service data
    processing and fusing of data by Fusion Component
    classifying events from fused data
    if detected an event then
        send event to PerfDrive component
termination of Sensor and Service data collection
termination of Sensor-Service Fusion component
finalization of PerfDrive application by calculating the DPI
return DPI
```

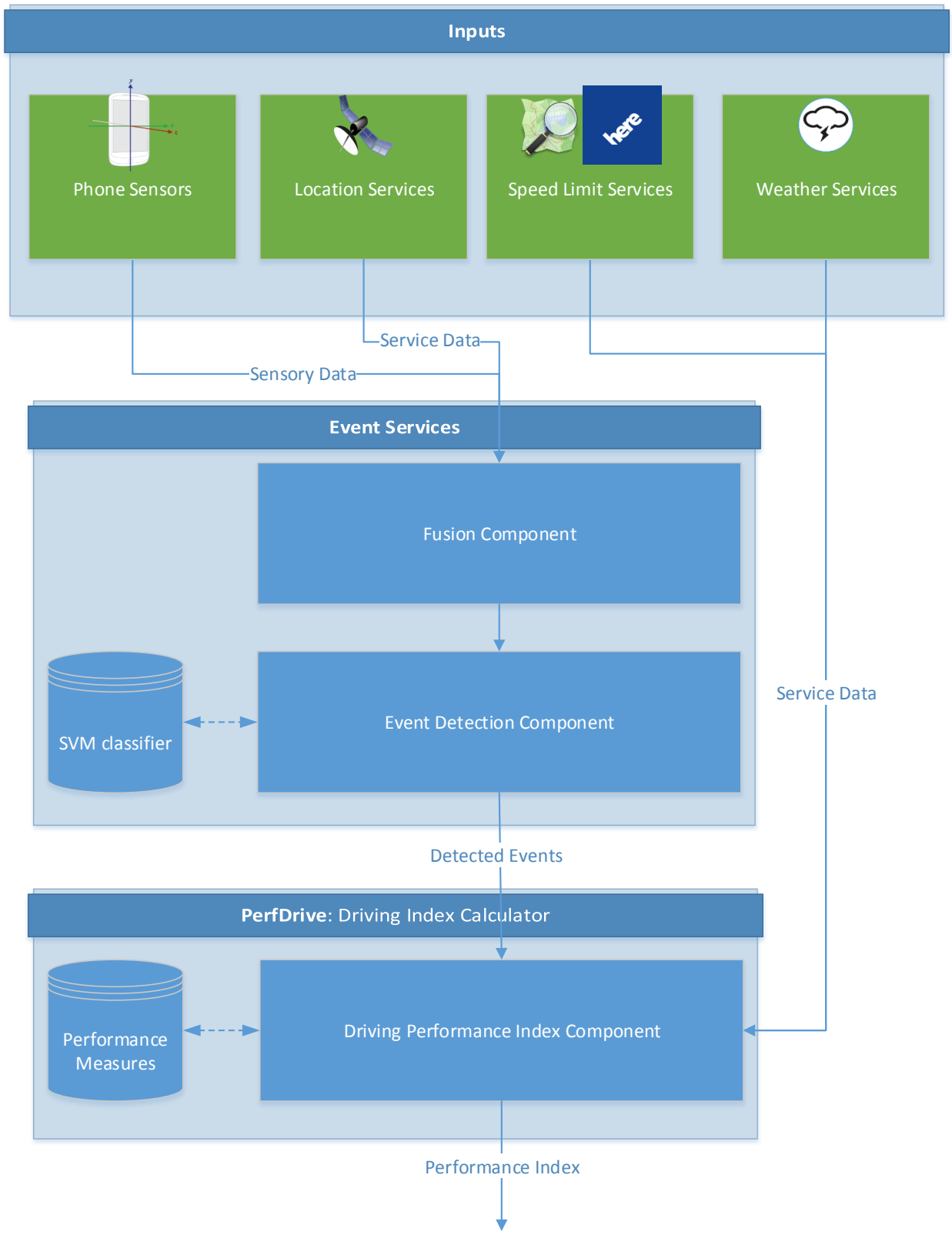

Figure 3.1: **System Architecture.** The three main entities of the framework can be seen here: The set of services and sensory inputs, our proposed event services, and PerfDrive application which calculates the performance index.

3.1.1 Performance Measures

Tracking driving habits and informing drivers about them provides a good incentive to rectify any bad habits and improve driving [24]. It has been shown in Wouters (2000) that providing feedback and records to drivers reduced the rate of accidents around 20% on average [49]. A list of all the factors that affect this assessment is provided later in this section. The intention is to eventually derive an algorithm to measure a cumulative assessment of the driving by performing an evaluation on these performance indices.

After going through the related literature, we were able to provide the following list of measures, which can be used for reaching our performance index. We divided them into to the following categories: *sensed measures*, *service measures* as can be seen in Table 3.1 and Table 3.2 and *driver measures*.

First in Table 3.1, the events that are considered in the existing literature when assessing driving quality are listed. These events can furthermore be measured by sensors, as displayed above. Three main events are considered, as follows:

1. Brakes and car acceleration, measured by longitudinal acceleration values (y-axis); acc_y.
2. Turns that are measured similarly by lateral acceleration values (x-axis); acc_x.
3. Speeding, measured by referring data to the speed-limit facts.

Similarly, Table 3.2 displays other contributing factors that are not received directly from sensors on the smart-phone. The main source for these factors are the services, or more importantly web-services, that offer these values to the users as well as the driver.

Sensed Event	**Input**	**Detection Method**	**Accuracy F_1-score**	**Baseline**	**Detection Reason**
Deceleration and Acceleration	Acceleration Sensor (Longitudinal Values)	Classification on PSD_y from acc_y	0.930	Clustering on on PSD_y from acc_y, Location-based data (GPS)	The frequency and magnitude of decelerations (brakes) and accelerations are amongst the main indications of driving assessment according to the SAS Institute [24].
Turning Intensity	Acceleration Sensor (Lateral Values)	Classification on PSD_x from acc_x	0.904	Clustering on on p_x from acc_x, Location-based data (GPS)	Cornering is another important factor in assessing drivers, in particular by measuring intensity and duration, and noting how drivers manoeuvre [24].
Speeding	Location services, speed limit services	Checking whether the vehicle exceeded speed limit v_l.	N/A (Depends on the accuracy of the provided speed limit and location data)	N/A	Speeding-related accidents is one of the main causes of fatal crashes in Ontario according to the 2011 ORSAR report [36]. Speeding also accounts for 11% of total accidents and 29% of fatalities on roads in the United states in 2012 [16].

Table 3.1: Table of Events Sensed for Driving Assessment

Factor	Detection Input	Reason of Use
Weather	Weather services	One of the environmental factors affecting collisions, with an annual $1 billion weather-related collision costs in Canada [10]. Bad weather can cause poor visibility and road surface friction, increasing the accident rate [9, 10].
Time of Day	Time services	The driving Performance starts getting impaired between 22:00 to 10:00 and also impaired again for a short time in early afternoon. The worst reaction response rate of drivers occur around 2:00, 6:00 and 14:00 [30]. At night, the fatal crash rate among teenagers is almost four times as high than it is during the day and almost half of fatal crashes for teenagers occur between 21:00 and 6:00 [19].
Driving Duration	Time services	The long durations of drive and short previous breaks can negatively affect driving performance [39]. It is suggested that maximum number of driving hours should not exceed more than 6 hour per day, the higher the chances of having an accident [39].
Use of Smart-Phones	Phone services	Using phones in general is linked to a greater risk of accidents. People who use messaging services or talk while they are driving are more prone to accidents. Impermanent distractions such as phones, smoking, eating and vehicle passengers have been linked to unsafe driving [10]. Also using mobile-phones while driving can cause safety implications especially when a hand-held unit was being used by drivers [23].

Table 3.2: Table of Measures Taken from Services Used for Driving Assessment

3.2 Fusion Component

This section focuses on several layers of processing that is conducted on the raw sensor data in order to acquire the needed input for the Event Detection component. As in this section, a series of activities takes place to make the raw data suitable for use with the other components. The block diagram in Fig. 3.2 displays all the processing that takes place on the data in this component.

As mentioned before, the phone orientation or FoR on which the sensor values are based is normally arbitrary in respect to the vehicle FoR. To compensate for this, a series of rotations is required on the phone's default axis data.

1. Rotation toward earth FoR
2. Rotation toward the vehicle FoR
3. Finally another rotation for mitigating the effects of road inclination

The first and second rotations have already been covered in the background chapter; therefore, the third rotation which is unique to our work, will be covered here.

3.2.1 Rotation Toward the Road Inclination

A minor problem with the re-calibrated Linear Acceleration values is that in every step they consider the effects of gravity and magnetic fields to set the rotation toward the earth's frame of reference. This will not consider the side effect of road inclination, which is still apparent in the second round of rotation based on the movement bearing. Therefore, another round of rotation is necessary to counter that effect.

To mitigate these effects, altitude changes in all consecutive data coming from the GPS must be checked. To find the degree of inclination, having some extra information about the road is necessary.

As discussed in the previous chapter, the data fixes arriving from Android location services contain altitude values A with them. The deviation of A over consecutive coming data fixes' time is required, so we can calculate the current value of rise, as can be seen in Fig. 3.3.

However we are more interested in finding the angle of inclination α

$$\alpha_t = \arctan \frac{\frac{\Delta A}{\Delta t}}{\Delta x}. \tag{3.1}$$

Where Δx is the distance between two consecutive location fixes coming from Location Services. For calculating this, we used "haversine" formula, which gives us the great-circle distance between the points [47]. First, let us introduce these variables that are going to be used in this formula:

1. ϕ_1, ϕ_2: The latitude of two consecutive points 1 and 2
2. λ_1, λ_2: The longitude of those points
3. r: The radius of the sphere (here this is Earth, which is 6371km)
4. Δx: The distance that we want to find

The haversine formula is defined as follows [47]:

$$\text{haversin}\left(\frac{\Delta x}{r}\right) = \text{haversin}(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)\text{haversin}(\lambda_2 - \lambda_1) \tag{3.2}$$

Where *haversin* function is defined as:

$$\text{haversin}(\theta) = \sin^2\left(\frac{\theta}{2}\right) \tag{3.3}$$

Where θ is an angle in radians.

As you may have noticed, Δx is nested within the function so to extract it we need to use the inverse haversin function given by:

$$\text{haversin}^{-1}(\theta) = 2\sin^{-1}(\sqrt{\theta}) \tag{3.4}$$

Now applying this knowledge we can give a full formula for Δx:

$$\Delta x = 2r\sin^{-1}\left(\sqrt{\sin^2(\frac{\phi_2 - \phi_1}{2}) + \cos(\phi_1)\cos(\phi_2)\sin^2(\frac{\lambda_2 - \lambda_1}{2})}\right). \tag{3.5}$$

Further, the resulting Δx from Eq. (3.5) needs to be replaced in Eq. (3.1) so that α_t can be calculated.

Now that we have the angle of inclination α_t, it is fairly simple to use the rotation formula of Eq. (2.2) and Eq. (2.6) to be able to finally mitigate the effect of road inclination on our sensor values. For a better understanding of the inclination angle process, please check the Fig. 3.4.

3.2.2 Cleaning and Filtering

A series of other processes also takes place on our data–namely, synchronizing, cleaning and filtering processes. It is important to clean the data so that no unnecessary data remain. By the same token the synchronization is also an essential step, because without that the fusion is fruitless. For this reason, in our implementations we made sure that no data were left without being properly time-stamped.

On the other hand, conducting filtering on the sensor data, especially acceleration, is also an importance task, because of the noises that exist in the values, which is caused by the vibrations and disturbances of the vehicle. We used a Butterworth filter with a low-pass cut-off frequency of $1Hz$ as mentioned in [26], to remove the effects that are not created by vehicle acceleration, when we needed access to clean data.

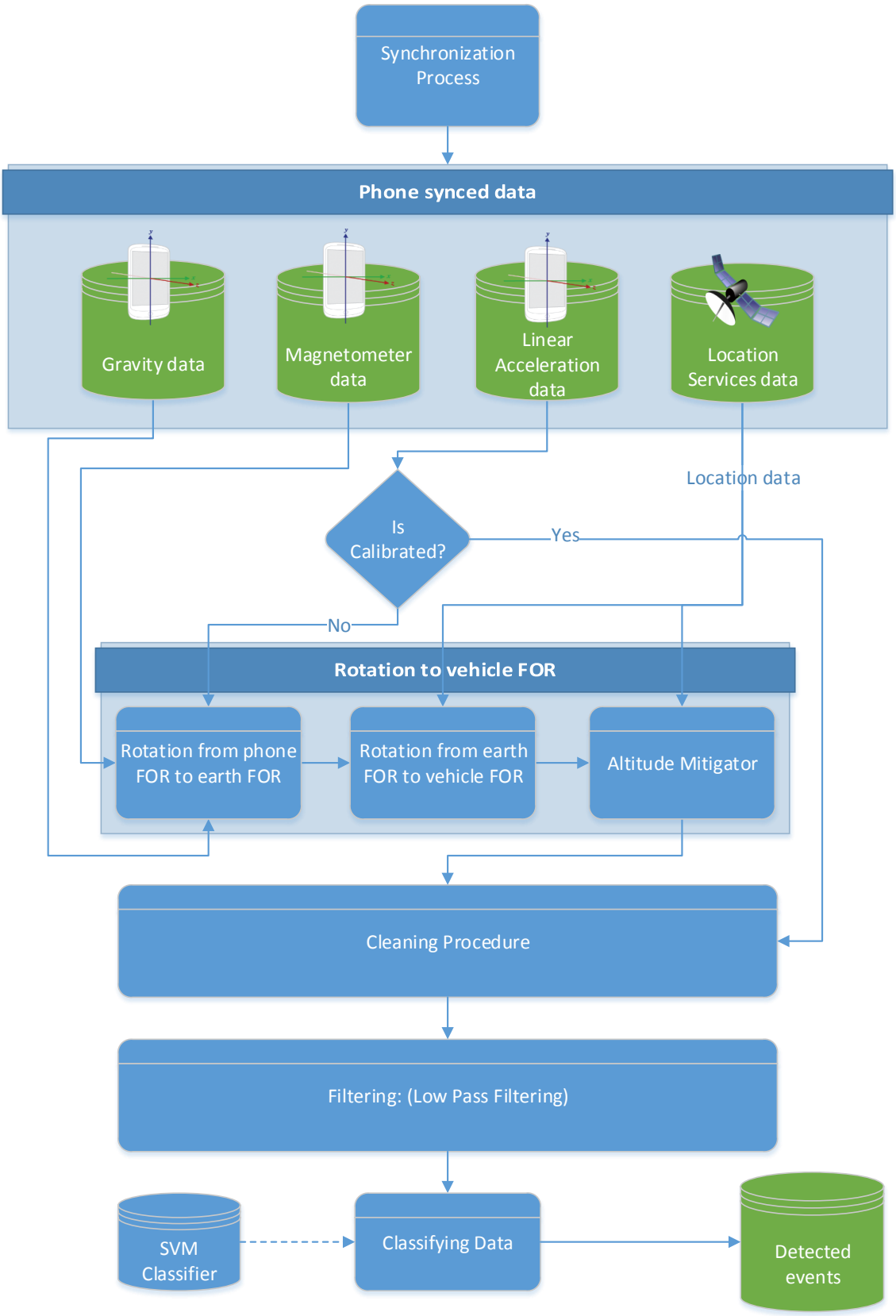

Figure 3.2: Sensors and services fusion component block diagram

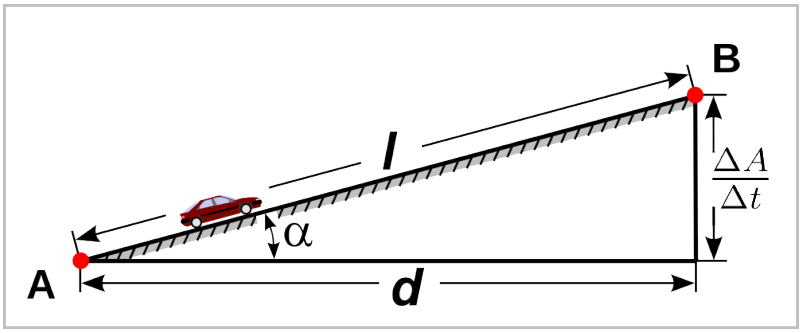

Figure 3.3: Inclination Angle α of two consecutive geographical Points $A(\phi_A, \lambda_A)$ and $B(\phi_B, \lambda_B)$.

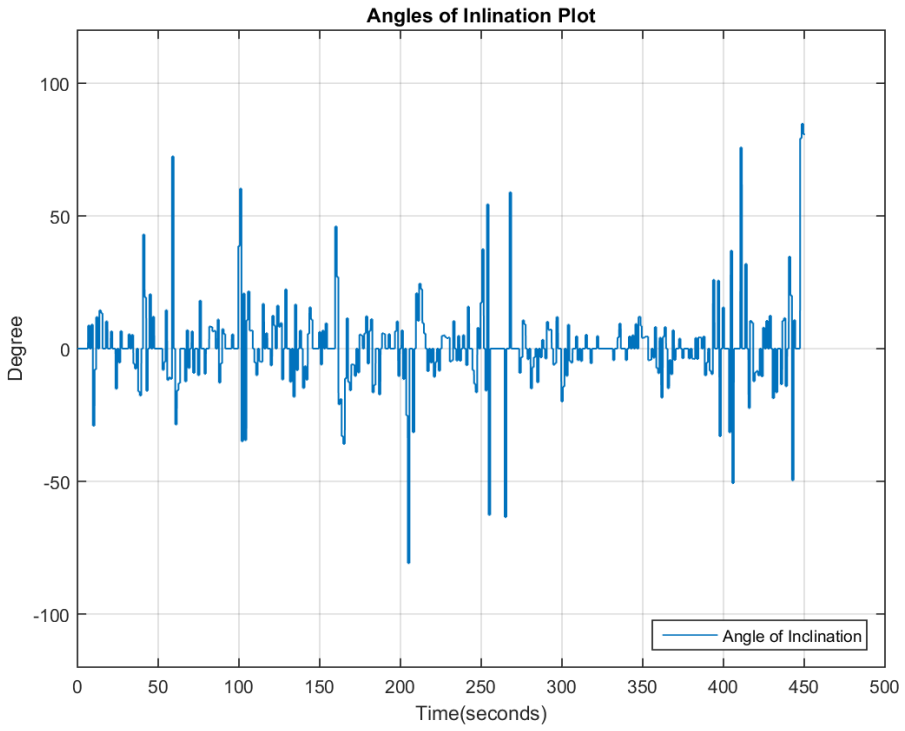

Figure 3.4: Inclination angles over time for sample experiment data. As can be seen, the inclination effect is mostly negligible except at certain points of time where there is a noticeable descent or slope in the road.

3.3 Event Detection Component

This section will discuss the methods that are used to detect the events that leave an acceleration trace on in-vehicle sensing–namely, brakes (deceleration), accelerations, and turns. The frequency, and intensity (sharpness) of these events are an indication of driving quality [37, 18, 45].

For detection and labelling purposes, we divided the sensor data by their time-stamps into windows of two-second length each. Subsequently, the frequencies for phone and GPS were $50Hz$ and $1Hz$. For location updates, we were limited to the Android phone's restrictions because the update rate of location services has an upper limitation of $1Hz$, so the frequency cannot get any higher than that in the phone.[1]

This rate was chosen for the reason of having adequate values to capture the different features of the data chunks. It also should be noted that the actual update rate for Android phones (Nexus 5) varies as it is not a real-time system, so having this limitation, helps with maintaining a constant rate.

3.3.1 Acceleration Detection

After the fusion component has synchronized, cleaned, and applied the FoR correction process on the acceleration and deceleration data, the next step will be event detection. To be able to extract the features from the processed data, we divided the data into windows of two-seconds size or 100 samples with no overlap between them. According to our experiences, a smaller window size does not allow for the precise detection of events. A larger window size, on the other hand, results in delayed detections and higher number of missed events. This division process facilitates the later labelling process.

To approach the detection problem, we used a hybrid method of supervised and unsupervised learning. Unsupervised clustering on appropriate feature of data leads to clusters of events that are of interest in driving assessment and can be used later for training a classifier.

Feature Extraction: Selecting a relevant feature that can make the desired events stand out is the first step of this work. Features that have been traditionally extracted in previous works, were in the time-domain. Examples are mean, median and standard deviation. In our work we focused on frequency domain as an alternative approach. Our results showed that clustering on PSD feature of data can provide better accuracy in comparison to similar works. For calculating the PSD values for each window, we used the PSD estimation Eq. (2.13), previously discussed in Section 2.1.5.

[1] http://stackoverflow.com/questions/17827142/getting-the-highest-gps-update-rate-from-the-gps-hardware-in-my-android

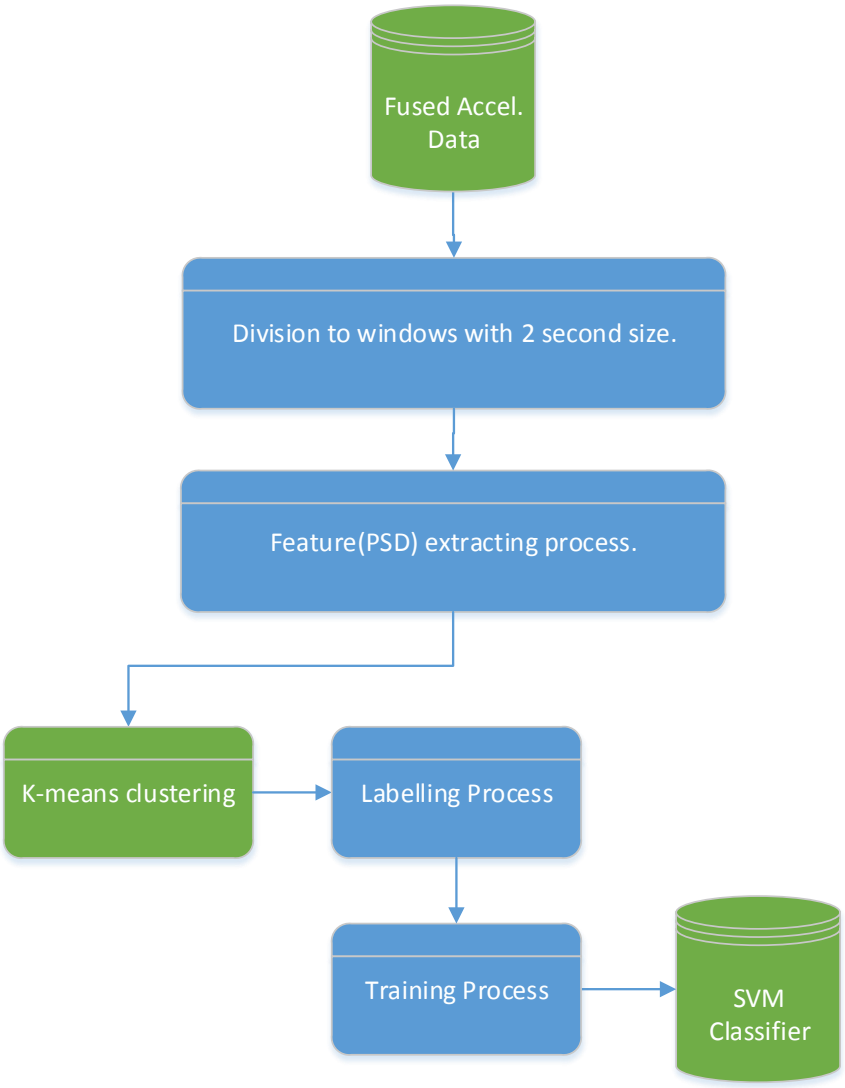

Figure 3.5: The diagram of Event detection component.

Clustering and Training: For detecting vehicle acceleration (or longitudinal accelration) PSD features are excellent for categorizing general acceleration events vs. non-acceleration ones. Therefore, once having been fed into K-means clustering, the sought events are clustered from non-event ones. As an example, in Fig. 3.6, the effect of applying the k-mean clustering on a sample acc_y signal collected using a mobile phone, can be observed. The result of clustering on the p_y of the three-dimensional PSD values for the same example is in Fig. 3.7.

Training the classifier: The extracted p_y features of the clustered data then ran through the SVM algorithm to start the training phase. We used linear SVM and let it train on the clustered data. SVM is one of the most popular classification algorithms and provides very good results for this type of classification. We tried other classification algorithms as

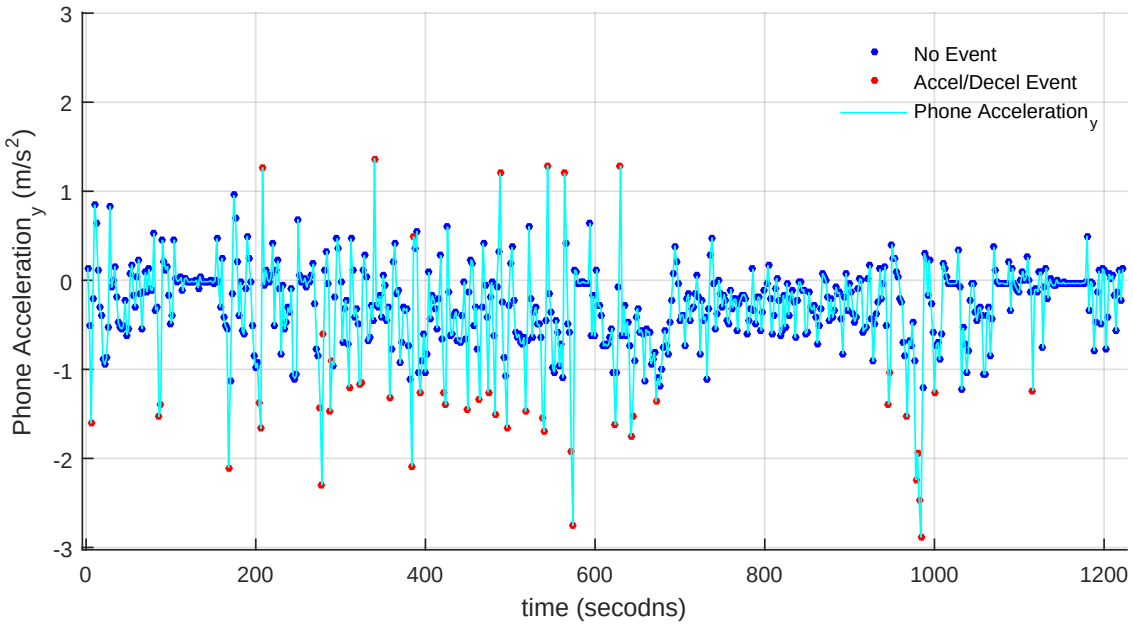

Figure 3.6: The clustered data over p_y of a sample longitudinal signal collected using a mobile phone in a vehicle that detects vehicle deceleration or acceleration events.

well and compared their results with each other. However, SVM provided the best overall accuracy.

3.3.2 Turn Detection

For detection of turn events, we followed a similar approach to that of the above acceleration detection. K-means clustering with $k = 2$ were used again on the relevant p_x *the PSD of lateral acceleration* features of data from acc_x at first. The clustering results shown in Fig. 2.5, clearly made two distinct cluster of turns (right or left) and non-turn events.

We then labelled each cluster manually to prepare our data for training. As with acceleration detection, we then ran the linear SVM classification on the p_y features and let it be trained for future classifications.

The detection of turning were adequate for our study. To further differentiate between right or left turns, yet another level of classification is required. Hence, a left or right turn can be detected from the mean value of the windows, whether if it is negative(left) or positive(right).

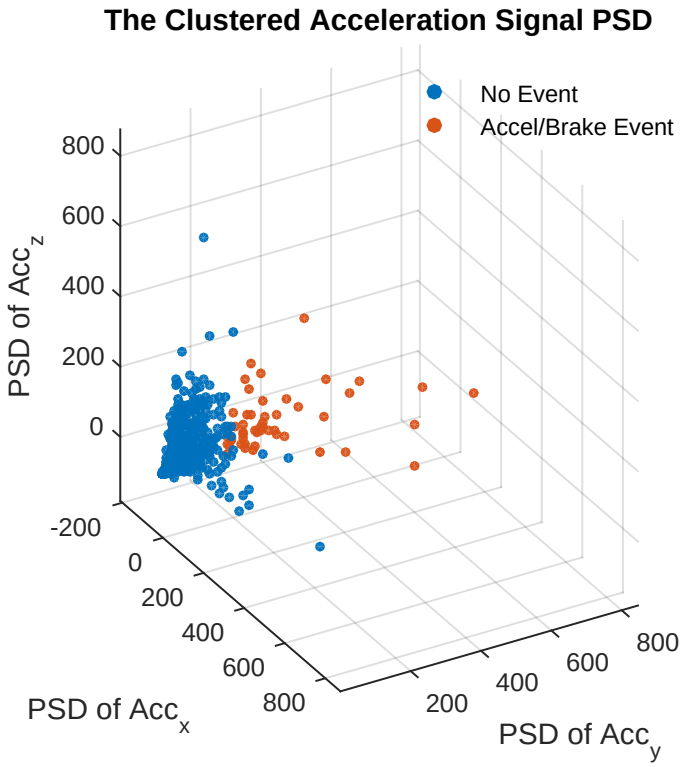

Figure 3.7: The clustered data over p_y on three-dimensional acceleration PSDs.

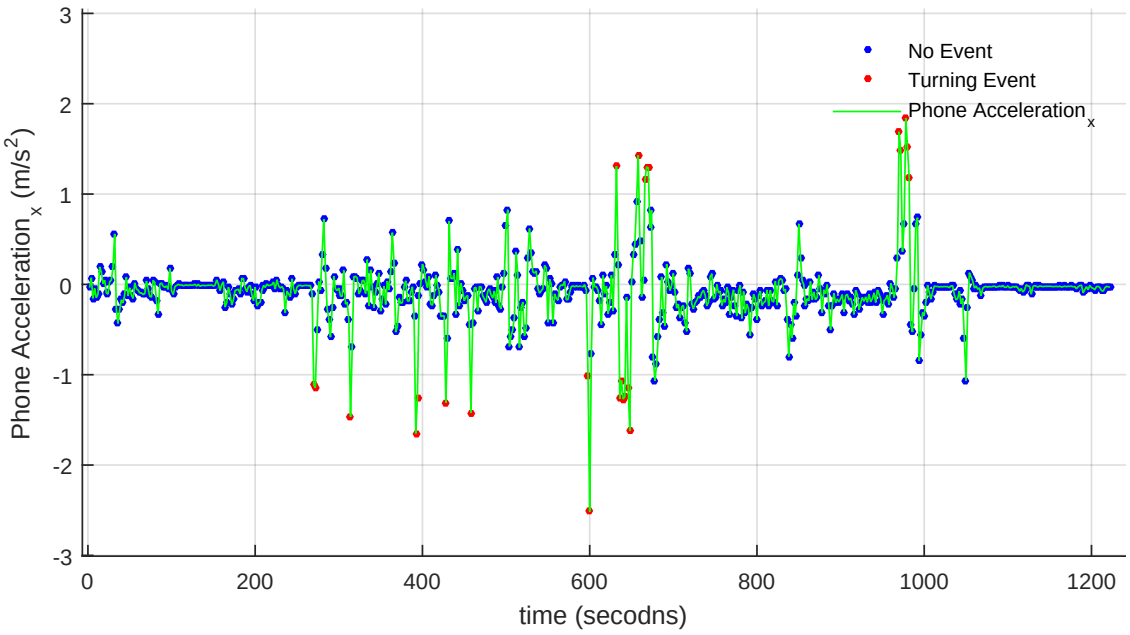

Figure 3.8: The clustered data over p_x of a sample lateral linear acceleration signal collected using a mobile phone in a vehicle that finds turn events for the vehicle.

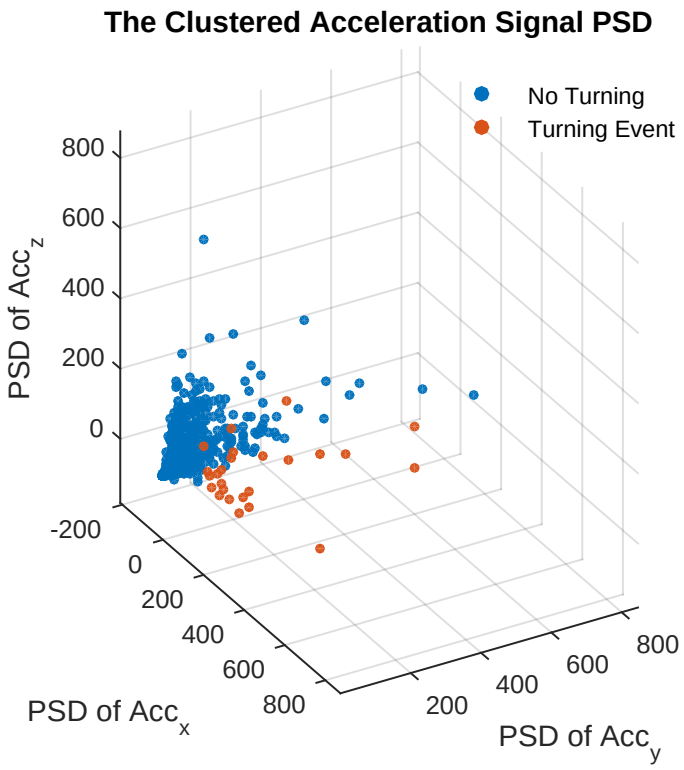

Figure 3.9: The clustered data over p_x on three-dimensional acceleration PSDs.

3.4 Driving Performance Index

In this section, we will propose and discuss one possible way to assess driving performance. The objective of this section is to provide a performance index to assess the process of driving. The fusion of the events detected in the Event Services, into a single index defined in this section, is an example of how these events can be employed in driver assessment.

The goal for defining any such index is to obtain a statistical measure of how the performance of a factor or a set of factors in a specific process changes over a period of time [42]. By observing these changes in the index, performances may be monitored, and necessary further steps may subsequently be taken to improve the process where such improvements are called for, as indicated by the performance index score.

In the first step, we will define the driving process, as a definition is required for any process before a performance index can be designed for it. A formal definition of the driving performance based on the measures previously identified in this work will then be given. The measures in question have been proposed by scientific, industry, and other literature as the most important ones for the evaluation of driving style.

3.4.1 Driving Process Background

The driving process is defined as a set of activities that a driver executes in order to perform a driving event. The definition used, in turn, for a driving event is as follows:

"Use a vehicle V_i to get from geographical point $L_a(\phi_a, \lambda_a)$ to point $L_b(\phi_b, \lambda_b)$, where L_a and L_b can be the same, and the event lasted for more than 60 seconds."

This process contains contexts from the task of driving $perse$, and external ones such as time, location, weather, and the driver herself (for example, her age and driving history).

3.4.2 Definition

A series of steps must be taken to extract the overall Performance Index for driving, or DPI. To reach an optimal level of driving, we are trying to reach a set of goals based on what we discussed earlier in Section 3.1.1. The outcomes desired are:

1. To encourage drivers to drive within the speed limit by increasing their index (lowering their performance level) if they drive faster than the v_l.
2. To discourage drivers from excessive driving, which causes exhaustion and increased risk of accidents. Driving more than 6 hours per day needs to be discouraged.
3. To encourage taking regular breaks while driving. Driving for more than an hour without taking any break, will be penalized.
4. To have drivers consider their environmental impact and to encourage reducing pollution by not using cars excessively–that is, more than that which is required.

Symbol	Unit	Description
$L_a(\phi_a, \lambda_a)$	degree pair	Geographical point A (latitude, longitude)
v_t	m/s	Current speed (Velocity)
v_l	m/s	Speed limit
p	$1-10$ (see Table 3.4)	Performance measure
S_L	seconds	Trip length
S_T	km	Trip duration
Δt	seconds	An event's overall time period
n	N/A	Number of occurrences for the measured event such as sharp decelerations, sharp accelerations, sharp turns, etc.
P	N/A	Set of all performance measures contributing to DPI
DPI	1-1000	Driving performance index or the overall driving score
w	$1-100$	The weight of a performance indicator

Table 3.3: Performance Measures Symbols and Descriptions Table

5. To make sure drivers are discouraged from driving at risky times of the day: from before midnight to sunrise–more precisely, 09:00pm - 5:00am.

6. To be sure that drivers avoid adverse weather such as heavy snowfall or rainfall as much as they can to reduce the chance of weather-related accidents.

7. To discourage drivers from using their smart-phones while driving.

Next, we would like to explain the major work processes that are involved. Let P be a set of all measures composing our total Performance Index, defined as:

$$P = \{p_{decel},\ p_{accel},\ p_{turn},\ p_{speeding},\ p_{weather},\ p_{time},\ p_{duration},\ p_{phone-usage}\}\,. \tag{3.6}$$

This set is composed of all performance measures that overall create the Performance Index for this work. As demonstrated, the elements of the set are the performance indexes of all the main factors that were discussed in Section 3.1.1. As for the DPI, it is defined as follows:

$$DPI = \sum_{i=1}^{n} w_i * p_i. \tag{3.7}$$

$$\sum_{i=1}^{n} w_i = 100. \tag{3.8}$$

The cumulative sum of all weights shown in Eq. (3.8) associated with each performance measure used in Eq. (3.7) equals 100.

3.4.3 P Set

So far, there has been a formal definition given for the set of P and the overall score of DPI for each of its components. Now, in this section each element of this set is defined, which is to say the performance measures that make up the overall score. All of the measures have been normalized over S_L, which is the total duration of the measured driving event in seconds.

- To measure **speeding** the following is used (Eq. (3.9)):

$$p_{speeding} = \sum_{i=1}^{n} \frac{\Delta t_i * \Delta v_i}{S_l}, \quad \text{where } \Delta v_i = v_t - v_l \ > 0. \tag{3.9}$$

In the above formula, n is the number of times that a speeding has occurred, Δt_i is the amount of time in seconds the vehicle was speeding for each occurrence, with Δv_i being the speed difference of current speed v_t from the speed limit v_l, which is a positive value.

- Performance for **sharp decelerations**, **sharp accelerations**, and **sharp turns**, are defined by the following measures:

$$p_{decel} = \frac{n_d}{S_L} \ , \tag{3.10}$$

$$p_{accel} = \frac{n_a}{S_L} \ , \tag{3.11}$$

$$p_{turn} = \frac{n_t}{S_L} \ . \tag{3.12}$$

While in these formulae, the n values are the number of times a sharp deceleration (brake), a sharp acceleration, a sharp turn, or phone-usage has taken place during the trip's length.

- The performance for **phone usage**, is defined by the following measures:

$$p_{phone-usage} = \sum_{i=1}^{n} \frac{\Delta t_i}{S_l} \ . \tag{3.13}$$

Where n is the number of times the phone was used.

- The **weather measure** will give its best value when the weather is calm and moderate. When it is raining or snowing, the value of $p_{weather}$ increases. Since Andrey et al. (2001) mentioned that snowfall has a greater influence than rainfall on accident occurrences [10], snowfall was given a weight two times the amount of the rain's weight.

$$p_{weather} = \sum_{i=1}^{n_s} \frac{\Delta t_i \ Snw}{S_l} + 2 \sum_{i=1}^{n_r} \frac{\Delta t_i \ Per}{S_l}. \tag{3.14}$$

While in the above equation (3.14), the n_s, and n_r are the number of times a snowfall and a precipitation has taken place during the trip's length. Keep in mind that for every one of the above performance indices, increases in value of P represents decreases in performance.

- For measuring performance related to time, we have three indicators: **time**, **duration**, and **breaks**:

$$p_{duration} = \Delta t \ , \tag{3.15}$$

$$p_{time} = \sum_{i=1}^{n} \Delta t_i \ , \tag{3.16}$$

$$p_{break} = 300 * \frac{S_L}{3600} = \frac{S_L}{12} \ . \tag{3.17}$$

While in Eq. (3.15), Δt is the amount of driving in excess of 6 hours in seconds. For p_{time} measured in Eq. (3.16), we add all the drivings' (n) amount in seconds that happened between 22.00 to 10.00 and again between 13.30 to 14.30. In Eq. (3.17), we are penalizing 5 minutes (or 300 seconds) for every missed hourly break.

3.4.4 Calculation of Performance Index

This portion will outline the performance measures calculation and overall performance index of the driving process; the methodology here is based on the Eastman Kodak Safety Performance Index [27, 42]. In this method, the performance ranges of metrics are mapped into an overall index [27]. The development process is explained in 10 steps , which have been modified here to match the objectives of this thesis.

Step 1. First all the indicators that have been detected for developing the index should be listed (the result can be seen in Table 3.5 left column)

Performance Level	Level Description
1	Stretch Goal, Most ideal
2	Excellent Performance
3	Goal, Very Good Performance
4	Upper Intermediate Level
5	Intermediate Level
6	Lower Intermediate
7	Average Level
8	Below Average
9	Very Poor Performance
10	Least Ideal, Dismal Performance

Table 3.4: Performance Descriptions for All Levels of Measures

Step 2. For every performance indicator, determine its importance by assigning an appropriate weight to it, with a total sum of 100 (the result can be seen in Table 3.5, far right column).

Based on the related literature on the effects of different measures on driving performance, we gave the following weights to each indicator:

- Since speeding is a major cause of accidents; therefore, we assigned a large weight of 20 to this.
- For phone-usage, accelerations, and decelerations, we used a heuristic-based measure that take into account the frequency of occurrence for each one of these measures.
- Driving duration, over 6 hours is considered bad performance.
- For night time driving, any driving between 22:00 to 08:00 will receive negative points with 02:00, 06:00, 14:00 receiving the worst performance results.
- Weather can be a contributing factor to collisions. We used weight of 10 for it. We were more interested in times when it was snowing or raining, and we took into account the precipitation and snow amount in mm to calculate the measure.

Step 3. Measure with a scaling system from 1 to 10, with 1 being the most ideal score and 10 being the least ideal score, based on Table 3.4. To have the measure be compatible with standard normalized levels, the measures must be mapped to their specific levels. Therefore, we start by determining and putting the average measure of performance of all indicators corresponding to a score in Level 7.

Step 4. Establish a goal for each measure in Level 3.

Step 5. Determine the stretch goal for each measure in Level 1. This goal is reachable only when everything goes superbly [42].

Step 6. Determine the intermediate goals in Levels 4, 5, and 6.

Step 7. Determine the insufficient levels of performance when the measures are worse than their normal amount for Levels 8, 9, and 10.

Step 8. Assign a value to Level 2 to fill all the performance levels.

Step 9. Debug, evaluate, and get feedback from experts and stockholders. After that, review the matrix.

Step 10. The scoring system for PerfDrive is ready.

	Performance Level										
Performance Indicator	1	2	3	4	5	6	7	8	9	10	**Weight**
$p_{speeding}$	0	0.1	0.25	0.5	1.5	3	7.5	10	15	20	20
p_{accel}	0	0.002	0.005	0.015	0.025	0.05	0.1	0.2	0.5	1	5
p_{decel}	0	0.001	0.0025	0.005	0.01	0.025	0.05	0.1	0.25	1	15
p_{turn}	0	0.001	0.005	0.025	0.050	0.075	0.15	0.45	0.75	1	10
$p_{phone-usage}$	0	0.001	0.005	0.01	0.02	0.035	0.05	0.1	0.5	1	20
$p_{weather}$	0	5	10	12	15	18	20	30	40	50	5
p_{time}	0	15	30	45	60	90	120	150	210	270	10
$p_{duration}$	120	280	210	240	260	275	300	360	450	600	10
p_{break}	45	50	60	75	90	105	120	140	160	180	5
Total											**100**

Table 3.5: Performance Index Table Based on the Kodak Safety Performance Index

3.4.5 DPI Scenario

To further clarify the application of this index, a sample scenario is described below:

We would like to measure John's Performance Index DPI_{John} who has used the PerfDrive app on his phone for a test driving session. (The data in this scenario are based on real driving test). John installed the PerfDrive app on his Samsung Galaxy 5 phone. He drove with his Nissan Armada for 450 seconds (approx. 7 minutes), while the app was running in the background. After the end of the driving session, the following measures for each of his performance indicators were achieved:

- **Speeding measure:** As it can be observed in Fig. 3.10, for some portions of drive, John has been speeding; as a result; his $p_{speeding}$ is calculated based on Eq. (3.9). Therefore, the calculated value is $p_{speeding} = 0.363$.

- **Acceleration/deceleration and turn measure:** There exist in total 3 acceleration events (including decelerations) and 5 turn events in which none of them were a sharp deceleration; hence, his scores were $p_{decel} = 0$, $p_{accel} = 0$ and $p_{turns} = 0$.

- **Phone-usage measure:** The phone was used once during the session for messaging that lasted $10s$, while the whole driving session length was 444 seconds, so for him $p_{phone-usage} = 1 * 10/444 = 0.0225$.

- **Duration, break, and time measure:** His driving session started at 13:13:42 and finished at 13:21:07. As can be seen, the car was not used during the driving impaired period (such as late night or around 14:00). Therefore, the time score for him was $p_{time} = 0$. By the same token, since the car was not used for more than an hour, we reached the followings: $p_{break} = 0$ and $p_{duration} = 0$.

- **Weather measure:** Since the sky was clear that day, there was no weather factor to impair the performance, which results in $p_{weather} = 0$.

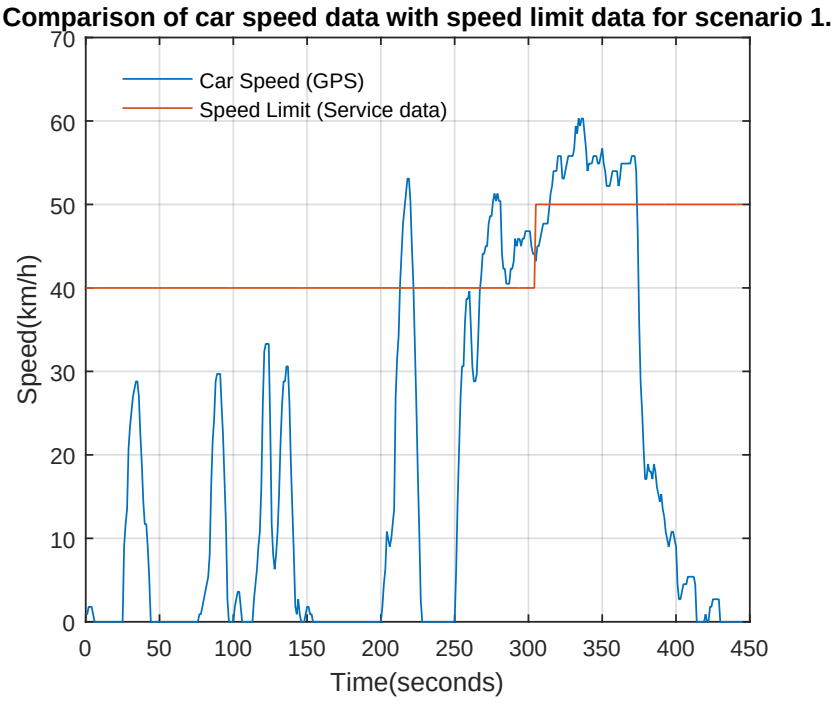

Figure 3.10: Comparison of car speed data with speed limit data for scenario 1.

If the results are mapped to a table, the following Table 3.6 can be constructed. The total score will then be calculated as:

$DPI_{John} = 60 + 100 = 160$ (see score column in Table 3.6).

To understand where this score stand, consider the average Performance Index also known as $DPI_{average} = 700$. $DPI_{average}$ is obtained from the average level (level 7) from Table 3.4 multiplied by total amount of weights that is 100. It can be assessed that in this example, the calculated index is better than the average index, while the specific score for p_{phone_usage} does not show very promising performance.

	Calculations			
Performance Indicator	**Value**	**Performance Level**	**Weight**	**Score**
$p_{speeding}$	**0.363**	3	20	**60**
p_{decel}	**0**	**1**	15	**0**
p_{accel}	**0**	**1**	5	**0**
p_{turn}	**0**	**1**	10	**0**
$p_{phone-usage}$	**0.022**	**5**	20	**100**
$p_{weather}$	**0**	**1**	5	**0**
p_{time}	**0**	**1**	10	**0**
$p_{duration}$	**0**	**1**	10	**0**
p_{break}	**0**	**1**	5	**0**
Total			100	**160**

Table 3.6: Example 1 of Performance Index Calculation Table with Input Values from John's driving session. His total score is $DPI_{Jphn} = 160$, while $DPI_{average} = 700$.

Chapter 4

Implementation and Evaluation

Thus far, the problem of detecting driving related events and a presentation of a comprehensive solution to finding a driving performance index have been discussed. Later, we discussed our proposed solution along with its details and related works. Consequently, in this chapter, the focus will be on the experimental setup, the implementation details of the proposed framework, and its related evaluation.

4.1 Implementation

As most of the event detection happened through phone sensors, we needed to select an appropriate smart-phone. We decided to use Android-based Nexus 5 from Google which is manufactured by LG Electronics [25].

The sensors Nexus 5 features are the following: compass/magnetometer, proximity sensor, accelerometer, ambient light sensor, gyroscope, and barometer. We were mostly interested in accelerometer and gyroscope sensors. The phone uses an MEMS(micro electro-mechanical systems) based accelerometer; Invensense MPU-65xx (also used in Apple iPhone 6 and Samsung Galaxy S5), with a highest sampling rate frequency of $4000Hz$ [53]; However, in reality the Android OS restricts that to $200Hz$ for both the accelerometer and gyroscope to conserve battery power [53, 32].

The PerfDrive smart-phone app was written in Java with Android SDK 24.0.2 release,[1] for recording the data gathered by the phone sensors and services. The captured data during the driving session were then saved and recorded in the device memory at the end of each session.

Most of cleaning, filtering, and minor processing on the sensor data was done later in the driving session data, with scripts written in the Python programming language [46]. Matlab was used later for synchronization, FoR correction, and time windowing. The detection process and later clustering, training, and testing of the data were also implemented in Matlab using its Statistics and Machine Learning Toolbox [31].

[1] Android SDK - retrieved from https://developer.android.com/sdk/index.html

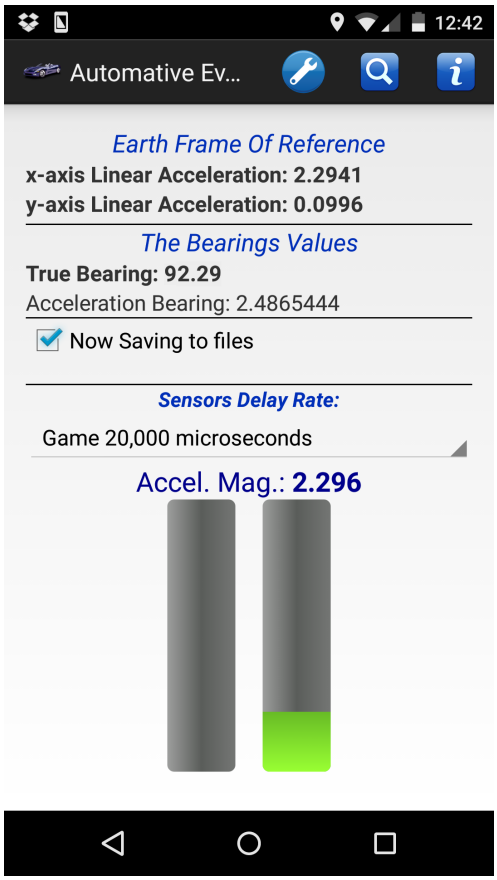

Figure 4.1: A screenshot of the Automotive Evaluator App–PerfDrive sensing and recording data running on Nexus 5 phone.

4.2 Experimental Setup

Two different cars were used for conducting the experiments:

- Honda Civic 2001
- Hyundai Elantra 2015

As can be seen in Fig. 4.4, We considered placing the phone on locations 1 to 6 of the car. We were able to get the best results on locations 4 and 1, and thus, we used them for our experiments. The phone was not essentially calibrated to the vehicle's FoR during the experiments.

The cars were driven for approximately 6 hours around the city of Ottawa at different times of the day under different weather conditions. The sampling rate for weather data was 60 minutes because of the limitation of the OpenWeatherMap services, while for location updates and speed limit data, it was $1Hz$ and for the linear acceleration sensor values it was $50Hz$.

Figure 4.2: Hyundai Elantra 2015 used for the experiments.

Figure 4.3: Honda Civic Coupe Black 2002, another car used in our evaluations.

4.3 Evaluation

The evaluation results are categorized into two parts. The first was an evaluation of the Event Detection component from Event Services with comparison to previous works. The second was a study of the performance index measurement, with details being covered in the second section.

4.3.1 The Deceleration/Acceleration Detection

As discussed in the event detection component section earlier, we followed an approach of combined supervised and unsupervised machine learning. After using the K-means clustering on PSD_y feature of acc_y data, we labelled the acceleration vs. non-acceleration data. Then, we trained four different classifier algorithms for future usage. By using Matlab statistical and machine learning toolbox, we employed four different classification

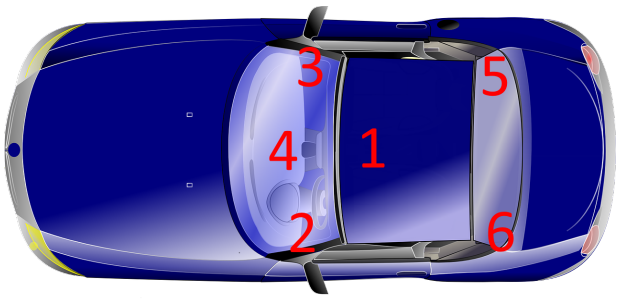

Figure 4.4: Possible locations for placing the phone, with location 1 being the best option.

algorithms on the data:

- Linear SVM
- Naive Bayes
- Classification Trees
- K-Nearest-neighbour

For evaluation, we compared the results of following different algorithms with the K-means clustering results as a baseline. Since SVM classification can be ran for real-time applications while clustering cannot, it was an excellent choice for running on the phone. In total we conducted five experiments on the data.

Next, we compared classified data against the baseline data by calculating their relevance measures. Three standard measures of relevance were used:

1. Precision
2. Recall
3. F_1-score

Recall r or total correct rate is defined as the ratio of correct classified data points divided by the total number of correct points, while Precision p is the ratio of correctly classified data points divided by the total number of classifier assignments [51]. The F_1-score is the harmonic mean of precision and recall values (see Eq. (4.1)) and is a statistical significance test for comparison between the performance of different classifiers [51].

$$F_1(r, p) = \frac{2rp}{r + p}. \tag{4.1}$$

The charts comparing the average results of the four algorithms, can be seen in Fig. 4.5 and Fig. 4.9. SVM had the best results as its F_1-score was 0.9037 which, was better than the other algorithms. The classification trees also had the second best results. Naive Bayes was the third in rating, and at the end of the list was the k-nearest neighbour, with an F_1-score of 0.7924

Clustering and training on PSD_x	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	17	0	26	1	0.654	0.791
N-B	18	1	26	0.994	0.692	0.816
C-T	15	0	26	1	0.577	0.732
KNN	13	1	26	0.994	0.500	0.665

Table 4.1: The results of running different classification algorithms on the experiment data of experiment 1. The feature used was PSD_y of the acc_x axis.

Clustering and training on PSD_y	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	54	0	64	0.844	1	0.915
N-B	45	1	64	0.703	0.978	0.818
C-T	54	0	64	0.844	1	0.915
KNN	48	0	64	0.750	1	0.857

Table 4.2: The results of running different classification algorithms on the experiment data of experiment 2. The feature used was PSD_y of the acc_x axis.

Clustering and training on PSD_y	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	41	0	55	0.745	1	0.854
N-B	28	2	55	0.509	0.933	0.659
C-T	40	0	55	0.727	1	0.842
KNN	41	0	55	0.745	1	0.854

Table 4.3: The results of running different classification algorithms on the experiment data of experiment 3. The feature used was PSD_y of the acc_x axis.

Clustering and training on PSD_y	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	39	0	41	0.951	1	0.975
N-B	28	4	41	0.683	0.875	0.767
C-T	29	0	41	0.707	1	0.828
KNN	31	0	41	0.756	1	0.861

Table 4.4: The results of running different classification algorithms on the experiment data of experiment 4. The feature used was PSD_y of the acc_x axis.

Clustering and training on PSD_y	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	64	0	64	1	1	1
N-B	64	15	64	1	0.810	0.895
C-T	63	0	64	0.984	1	0.992
KNN	64	0	64	1	1	1

Table 4.5: The results of running different classification algorithms on the experiment data of experiment 5. The feature used was PSD_y of the acc_x axis.

Clustering and training on PSD_y	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	48	5	49	0.980	0.901	0.941
N-B	45	12	49	0.918	0.789	0.849
C-T	49	8	49	1	0.859	0.924
KNN	47	5	49	0.960	0.904	0.931

Table 4.6: The results of running different classification algorithms on the experiment data of experiment 6. The feature used was PSD_y of the acc_x axis.

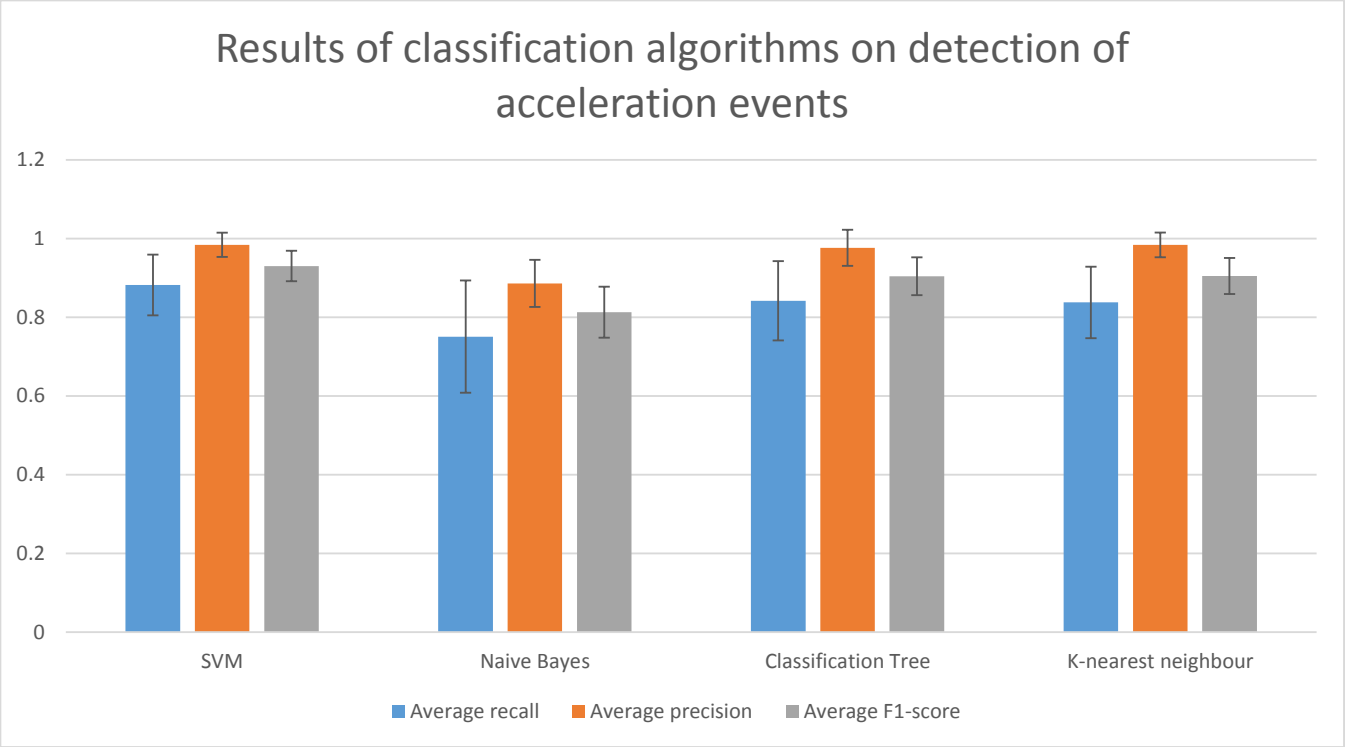

Figure 4.5: Chart of the average results with %95 confidence interval of four different classification algorithms on the PSD_y feature of clustered data from acc_y. SVM had the best results.

Classification Algorithm	**Average precision**	**Average recall**	**Average F_1-score**
SVM	0.882 ± 0.077	0.984± 0.031	0.930± 0.038
Naive Bayes	0.751 ± 0.142	0.886 ± 0.060	0.813 ± 0.064
Classification Tree	0.842 ± 0.100	0.976 ± 0.046	0.904 ± 0.048
K-nearest neighbour	0.838 ± 0.091	0.984 ± 0.031	0.905 ± 0.046

Table 4.7: The average results with %95 confidence interval of the clustered data over PSD_y values from acc_y, vs. the four different classification of those data. SVM had the most promising accuracy result with high levels for brake and acceleration detection.

We compared our work with two similar studies that detect decelerations. The Wolverine study [12] discussed in Section 2.2.2, reached a total correct rate or recall of 0.7838 in detecting brakes by using maximum-minimum difference feature against the windows of data (see Table 4.8). It was not clear weather they ran several experiments, but they mentioned the result from one of their experiments. There were no other experimental data in their work. The F_1-score achieved was 0.8656.

Nericell [34], which was also discussed earlier in Section 2.2.1, attained the best correct rate of 0.845 on a well-oriented smart-phone by using threshold-based approach for classification of brake data. Since they fortunately provided values for different experiments, we were able to get an average of those. Their average F_1-score was 0.8056 (see Table 4.9). Overall, our algorithm shows better results with a higher F_1-measure (Fig. 4.6).

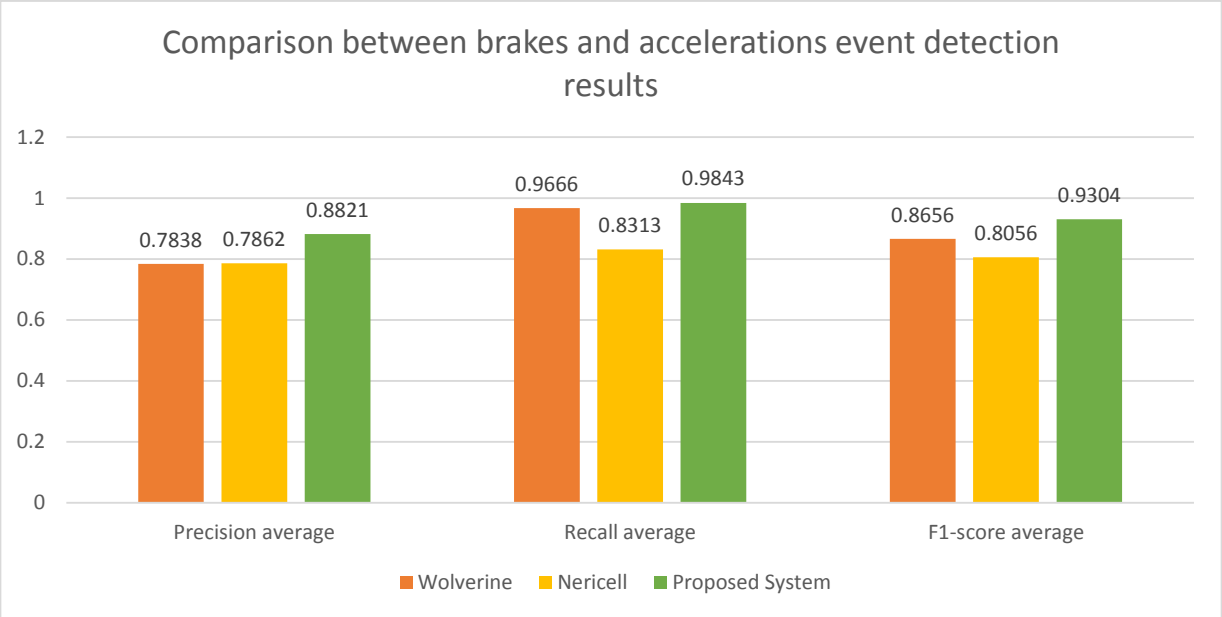

Figure 4.6: Chart of the average results of four different classification algorithms on the PSD_y feature of clustered data from acc_y.

Wolverine - Classification over $\lvert Max_{acc_y} - Min_{acc_y} \rvert$	**True Brake Events (TP)**	**False Brake Events (FP)**	**Actual Brake events (Baseline)**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM	29	1	37	0.783	0.966	0.865

Table 4.8: The results from Wolverine with SVM claasification over the $\lvert Max_{acc_y} - Min_{acc_y} \rvert$ feature of windows of data.

Nericell: Heuristic Threshold	**False Brake Events rate**	**Recall (sensitivity) Correct rate**	**False Negative Rate**	**Precision**	F_1**-score**
Well Oriented Accel sensor - T = 0.11 G	0.44	**0.778**	0.222	**0.946**	**0.854**
Well Oriented Accel sensor - T = 0.12 G	0.111	**0.845**	0.155	**0.884**	**0.864**
Non calibrated Accel sensor - T = 0.11 G	0.44	**0.689**	0.311	**0.613**	**0.653**
Non calibrated Accel sensor - T = 0.12 G	0.111	**0.786**	0.177	**0.881**	**0.851**
Average Results		**0.831**		**0.831**	**0.805**

Table 4.9: The results of the Nericell work on brake detection with well-orieted and non-calibrated smart-phones. The value T refers to the threshold that was used on the baseline data to detect a brake event.

4.3.2 The Turn Detection

For detecting turns, we followed a similar approach by using the PSD_x feature of windows. The main difference this time was on using the acc_x from the x-axis that measures the lateral forces of in-vehicle acceleration. Thus, we then clustered the data into 2 groups by using K-means clustering. A sample of this algorithm on our data is shown in Fig. 4.7.

We then labelled the resulting clusters for turn and non-turn events. Later, the SVM classification algorithm was trained on the labelled data. From this point on we used the SVM trained classification for detecting turns in new data. The same sample in Fig. 4.7 was classified with the SVM, which resulted in Fig. 4.8. As can be observed, the results of the classification algorithm are quite similar to those of the baseline data.

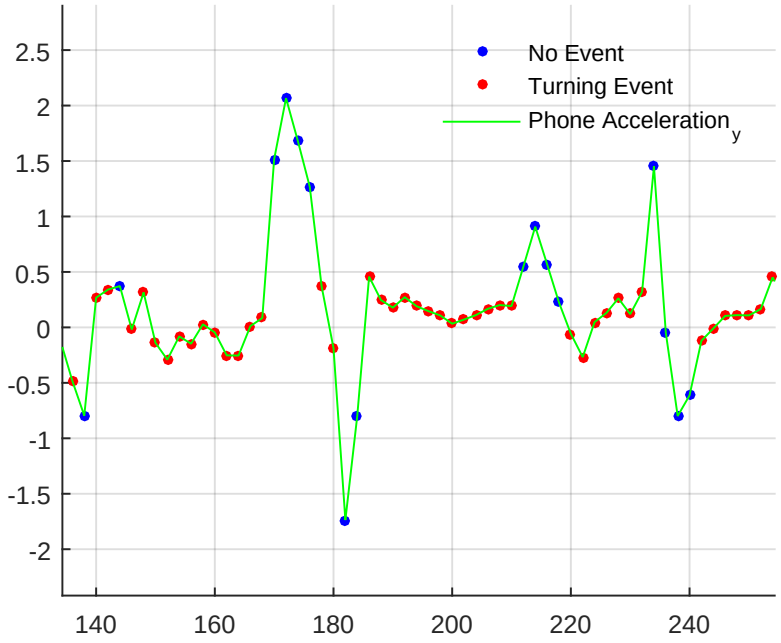

Figure 4.7: The Baseline Clustered data sample for turn events.

Unfortunately, to the best of our knowledge, there was no other comparable work that also detects turns and that has published their accuracy results.

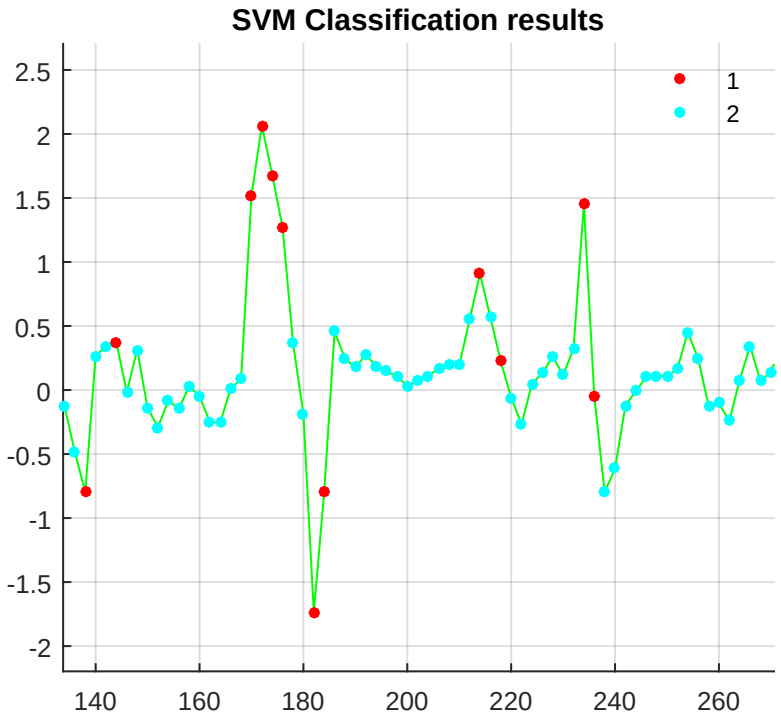

Figure 4.8: The result of SVM classification on the same sample of data for turn events. Points represent windows in the data, and the one in red (1), shows turns events the one in blue (2), shows non-turns.

Clustering and training on PSD_x	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	17	0	26	1	0.654	0.791
N-B	18	1	26	0.994	0.692	0.816
C-T	15	0	26	1	0.577	0.732
KNN	13	1	26	0.994	0.500	0.665

Table 4.10: The results of the classification on the experiment data on the acc_x axis in the first experiment. The baseline data used, was the clustered PSD_x feature of the data.

Clustering and training on PSD_x	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	36	0	40	1	0.900	0.947
N-B	34	1	40	0.997	0.850	0.918
C-T	30	0	40	1	0.750	0.857
KNN	31	0	40	1	0.775	0.873

Table 4.11: The results of the classification on the experiment data on the acc_x axis in the second experiment. The baseline data used, was the clustered PSD_x feature of the data.

Clustering and training on PSD_x	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	48	7	48	1	0.857	0.923
N-B	48	31	48	0.997	0.607	0.755
C-T	48	17	48	1	0.738	0.849
KNN	42	23	48	0.875	0.646	0.743

Table 4.12: The results of the classification on the experiment data on the acc_x axis in the third experiment. The baseline data used, was the clustered PSD_x feature of the data.

Clustering and training on PSD_x	**True Brake Events**	**False Brake Events**	**Actual Brake events**	**Recall (Sensitivity)**	**Precision**	**F_1-Score**
SVM - Linear	58	5	60	0.967	0.921	0.943
N-B	59	13	60	0.983	0.820	0.894
C-T	60	0	60	1	1	1
KNN	48	13	60	0.800	0.787	0.793

Table 4.13: The results of the classification on the experiment data on the acc_x axis in the forth experiment. The baseline data used, was the clustered PSD_x feature of the data.

Clustering and training on PSD_x	True Brake Events	False Brake Events	Actual Brake events	Recall (Sensitivity)	Precision	F_1-Score
SVM - Linear	32	5	35	0.970	0.865	0.914
N-B	29	6	35	0.879	0.829	0.853
C-T	35	6	35	0.875	0.853	0.864
KNN	31	5	35	0.914	0.861	0.887

Table 4.14: The results of the classification on the experiment data on the acc_x axis in the fifth experiment. The baseline data used, was the clustered PSD_x feature of the data.

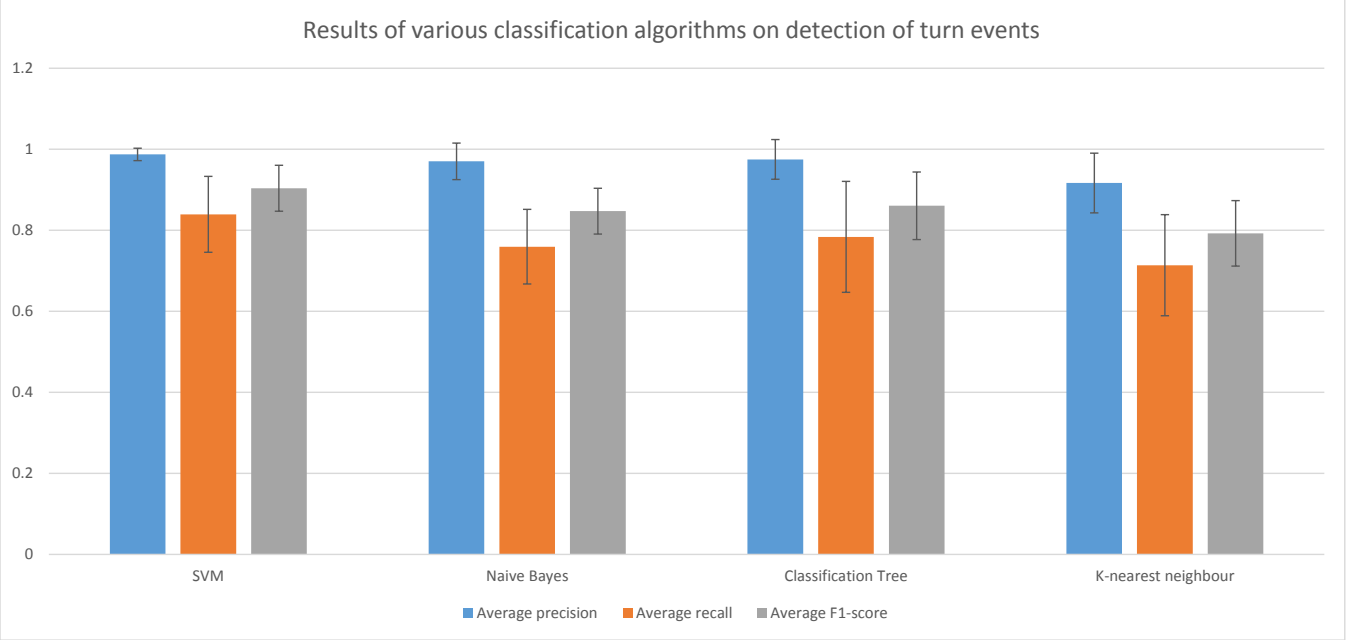

Figure 4.9: Chart of the average results with %95 confidence interval of the four different classification algorithms on the PSD_x feature of clustered data from acc_x to detect turn events.

Classification Algorithm	**Average precision**	**Average recall**	**Average F_1-score**
SVM	0.987 ± 0.015	0.839 ± 0.093	0.904 ± 0.056
Naive Bayes	0.970 ± 0.045	0.759 ± 0.092	0.847 ± 0.056
Classification Tree	0.975 ± 0.049	0.784 ± 0.137	0.860 ± 0.083
K-nearest neighbour	0.916 ± 0.073	0.714 ± 0.124	0.792 ± 0.080

Table 4.15: The average results with %95 confidence interval of the four different classification on PSD_x feature of clustered data from acc_x. SVM classified data, showed the most promising results of turn detection with the highest measures.

4.3.3 PerfDrive Case Study

The details of PerfDrive mobile app assessment of driving task was covered in Chapter 3. In this section, a progressive analysis of all the data that have been gathered over time in our experiment is put forth.

The series of experiments was conducted by one driver:

- Age = 25
- Driver license = G2
- Gender = male
- Driving experience = 7 years
- Vehicles used = Honda Civic 2002 and Hyundai Elantra 2015

As mentioned, a large amount of data had been previously gathered for the evaluation of the detection algorithms in Event Detection component. We employed the same data this time to put forth a progressive study of performance measures over time. Unfortunately, we faced limitations in acquiring weather data; therefore, it was not used in this study. These performance data sets' raw values are depicted in Table 4.16 and Table 4.17.

Using Table 3.5, the normalised performance measure for each indicator is achieved in Table 4.18 and Table 4.19, consecutively. These measures' range are between $[0, 1000]$. It should be recalled that a higher number for a measure means a lower actual performance.

Date	2015_03_30 04_16_43	2015_03_30 10_30_57	2015_03_30 13_51_29	2015_04_02 00_42_53	2015_04_17 09_13_41	2015_06_14 00_24_58
Acceleration	0	0	0.003	0	0.009	0.002
Deceleration	0	0.003	0	0	0	0
Turn	0	0.003	0	0.002	0.003	0
Speeding	2.799	0.129	0.529	0.004	0.363	0.157
Duration	0	0	0	0	0	0
Time	452	0	319	296	446	866
Breaks	0	0	0	0	0	0
Phone	0	0	0	0	0	0
Weather	0	0	0	0	0	0

Table 4.16: The progressive results of our experiments with data on the values recorded for each one.

The line chart in Fig. 4.10 provides the overall trend of performance indicator measures and the overall index as well. A large amount of information can be perceived here. There are repeating patterns of sharp decelerations/accelerations and also turns. It can

Date	2015_06_14 - 00_40_54	2015_06_15 - 14_20_22	2015_06_15 - 14_45_37	2015_06_15 - 15_39_32	2015_06_15 - 16_01_48
Acceleration	0	0.008	0.006	0.003	0
Deceleration	0.011	0.005	0.015	0	0
Turn	0.103	0.003	0.001	0	0
Speeding	0	0.392	0.991	1.727	1.182
Duration	0	0	0	0	0
Time	740	577	0	0	0
Breaks	0	0	0	0	0
Phone	0	0.015	0	0	0
Weather	0	0	0	0	0

Table 4.17: The progressive results of our experiments with data on the values recorded for each one, continued here with the normalized first series of experiments data and also the DPI value.

Date	2015_03_30 04_16_43	2015_03_30 10_30_57	2015_03_30 13_51_29	2015_04_02 00_42_53	2015_04_17 09_13_41	2015_06_14 00_24_58
Acceleration	0	0	10	0	15	10
Deceleration	0	45	0	0	0	0
Turn	0	30	0	30	30	0
Speeding	100	40	80	0	60	40
Duration	0	0	0	0	0	0
Time	100	0	70	60	100	100
Breaks	0	0	0	0	0	0
Phone	0	0	0	0	0	0
Weather	0	0	0	0	0	0
Per. Index Score	200	115	160	90	205	150

Table 4.18: The progressive results of our experiments with data on the values recorded for each one, continued here with the normalized second series of experiments data and also the DPI value.

be assumed that these are not going to change in the future and this small amount of variation will be found to repeat again.

The time score shows driving at unsuitable times of the day, such as late at night. This data also seems to be consistent with DPI score as well. While the time score improved in the last quarter of the driving experiments due to no occurrence of badly timed driving, the speeding score increased. This could be the result of using the newer Hyundai Elantra car instead of the older Honda Civic, since it was around that time that the switching of the cars for the experiments took place.

Date	2015_06_14 - 00_40_54	2015_06_15 - 14_20_22	2015_06_15 - 14_45_37	2015_06_15 - 15_39_32	2015_06_15 - 16_01_48
Acceleration	0	15	15	10	0
Deceleration	75	60	75	0	0
Turn	60	20	20	0	0
Speeding	0	60	80	100	80
Duration	0	0	0	0	0
Time	100	100	0	0	0
Breaks	0	0	0	0	0
Phone	0	80	0	0	0
Weather	0	0	0	0	0
Per. Index Score	235	335	190	110	80

Table 4.19: The progressive results of our experiments with data on the values recorded for each one, continued here with normalized first series of experiments data and also DPI value.

Another argument could be that by having increased speed (and speeding as well), the measures for deceleration, acceleration, and turns have improved. Also it is important to note that except for some outlier results near the end, the overall score, or DPI, which is in green, shows a very slight declining pattern. The decline in the DPI measure illustrates a gradual improvement of the driver's performance over the length of this case study.

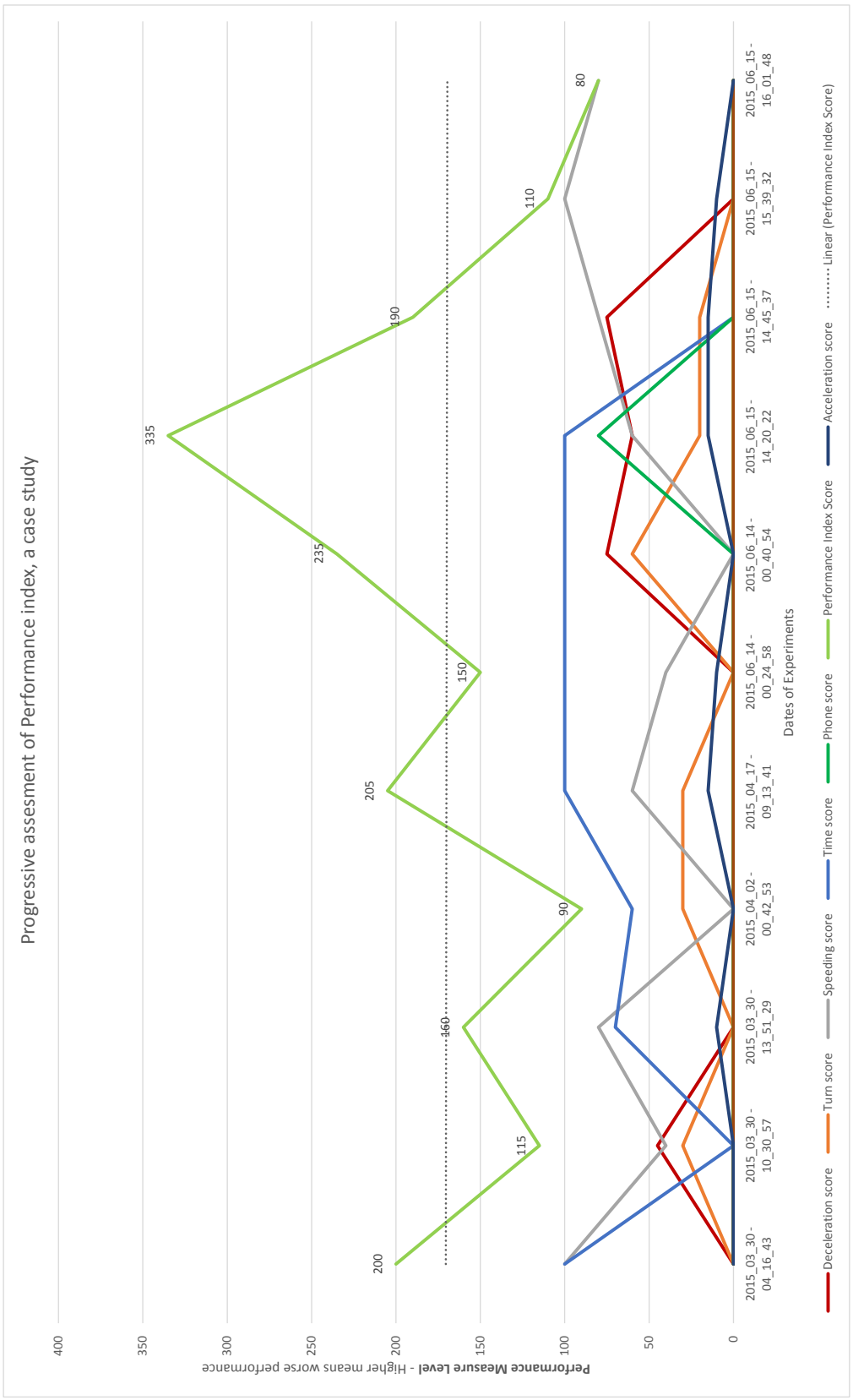

Figure 4.10: Line charts demonstrating progressive assessment of the performance indicators used for calculating the DPI score and performance case study.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

By employing the latest technology in the field of mobile sensing, the auto-insurance industry is moving toward alternative insurance plans such as pay-as-you-drive (PAYD) and usage-based (UBI) plans,[14] which take into consideration the current behaviour of drivers as opposed to their past history. This new method is made possible by collecting data from users on the road and using them as measures, including how drivers accelerate, brake or turn, how fast they drive, the duration of trip, the time and weather when they drive, etc.

In this thesis, a new framework was proposed for sensing, detecting events and measuring the performance of driving. The framework was composed of several components–namely the Fusion Component (fusion of service and sensors), the Event Detection Component, and the PerfDrive application.

Event Detection was one of the major components of the proposed framework. The contribution of this work on this component, was to propose a new machine-learning hybrid approach for the detection of acceleration-based events. Machine learning-based algorithms are more robust in comparison to the threshold-based approaches. Our novelty in detection was to consider the frequency-domain features of data–namely PSD. Results with better accuracy, comparable to the detection rate in similar works, were obtained.

PerfDrive is aimed at filling the gap that exists in the related literature, mainly improving the accuracy of event detection and improving the limited scope of inputs that similar systems use for driving assessment. The PerfDrive application, in contrast, employs data from services and context data in addition to sensor data.

Another common trait found in the current literature is the categorization of driving events to either aggressive or non-aggressive ones, which consequently contributes to a black and white driving-style categorization. The case study of the proposed framework, further enhanced the performance measurement by proposing a performance index-based model with continuous categorization. A modified version of the Kodak Safety Performance Index to design the performance index was used. The measures taken into consideration in this work were the following: weather, time of day, driving duration, number of sharp

turns, sharp accelerations and brakes, instances of driving faster than the speed limit, and number of phone usage when driving. These measures were normalized over time before being fed into the framework.

The system, in its current implementation form, has some limitations. The inputs are either from sensors or services, so the system is limited to their quality and their sampling rate. Also a major concern is the periods that Location data become unavailable, for example, when entering a tunnel. For the movement bearing, one solution is to use the last received value, since in reality the value of bearing does not change as often as other inputs. For the speed value, two solutions can considered: the use of OBD-derived speed or considering both the last received value and the integration of accelerometer as an approximation. To provide an enterprise solution, restrictions of service data should also be considered. Hopefully most service data providers would already have enterprise solutions that meet the requirements needed for real-time assessment of driving with payment of fees.

5.2 Future Work

Driving is a very complex process, and measuring its performance is neither a simple nor a trivial task. Many different factors contribute to its level of measurement. We proposed a flexible model with extension capabilities by adding the possible incorporation of new measures from different sources.

One of the goals for designing the framework introduced in this work was the capability of extension. With regards to this goal, the system can be extended further with sensors and services in the near future. There are, meanwhile, many unexplored sources. Sensing with cameras is one example that has been extensively covered in the literature [40, 43]. Cameras have been used to detect drowsiness, stress levels, and distractions–all negatively effecting the performance of drivers [43, 22, 20, 28]. Visual systems can in turn enhance the system by providing an increased number of measures that normal sensors are not capable of measuring or cannot measure easily.

The results of the case study provided here shows that the PerfDrive index works quite well, yet a formal evaluation of its accuracy remains to be undertaken. Two types of evaluation can be performed. The first is thorough evaluating the proposed performance index and its correlation to real life in-vehicle driving tests. This evaluation should compare the measures of our work with related in-vehicle testing to ensure a more rigorous evaluation. The evaluation needs to contain a large number of sample drivers and also requires the wise selection of the measures to be evaluated and compared. It can be decided at a later time whether or not to put more focus on the testing aspect of driving assessment of this framework as the focus of the current system is on the routine monitoring of drivers.

On the other hand, another similar comprehensive analysis on telematics and monitoring can compare these results to the old-fashioned styles of driver-performance assessment used in the automotive industry. This could be done by comparing the results of the

current framework with those of the insurance records of drivers. A comprehensive statistical analysis including correlation tests need to be conducted, and then it should be clear whether or not a connection can be drawn between the two assessment models or not.

Moreover, it could be the case of a comparison and correlation between similar performance evaluations from related industries to see what is the level of connection between the currently proposed method and other calculation methods. This comparison could bring benefits to related industries and in turn to broad sectors of the general public.

Additionally, telematics and the human machine interface design for new generations of vehicles, are rapidly becoming the norm in the mainstream automobile industry [52]. Tools such as Android Auto, CarPlay, and MirrorLink, which provide integration between smartphones and infotainment systems in vehicles are the future of telematics. The PerfDrive framework can be further extended to be compatible with these new tools so that it can take into consideration the safety and performance measures these systems would require. Android Auto is of special interest as it can be easily integrated into the current framework.

Furthermore, an "Open Automotive Alliance" exists, which is an alliance of automobile manufacturers that use Android in their automobiles; therefore, in future, we will observe a further incorporation of sensors and services in the vehicles. Not only Android but also Apple has their own product, CarPlay, that has a comparable group of manufacturers who are interested in implementing it in their products. Our work can easily fit in this framework.

References

[1] Android sensor support, the mathworks, inc. hardware support guide. http://www.mathworks.com/hardware-support/android-sensor.html. Accessed: 2015-05-20.

[2] Geomagnetic Field, google and open handset alliance n.d. android api guide. http://developer.android.com/reference/android/hardware/GeomagneticField.html. Accessed: 2015-05-20.

[3] Location and Maps, google and open handset alliance n.d. android api guide. https://developer.android.com/guide/topics/location/index.html. Accessed: 2015-06-05.

[4] Location Manager, google and open handset alliance n.d. android api guide. http://developer.android.com/reference/android/location/LocationManager.html. Accessed: 2015-06-05.

[5] Sensor Event, google and open handset alliance n.d. android api guide. http://developer.android.com/reference/android/hardware/SensorEvent.html. Accessed: 2015-05-20.

[6] Sensor Manager, google and open handset alliance n.d. android api guide. http://developer.android.com/reference/android/hardware/SensorManager.html. Accessed: 2015-06-05.

[7] Sensor Overview, google and open handset alliance n.d. android api guide. http://developer.android.com/guide/topics/sensors/sensors_overview.html. Accessed: 2015-05-20.

[8] Telematics, insurance bureau of canada. http://www.ibc.ca/sk/auto/buying-auto-insurance/how-auto-insurance-premiums/telematics. Accessed: 2015-05-24.

[9] Bruce Abernethy. Watching brief: If we go beyond the marketing hype, there are real concerns surrounding the plethora of driver assistance and in-car office systems currently being touted as the way of the future.. *Traffic Technology International, June/July*, pages 58–64, 2001.

[10] Jean Andrey, Brian Mills, and Jessica Vandermolen. Weather information and road safety. *Institute for Catastrophic Loss Reduction, Toronto, Ontario, Canada*, 2001.

[11] Department of the Army. *U.S. Army Map Reading and Land Navigation Handbook*. U.S. Army. Lyons Press, 2004.

[12] Ravi Bhoraskar, Nagamanoj Vankadhara, Bhaskaran Raman, and Purushottam Kulkarni. Wolverine: Traffic and road condition estimation using smartphone sensors. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1–6. IEEE, 2012.

[13] Lucienne T.M. Blessing and Amaresh Chakrabarti. *DRM, a Design Research Methodology*. Springer London, 2009.

[14] Jason Bordoff and Pascal Noel. Pay-as-you-drive auto insurance: A simple way to reduce driving-related harms and increase equity. *Hamilton Project Discussion Paper*, 2008.

[15] Maged NK Boulos, Steve Wheeler, Carlos Tavares, and Ray Jones. How smartphones are changing the face of mobile and participatory healthcare: an overview, with example from ecaalyx. *Biomedical engineering online*, 10(1):24, 2011.

[16] Speed Enforcement and Michael D Fagin. Insurance institute for highway safety. 2014.

[17] Brian S. Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. *Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, 2011.

[18] Mohamed Fazeen, Brandon Gozick, Ram Dantu, Moiz Bhukhiya, and Marta C González. Safe driving using mobile phones. *Intelligent Transportation Systems, IEEE Transactions on*, 13(3):1462–1468, 2012.

[19] Insurance Institute for Highway Safety (IIHS). Beginning drivers and crash risk. 2015.

[20] I Garcıa, S Bronte, LM Bergasa, J Almazan, and J Yebes. Vision-based drowsiness detector for real driving conditions. In *2012 Intelligent Vehicles Symposium Alcalá de Henares*, pages 618–623, 2012.

[21] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[22] Richard Grace, Vicky E Byrne, Damian M Bierman, Jean-Michel Legrand, David Gricourt, Robert K Davis, James J Staszewski, and Brian Carnahan. A drowsy driver detection system for heavy vehicles. In *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE*, volume 2, pages I36–1. IEEE, 1998.

[23] DE Haigney, RG Taylor, and SJ Westerman. Concurrent mobile (cellular) phone use and driving performance: task demand characteristics and compensatory processes. *Transportation Research Part F: Traffic Psychology and Behaviour*, 3(3):113–121, 2000.

[24] SAS Institute Inc. Sas, telematics: How big data is transforming the auto insurance industry. 2013.

[25] Anick Jesdanun. Review: Nexus 5 delivers basics at great price. *Sci-Tech Today*, 11 2013.

[26] Derick A Johnson and Mohan M Trivedi. Driving style recognition using a smartphone as a sensor platform. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1609–1615. IEEE, 2011.

[27] David M Kiser and J Grant Esler. Kodak's safety performance indexinga tool for environmental improvement. *Environmental Quality Management*, 5(1):35–49, 1995.

[28] Matti Kutila, Maria Jokela, Gustav Markkula, and Maria Romera Rué. Driver distraction detection with a camera vision system. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 6, pages VI–201. IEEE, 2007.

[29] Alex Laurie. Telematics-the new auto insurance., 2012. Retrieved September 13, 2012, from http://www.towerswatson.com/assets/pdf/4125/1101-Telematics-FIN.pdf.

[30] Michael G Lenné, Thomas J Triggs, and Jennifer R Redman. Time of day variations in driving performance. *Accident Analysis & Prevention*, 29(4):431–437, 1997.

[31] MATLAB. *Version 8.5 (R2015a) and Statistics and Machine Learning Toolbox Version 10.0*. The MathWorks Inc., Natick, Massachusetts, 2015.

[32] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *Proc. 23rd USENIX Security Symposium (SEC14), USENIX Association*, 2014.

[33] Greg Milette and Adam Stroud. *Professional Android sensor programming*. John Wiley & Sons, 2012.

[34] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[35] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. Eucalyptus : A technical report on an elastic utility computing architecture linking your programs to useful systems, 2008.

[36] Road Safety Policy Office Vehicles Safety Policy & Education Branch Ministry of Transportation. *Ontario Road Safety Annual Report*. Ontario Ministry of Transportation, available from http://www.ontario.ca/orsar, 2011.

[37] Johannes Paefgen, Flavius Kehr, Yudan Zhai, and Florian Michahelles. Driving behavior analysis with smartphones: insights from a controlled field study. In *Proceedings of the 11th International Conference on mobile and ubiquitous multimedia*, page 36. ACM, 2012.

[38] Johannes Paefgen, Thorsten Staake, and Frederic Thiesse. Resolving the misalignment between consumer privacy concerns and ubiquitous is design: The case of usage-based insurance. 2012.

[39] P Philip, J Taillard, MA QUERA-SALVA, B Bioulac, and T Åkerstedt. Simple reaction time, duration of driving and sleep deprivation in young versus old automobile drivers. *Journal of Sleep research*, 8(1):9–14, 1999.

[40] Michelle M Porter and Michael J Whitton. Assessment of driving with the global positioning system and video technology in young, middle-aged, and older drivers. *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences*, 57(9):M578–M582, 2002.

[41] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes in C*, volume 2. Cambridge university press Cambridge, 1996.

[42] Training Resources, Data Exchange Performance-Based Management Special Interest Group, United States. Assistant Secretary for Defense Programs. Special Project Group, Analysis United States. Department of Energy. Office of Operating Experience, Feedback, and Oak Ridge Associated Universities. *How to Measure Performance: A Handbook of Techniques and Tools*. Oak Ridge Associated Universities, 1995.

[43] Maria Rimini-Doering, Dietrich Manstetten, Tobias Altmueller, Ulrich Ladstaetter, and Michael Mahler. Monitoring driver drowsiness and stress in a driving simulator. *Driving Assessment*, 2001.

[44] William Swokowski. *Calculus with Analytic Geometry*. Prindle, Weber & Schmidt, 1979.

[45] Rosolino Vaiana, Teresa Iuele, Vittorio Astarita, Maria Vittoria Caruso, Antonio Tassitani, Claudio Zaffino, and Vincenzo Pasquale Giofrè. Driving behavior and traffic safety: an acceleration-based safety evaluation procedure for smartphones. *Modern Applied Science*, 8(1):p88, 2014.

[46] Guido van Rossum et al. The python language reference. python software foundation. Available at http://www.python.org, 2015. Accessed: 2015-03-10.

[47] Chris Veness. Calculate distance and bearing between two latitude/longitude points using haversine formula in javascript. *Movable Type Scripts*, 2011.

[48] Jules White, Chris Thompson, Hamilton Turner, Brian Dougherty, and Douglas C Schmidt. Wreckwatch: Automatic traffic accident detection and notification with smartphones. *Mobile Networks and Applications*, 16(3):285–303, 2011.

[49] Peter IJ Wouters and John MJ Bos. Traffic accident reduction by monitoring driver behaviour with in-car data recorders. *Accident Analysis & Prevention*, 32(5):643–650, 2000.

[50] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[51] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM, 1999.

[52] Shuo Yuan. Human machine interface design for next generation of vehicle. Master's thesis, University of Illinois at Urbana-Champaign, 5 2014.

[53] Li Zhang, Parth H Pathak, Muchen Wu, Yixin Zhao, and Prasant Mohapatra. Accelword: Energy efficient hotword detection through accelerometer. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 301–315. ACM, 2015.